A Utility-Preserving and Scalable Technique for Protecting Location Data with Geo-Indistinguishability

Ritesh Ahuja
University of Southern California
riteshah@usc.edu

Gabriel Ghinita
University of Massachusetts Boston
gabriel.ghinita@umb.edu

Cyrus Shahabi
University of Southern California
shahabi@usc.edu

ABSTRACT

Location-based apps provide users with personalized services tailored to their geographical position. This is highly-beneficial for mobile users, who are able to find points of interest close to their location, or connect with nearby friends. However, sharing location data with service providers also introduces privacy concerns. An adversary with access to fine-grained user locations can infer private details about individuals. *Geo-indistinguishability (GeoInd)* adapts the popular *differential privacy (DP)* model to make it suitable for protecting users' location information. However, existing techniques that implement GeoInd have major drawbacks. Some solutions, such as the planar Laplace mechanism, significantly lower data utility by adding excessive noise. Other approaches, such as the optimal mechanism, achieve good utility, but only work for small sets of candidate locations due to the use of computationally-expensive linear programming. In most cases, locations are used to answer online queries, so a quick response time is essential. In this paper, we propose a technique that achieves GeoInd and scales to large datasets while preserving data utility. Our central idea is to use the composability property of GeoInd to create a multiple-step algorithm that can be used in conjunction with a spatial index. We preserve utility by applying accurate GeoInd mechanisms and we achieve scalability by pruning the solution search space with the help of the index when seeking high-utility outcomes. Our extensive performance evaluation on top of real location datasets from social media apps shows that the proposed technique outperforms significantly the benchmark in terms of utility and/or computational overhead.

1 INTRODUCTION

The unprecedented growth in the area of mobile apps allows users to enjoy personalized services and receive information customized to their locations. In return for sharing their locations with the service provider, users can find restaurants and shopping malls nearby, can plan their travel itinerary with ease, or can connect with nearby friends. While the benefits of personalized location services are clear, there are also increasing risks associated with the sharing of fine-grained individual locations. A significant body of research [15–17, 21] shows that uncontrolled sharing of users' whereabouts can lead to a wide range of attacks, from stalking and assault, to various privacy breaches that may disclose sensitive personal details such as one's health status, political or religious orientation, etc.

The need for protecting users' locations has been the subject of intense study by the research community for more than a decade. Initial approaches considered cloaking of user locations to decrease the precision of coordinate reporting. However, it has been shown [15, 17] that this category of solutions does not provide sufficient protection, especially when dealing with sophisticated adversaries with access to background knowledge. Another category of techniques uses encryption [15]. While the privacy strength achieved with encryption is high, there are two drawbacks: first, only simple queries (e.g., nearest-neighbors) can be answered; second, the computational overhead of processing encrypted queries is high and requires extensive server-side changes.

The prominent model of *differential privacy (DP)* [13] has become the de-facto standard for data protection in the last few years. While there is work that shows how location can be protected with DP, these solutions (and in fact, the model itself) assume an aggregate release, where data are pooled together from a population of users over a period of time, and then aggregates are disclosed covering various regions of the dataspace. The objective of this publication model is to hide the presence of an individual in a released dataset, but cannot be used in operational mode (i.e., to protect the location attributes of a specific user asking a query). The more recent model of *Geo-indistinguishability (GeoInd)* [1, 2, 5] extends DP with a new distinguishability metric [6] which captures precisely the operational setting, and prevents the association of a user with an exact location. Specifically, GeoInd adds random noise to the actual user location in a way that prevents an adversary from inferring with high probability the user's whereabouts, *regardless* of the amount of background knowledge available to the attacker. Its powerful semantic protection guarantees derived from DP make GeoInd the only viable approach available at the present time for protecting locations in the operational setting[1].

Despite GeoInd being a promising model, existing techniques that implement it have some important drawbacks. We provide a simplified argument, without going into technical details, as the formal aspects of GeoInd will only be introduced in Section 2. In essence, GeoInd achieves protection by adding noise, but noise also decreases data utility. In addition, the protection achieved is independent of the adversary's background knowledge, further referred to as *prior*. However, an interesting result in [2] shows that, if one is aware of the adversary's prior, then it is possible to construct a GeoInd mechanism that improves utility considerably, while still providing protection for *any* prior. This result is important, because in practice locations where a user is expected to be are not random. For instance, datasets derived from social media apps show that users *check in* (i.e., report their location) at a set of well-defined *points of interests (POI)*. This discrete set of POI, combined with some other background knowledge factors (e.g., popularity of various POI) effectively functions as the adversary's prior. One can construct mechanisms for enforcing GeoInd in two ways: at one extreme, completely ignore the prior, and simply generate a reported location by adding (planar) Laplace noise to the actual location [1]. This approach is very efficient computationally, but can yield poor utility by generating large

[1] Another model exists which functions in the operational setting [24], but it only works against a well-defined set of adversarial prior knowledge. Constructing such a detailed prior is often not feasible in practice.

noise. At the other extreme, one can take into account the prior and attempt to add the *optimal* amount of noise to any possible user location such that the expected distance between actual and reported locations is minimized (while still preserving GeoInd). The latter is called *optimal mechanism* [2] and is implemented using linear programming (LP).

Consider a regular grid on top of a set of locations of interest in a city area (grid partitioning is used for ease of presentation, the concept we convey remains valid for any set of discrete, or *logical* locations [12]). Each cell has a certain prior value, corresponding, for instance, to the weighted popularity of all POI in the respective cell. An optimal GeoInd protection mechanism will produce an output given by a linear program that considers all possible combinations of actual and reported cells, with specific constraints related to location protection. The computational complexity is cubic to the number of cells. Even for a relatively small search space, such as $12 \times 12 = 144$ cells, the execution time for LP solvers can be in the order of hours (we provide exact measurement results in Section 6). Such an approach is too slow.

Our proposed approach uses a multi-step algorithm that applies GeoInd recursively along a data indexing structure, according to the *composability* property of DP (see Section 2 for details). We illustrate this in Figure 1, on a three-level index. The actual location is the black dot. Level A consists of nine cells, and assume $A5$ is selected by the mechanism in the first step. In step two, we "zoom in" cell $A5$, and re-apply the mechanism on top of a four-cell grid this time. Finally, at the third level, $C3$ is chosen for release. The privacy budget is split across the three levels. Our approach can work with a relatively fine-grained grid at the leaf level; in this example, at the leaf level we obtain 144 cells, same as in the non-hierarchical case. However, in our case the size of the problem at each step consists of nine, four and four cells, respectively, so our approach will perform much faster. Also, since index node selection is performed at each step according to the output of the mechanism at the previous index level, GeoInd is preserved. Note that, the actual location may fall outside the selected cell at each level, but that is less likely to occur at the higher (i.e., coarser) index levels, so utility will not suffer significantly.

Although the underlying concept is simple, our approach raises a number of difficult challenges. Specifically, one must take carefully into account several factors that affect utility and performance, such as: how to determine the parameters of the index, such as height and fan-out; and how to decide how to split the privacy budget across distinct index levels. We provide an analytical model to synthesize important relationships between system parameters on one hand, and performance metrics such as utility and computational overhead on the other.

Our specific contributions include:

- We identify important drawbacks of existing techniques for geo-indistinguishability in terms of utility and/or computational overhead.
- We propose a multi-step algorithm that applies GeoInd mechanisms in conjunction with an index data structure in order to prune the search space when seeking optimal solutions.
- We develop an analytical cost model to characterize the utility-performance trade-off, and to select an appropriate set of parameter values for our approach.
- We perform an extensive experimental evaluation to measure utility and execution time on real datasets, and we show that the proposed approach significantly outperforms the benchmark.

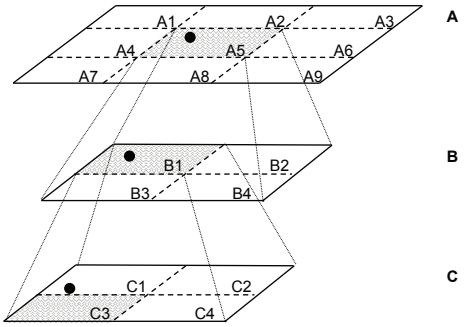


Figure 1: Multi-Step Approach Overview

The rest of the paper is organized as follows: Section 2 provides essential background information, followed by an overview of our approach in Section 3. We provide the details of our algorithm in Section 4. We derive an analytical model for performance characterization in Section 5 and we present a detailed experimental evaluation in Section 6. We survey related work in Section 7 and conclude with directions for future research in Section 8.

2 PRELIMINARIES

We introduce fundamental concepts used in the rest of the paper, such as the privacy model we adopt, and the mechanisms that implement it. We also discuss utility metrics, as well as the important property of *composability* on which our proposed multi-step algorithm for location protection relies.

2.1 GeoInd Definition

Geo-indistinguishability (GeoInd) was first introduced in [1], and extends the popular protection model of differential privacy (DP) [13]. DP is designed to prevent an adversary from learning with significant probability whether an individuals' data is present or not in a dataset. DP is achieved by adding random noise to the result of aggregate queries (e.g., count or sum). The fundamental concept behind DP is to bound the probability of distinguishing between the results of computations performed on *neighboring* datasets, defined as sets of records that differ in at most one entry. More formally, when the Hamming distance between two candidate datasets D_1 and D_2 is 1, then an adversary cannot determine with significant probability whether D_1 or D_2 was used to produce a certain result. The corresponding probabilities P_1 and P_2 are similar within a small multiplicating factor e^{ε}, where ε is called *privacy budget* (lower ε values correspond to tighter privacy settings).

However, DP is not suitable to protect the locations of mobile users in the *online* setting, for two reasons: first, DP works only for aggregate queries, and cannot be directly used for sanitizing individual records; second, DP can hide the presence of an individual record in a dataset, but it cannot protect the attribute values of individual data records, due to the *distinguishability metric* used (Hamming distance). The work in [4] extends DP by proposing a more flexible distinguishability metrics, resulting to the GeoInd definition.

Consider a set $\mathcal{X}$ of possible user locations, a set $\mathcal{Z}$ of possible reported locations (the two sets $\mathcal{X}$ and $\mathcal{Z}$ may coincide), and a distance metric $d_{\mathcal{X}}$. A probabilistic mechanism $K : \mathcal{X} \rightarrow \mathcal{P}(\mathcal{Z})$

takes as input a location in $\mathcal{X}$, and obfuscates it to produce some location $\mathcal{Z}$, which is reported to a location-based service (since K is a probabilistic function, its co-domain is the power-set of $\mathcal{Z}$, as it assigns each output location a probability of being reported). Mechanism K is said to achieve geo-indistinguishability with privacy level ε if for all $x, x' \in \mathcal{X}, z \in \mathcal{Z}$:

$$K(x)(z) \leq e^{\varepsilon d_{\mathcal{X}}(x,x')} \cdot K(x')(z) \quad (1)$$

Intuitively, Eq. (1) specifies that, given a reported value z, an adversary cannot distinguish whether the user location is x or x' by a factor larger than $e^{\varepsilon d_{\mathcal{X}}(x,x')}$. In practice, one appropriate choice for the $d_{\mathcal{X}}$ metric is the Euclidean distance, denoted in the rest of the paper as $d(\cdot,\cdot)$. Effectively, geo-indistinguishability enforces a constraint on the distributions $K(x)$, $K(x')$ produced by two different points x, x'. The authors in [5] also propose a practical interpretation of this definition: for all locations x' within a radius r from x, the user enjoys εr-indistinguishability. For small values of r, the adversary gains little information, i.e., s/he cannot pinpoint the user within a small region of a city (e.g., specific bar, restaurant, building). However, for large r values, the probability of locating the user within a circle of radius r increases, which preserves utility of location reporting. In other words, the mobile user is still able to retrieve information relevant to her current city or neighborhood, at a coarser granularity.

2.2 Utility and Composability

Utility. In order to enforce GeoInd, actual user locations must be perturbed through addition of random noise. Inherently, there is utility loss associated to this process: location-based services will return information relevant to the reported location, instead of the actual one. Ideally, a GeoInd mechanism should add the minimum amount of noise to achieve the GeoInd constraints, so utility is maximized and the deterioration in the quality of service received by the user is constrained. Inspired from previous work [1, 2, 5] we use the following practical measures of utility loss[2]:

- *Euclidean distance d:* this utility metric measures the Euclidean distance between the reported and actual user locations. It is a natural metric to describe the additional distance traveled by the user as a result of location obfuscation (assuming free-space movement). For instance, the nearest-neighbor Italian restaurant of the *reported* location could be several hundred meters away from the user location, even though there is another one situated only meters away from the user.
- *Squared Euclidean distance d^2:* the square of the Euclidean distance between reported and actual locations is also important, as it estimates the number of results that a user receives in a certain area. In practice, the user—knowing that the location will be obfuscated—may ask for more results. For instance, instead of asking for restaurants in a 200 meters radius, the user may increase the range to increase the chance that a nearby POI is actually returned. The size of the result set, and the effort required to filter the results, are estimated to increase linearly with the area of search, so the squared Euclidean distance may be a good predictor of utility loss.

[2]We emphasize that a utility loss metric is a *different* concept than a distinguishability metric. Although the Euclidean distance can serve as both, the purpose of the two types of metrics is different: distinguishability metrics characterize the ability of an adversary to attack location privacy, whereas utility loss metrics measure the decrease in quality experienced by the user.

Composability. An important property of GeoInd (inherited from its parent model DP) is *composability*. Specifically, if two mechanisms are applied in succession with budgets ε_1 and ε_2, the net amount of privacy obtained is equivalent to a privacy budget $(\varepsilon_1 + \varepsilon_2)$. Conversely, given a total privacy budget ε, we can split it into several terms and assign each term to a separate processing step, effectively obtaining a combined multiple-step algorithm that achieves the same amount of protection.

2.3 GeoInd Mechanisms

The GeoInd definition specifies the constraints that a protection technique must satisfy in order to obtain strong privacy guarantees. A GeoInd *mechanism* is a specific construction that achieves the GeoInd requirements. Next, we enumerate the most prominent mechanisms that implement GeoInd, introduced in [1, 5]. These mechanisms achieve various trade-offs in terms of performance and utility.

Planar Laplace Mechanism (PL). The simplest approach to preserving GeoInd is the Laplace mechanism [1]. The key idea is to perturb the exact user location by additive random noise drawn from a bi-variate Laplacian distribution with density defined as:

$$D_\varepsilon(x,z) = \frac{\varepsilon^2}{2\pi} e^{-\varepsilon d(x,z)} \quad (2)$$

Drawing an element from this distribution can be done by (i) switching to polar coordinates, (ii) selecting an angle θ uniformly from $[0, 2\pi)$, and (iii) selecting a radius r from the Gamma distribution $\Gamma_\varepsilon^{-1}(p)$, where $\Gamma_\varepsilon^{-1}(p) = 1 - (1 + \varepsilon p)^{(-\varepsilon p)}$, and p is uniformly chosen from (0,1). Finally, the reported location is $z = x + (r\cos\theta, r\sin\theta)$.

If only a restricted set of reported locations is allowed (e.g., discrete set $\mathcal{Z}$), the location produced by the mechanism can be mapped back to the closest element in the set. Although the Laplace mechanism provides an easy and practical way of achieving geo-indistinguishability, its utility may be low, as it introduces large noise.

Optimal Mechanism (OPT). While the PL mechanism is simple to implement and efficient to execute, it does not provide any optimality guarantees for utility. As a result, its resulting utility may be low in practice. To address this issue, the optimal mechanism of GeoInd (denoted further by *OPT*) has been introduced in [2]. We provide the details of OPT in Section 3.2. In a nutshell, OPT uses information about an adversary's prior knowledge, such as the likelihood of a user being present at a certain location. For instance, a user is more likely to be found in a city center area where there is a high concentration of POI, rather than in a secluded area near a suburb. OPT uses such prior information to produce a reported location z that minimizes utility loss. The disadvantage of OPT is that it has high computational overhead, as it requires solving a linear program with complexity cubic in the number of possible locations. In contrast, our multi-step algorithm allows us to reduce overhead by applying OPT recursively on an index structure.

A remarkable property of GeoInd related to the OPT mechanism is the following: even if the mechanism is tuned for a specific prior, it still preserves the privacy constraint for *any* prior. If the prior is not known by a GeoInd mechanism, then the utility cannot be improved significantly compared to PL. If the prior is known, then it can boost utility. Furthermore, GeoInd is satisfied even if the assumed prior and the adversary's background knowledge are different. In contrast, privacy models such as the one in [24] can

Table 1: Summary of Notations

Notation	Definition
x	*Actual* user location
z	*Reported* user location
$\mathcal{X}, \mathcal{Z}$	Set of actual and reported user locations
$\mathbb{G}, \mathbb{G}_i$	Set of grid cells (global, and at index level i)
$\hat{x}$	*Actual* location cell of user
$\hat{z}$	*Reported* location cell of user
g	Grid granularity
L	Side length of the square spatial region
h	Height of hierarchical grid index
ϵ_i	Privacy budget for grid level i
$\mathcal{B}$	Array storing budget at each index level
$d(x, z)$	Euclidean distance between locations x and z

only provide protection if the assumed prior and the adversary's knowledge coincide.

3 APPROACH OVERVIEW

In Section 3.1 we provide an overview of our system model; in Section 3.2 we introduce a baseline approach that applies the optimal mechanism of GeoInd on top of a regular grid. Table 1 summarizes the notations used in our presentation.

3.1 System Model

We consider an *online* scenario where mobile users are interested in retrieving points of interest relevant to their current location from an untrusted server. Based on the actual user location $x \in \mathcal{X}$, the sanitization algorithm computes a *reported* location $z \in \mathcal{Z}$. Both $\mathcal{X}$ and $\mathcal{Z}$ are assumed to be discrete and finite sets. Discrete locations, also called *logical* locations [12], are often used in the location privacy literature. In practice, this can be achieved by snapping continuous coordinates to an arbitrary granularity grid. We note that, in many existing geosocial network applications, the reported locations take the form of *check-ins* at discrete sets of two-dimensional coordinates corresponding to restaurants, coffee shops or other POIs. Our model is fully compatible with that setting.

In our model, the location sanitization is performed online at the user's mobile device. This approach is made possible by the properties of the GeoInd model, and it is an important advantage compared to other types of solutions that require a trusted centralized service, such as spatial k-anonymity (SKA) [16, 21]. Thus, the system model is much simpler, and it does not rely on unrealistic security assumptions, such as the presence of a trusted third party that performs anonymization, or the presence of a collaborating set of other mobile users who participate in the formation of cloaking regions. Furthermore, there are no changes required on the processing side at the server, which is an advantage compared to approaches that use encryption [15]. However, due to the fact that location protection is performed at the mobile device, the computational overhead incurred by sanitization is a very important concern. In our design, we focus on this constraint.

The mobile device will execute our sanitization technique before reporting its location (i.e., at runtime), and will also download in advance (*offline*) a set of objects that are required to support our technique, such as a set of maps annotated with additional pre-computed information (e.g., the properties of a certain city map and associated details required to construct a balanced index structure, as discussed in Section 5). This offline component is common for many software packages that support location-based apps, e.g., navigation services. Furthermore, in our approach, the amount of data that needs to be downloaded offline is small (in the order of tens of megabytes).

3.2 Baseline Approach

We provide a detailed description of the optimal GeoInd mechanism *OPT* [2] adapted to a regular grid. OPT is used as a building block in our multi-step approach, and also as a baseline in our evaluation. OPT produces the maximum utility achieveable under a given prior while preserving GeoInd. Specifically, given a privacy budget ε, a distinguishability metric $d_{\mathcal{X}}(\cdot,\cdot)$, a utility (or quality) metric $d_Q(\cdot,\cdot)$, and a prior Π defined over set $\mathcal{X}$, OPT determines mechanism K as the solution to the following linear programming problem:

Minimize:

$$\sum_{x \in \mathcal{X}, z \in \mathcal{Z}} \Pi_x \cdot K(x)(z) \cdot d_Q(x, z) \tag{3}$$

Subject to:

$$K(x)(z) \leq e^{\epsilon d_{\mathcal{X}}(x,x')} \cdot K(x')(z) \qquad x, x', z \in \mathcal{X} \tag{4}$$

$$\sum_{z \in \mathcal{X}} K(x)(z) = 1 \qquad x \in \mathcal{X} \tag{5}$$

$$K(x)(z) \geq 0 \qquad x, z \in \mathcal{X} \tag{6}$$

This linear optimization problem can be solved by using standard techniques such as the Simplex or the Interior Point methods. However, the number of linear constraints in the program is $O(|\mathcal{X}|^2 \times |\mathcal{Z}|)$ (or cubic in the total number of locations, assuming that actual and reported locations belong to the same set). As a result, the method is unfeasible even when the set of locations has low cardinality (i.e., several hundred). In previous work [5] the cardinality of $\mathcal{X}$ is reduced by using a coarser grid, and locations in $\mathcal{X}$ are snapped to the centers of the grid cells. Let L denote the side length of the dataset (assumed to be a square, but any rectangular region can be scaled to suit the assumption). Grid granularity g is achieved by splitting the data domain into a regular grid with $g \times g$ cells, each with dimensions $L/g \times L/g$. While this approach reduces computational overhead, it also decreases utility, as all locations are snapped to the coarse grid.

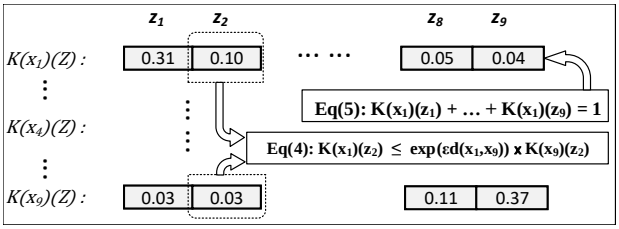


Figure 2: An instance of $K(\mathcal{X})(\mathcal{Z})$ for a 3×3 grid.

Next, we focus on the application of OPT on top of a grid. Denote by $\mathbb{G}$ the set of grid cells, which represent the logical locations from which both actual and reported locations are selected, i.e., $\mathcal{X} = \mathcal{Z} = \mathbb{G}$. The prior is also defined with respect to $\mathbb{G}$. We can abstract an invocation of the OPT mechanism as a function call with the following parameters: $OPT(\epsilon, \mathbb{G}, \Pi, d_Q)$.

To provide an in-depth understanding of how OPT works, we consider the example of a regular grid of granularity $g = 3$, i.e.,

a total of nine cells. We illustrate in Figure 2 the layout of the stochastic matrix $K(\mathcal{X})(\mathcal{Z})$ from Eq.(4). We represent three of the nine rows of matrix K, corresponding to cells x_1, x_4 and x_9. The box at the top of the diagram illustrates the constraint in Eq. (5), representing the normalization condition, namely: the sum of probabilities in each matrix row must be 1 (each input cell must be mapped to some output cell). Likewise, the box in the middle of the diagram illustrates one of the $|\mathcal{X}|^3 = 81$ ε-geo-indistinguishability constraints corresponding to Eq. (4) in the linear program of the optimal mechanism.

The main drawback of OPT is its prohibitively high computational cost. To illustrate this drawback, we show in Figure 3 the trade-off between performance and utility when varying granularity, using a state-of-the-art commercial linear program solver on the Gowalla dataset (please see Section 6 for experimental setup details). As grid granularity increases, the utility improves but at the expense of a sharp rise in computation time. For the next higher granularity $g = 12$ (not shown in the graph), the optimization program was terminated after 24 hours without a solution.

The objective of our approach is to address these conflicting trends between utility and performance. Next, we present our *multi-step mechanism (MSM)* which operates on top of a hierarchical index structure and achieves a good compromise between utility and performance.

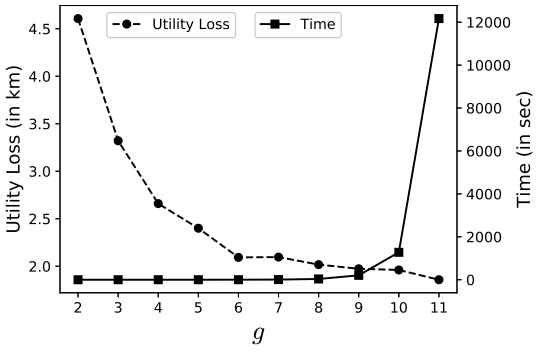


Figure 3: Effect of g on utility and running time of *OPT*

4 MULTI-STEP MECHANISM (MSM)

We introduce the multi-step mechanism (MSM) which operates over a *GeoInd-preserving Hierarchical Index (GIHI)* and transforms the user location according to OPT at each index level. The user inputs the total privacy budget ε to the algorithm, according to her privacy requirement. A separate component of our solution, the budget allocation strategy (discussed in detail in Section 5), determines how to distribute the privacy budget among the index levels. The output of the budget allocation procedure consists of the index height h, and the amount of privacy budget allocated to each level $B_i \in \mathcal{B}$, where $\mathcal{B}$ is the set of budget amounts per level, and $h = |\mathcal{B}|$. In the rest of the section, we focus on how the mechanism operates given the index height and budget splits.

Given granularity g and height parameter h, a GIHI is constructed over a data domain of size L^2 in a top-down fashion[3]. Each intermediate cell points to g^2 cells at the lower level that lie

[3]If the input dataset domain is not square, it can be scaled in advance of executing our algorithm to equalize the range in each dimension.

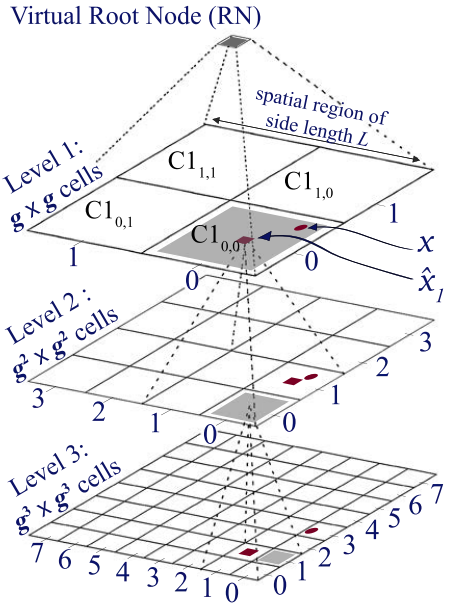


Figure 4: GeoInd preserving Hierarchical Index (GIHI)

inside its spatial extent. Figure 4 illustrates the index structure[4] for $g = 2$ and $h = 3$. The cells in the hierarchical grid are labeled per level, and also within each level, as shown for level 1 in the diagram (to improve readability, we do not include the labels at lower levels). For instance, the leaf cell $C3_{3,0}$ is a child of $C2_{1,0}$, which in turn is a child of node $C1_{0,0}$. MSM iteratively visits each level of the hierarchical structure, starting with the virtual root node that covers the entire region. The root node is not a level in the tree, but it serves the purpose of simplifying the presentation, so that the algorithm can be expressed recursively starting from a single node. We denote as $\hat{x}_i$ the logical location of the user snapped to the center of the enclosing cell in the grid of granularity $g^i \times g^i$ at level i. Likewise, $\hat{z}_i$ denotes the logical location of the user output by the mechanism at level i. We define two simple procedures: *EnclosingCell*(x, i), which returns the cell at level i enclosing the input location x, and *centerOf*(C), which takes as input a cell C and returns the logical location at its center.

Algorithm 1 presents the pseudo-code of our multi-step iterative process of location sanitization. In each iteration, the algorithm takes as input the location $\hat{z}_{i-1}$ selected in the previous iteration (in the first iteration it is set to the center of the virtual root node) and determines the cell that encloses $\hat{z}_{i-1}$, denoted by C (lines 4-6). Next, it translates the user's actual location x to a logical location $\hat{x}_i$ in the current level (lines 7-8) and constructs a partial grid $\mathbb{G}_i$ of g^2 cells within the spatial extent of C. Next (lines 11-13), MSM computes the matrix $K(\mathcal{X}_i)(\mathcal{Z}_i)$ as the solution to a linear optimization problem over the logical locations $\mathcal{X}_i, \mathcal{Z}_i \in \mathbb{G}_i$, consuming the privacy budget ε_i allocated to level i. MSM marks the end of the iteration at the current level by outputting $\hat{z}_i$ as a sample from the distribution $K(\hat{x}_i)(\mathcal{Z}_i)$. These iterations repeat for the entire height of the GIHI, consuming the entire privacy budget $\varepsilon = \varepsilon_1 + ... + \varepsilon_h$. After the final iteration (at the leaf level

[4]For simplicity, we focus in this paper on an hierarchical grid, but the MSM concept applies to any hierarchical data structure without node overlap, e.g., R^+-trees or k-d-trees.

of the index), the output is the sanitized location that is reported to the service provider (line 14).

Note that, the actual location x may fall outside the selected cell at one or more levels of the index. This is an effect of the privacy requirement imposed by GeoInd, and thus a necessary aspect of protecting locations. However, it may also increase utility loss, as the distance between reported and actual cell may be large. To control the amount of utility loss, the objective of our budget allocation strategy (Section 5) is to split the budget in such a way that it is less likely for this event to occur at the higher (i.e., coarser) index levels. If this objective is achieved, utility will not decrease significantly.

Algorithm 1 Multi-Step Mechanism (MSM)

Input: ϵ, g, Π, d_Q
Output: $\hat{z}$

1: **procedure** PRIVATELYREPORTLOCATION(x)
2: $\mathcal{B} \leftarrow$ getGridParameters(ϵ, g)
3: $h \leftarrow |\mathcal{B}|$
4: $\hat{z}_0 \leftarrow RootNode$ ▷ stores output cell of each iteration
5: **for** $i \leftarrow 1, h$ **do**
6: $C \leftarrow EnclosingCell(\hat{z}_{i-1}, i-1)$
7: $\mathbb{G}_i \leftarrow$ set of $g \times g$ cells in the spatial bounds of C
8: $\hat{x}_i = centerOf\,(EnclosingCell(x,i))$
9: **if** $\hat{x}_i \notin \mathcal{X}_i$ **then** ▷ $\mathcal{X}_i, \mathcal{Z}_i \in \mathbb{G}_i$
10: $\hat{x}_i \leftarrow$ a random location in $\mathcal{X}_i$
11: $\epsilon_i \leftarrow \mathcal{B}[i]$
12: $K(\mathcal{X}_i)(\mathcal{Z}_i) \leftarrow OPT(\epsilon_i, \mathbb{G}_i, \Pi(\mathcal{X}_i), d_Q(.))$
13: $\hat{z}_i \leftarrow$ sample from distribution $K(\hat{x}_i)(\mathcal{Z}_i)$
14: **return** location $\hat{z}$ to the service provider

We illustrate the proposed mechanism with a running example over the GIHI structure from Figure 4. The exact location of the user x is depicted as a red-colored dot, while the logical locations (snapped to cell centers) $\hat{x}$ are shown as red colored squares. The shaded cells depict the cells enclosing the output location ($\hat{z}$) of each iteration. The algorithm first takes as input the entire dataspace domain (represented as the root node), and splits it into a grid $\mathbb{G}_1$ consisting of cells $\{C1_{i,j} : 0 \leq i,j \leq 1 \wedge C1_{i,j} \in RN\}$. Next, it maps the user's exact location x to $\hat{x}_1$ at the center of its enclosing cell at level 1, i.e., $C1_{0,0}$. Then, MSM computes $K(\mathcal{X}_1)(\mathcal{Z}_1)$, and outputs a location $\hat{z_1}$ sampled from the distribution $K(\hat{x}_1)(\mathcal{Z}_1)$.

Denote the selected cell as $\hat{z_1} = centerOf(C1_{0,0})$. In the next iteration, $\mathbb{G}_2$ is composed of the set of $g \times g$ cells within the spatial extent of $C1_{0,0}$, which is the enclosing cell of the output in the previous iteration. Location $\hat{z_2}$ is selected as output. Suppose $\hat{z_2}$ =$centerOf(C2_{0,0})$, which implies that for the purposes of the last iteration, the actual location x falls outside the spatial extent of the cell enclosing $\hat{z_2}$. In this situation the user's location is assumed to be a random cell (e.g., $C3_{1,1}$) in $\{C3_{i,j} : 0 \leq i,j \leq 1\}$. The remainder part of the iteration continues by outputting $\hat{z_3}$ =$centerOf(C3_{1,0})$. Finally, $\hat{z_3}$ is reported to the service provider, with a quality loss $d_Q(x, \hat{z_3})$.

A consequence of the grid based discretization scheme is that, the position of every user is always approximated by the center of the enclosing cell before being obfuscated by the mechanism. For example, consider that a user with his exact coordinates at a random location in a 1km^2 cell, requests the nearest bar to his position. In this case, he will receive an answer that is tailored, in the best case, to the center of his cell, which is on average 0.38km [14] away from the current location of the user. It is clear that the situation gets more problematic as the grid cells are larger, i.e., at coarser granularities. This is often the case when applying the conventional optimal mechanism [2] over a coarse grid (as discussed in Section 3.2, OPT can work only for very coarse grids, otherwise the execution time is extremely high). In contrast, our approach operates on a much finer-grained grid at the leaf level, thus gaining significantly in terms of utility. MSM also limits the size of the linear optimization problem, given that there are exactly g^2 logical locations at each step. This enables efficient computation of the linear program, making the overall execution time practical.

We end this section by informally discussing the fact that MSM preserves GeoInd. MSM is a textbook example of applying the *composability* property [19, 20] of differential privacy. The initial iteration of MSM takes as input the real user location, and the final iteration outputs the perturbed location. At each index level i, the OPT mechanism is applied with a fraction ε_i of the total privacy budget ε. The output of each MSM step (i.e., level) is pipelined into the input of the following step (at the next level). According to the composability property introduced in Section 2, MSM satisfies GeoInd with budget $\sum_{i=1}^{h} \varepsilon_i = \varepsilon$.

5 BUDGET ALLOCATION STRATEGY

The utility of MSM depends on important system parameters such as the height of the index and the budget allocation across levels. In this section, we provide an analytical cost model of utility that guides our decision on how to choose GIHI parameters. We assume we are given as input the size of the data domain specified as side length L, the grid granularity g (corresponding to a fanout of g^2 at each level), and the total budget ε. The objective is to determine index height h and the budget allocation ε_i for each level $i, 1 \leq i \leq h$.

A fundamental factor that guides our allocation strategy is the observation that if the actual location is mapped to a reported location in another cell, the utility loss is likely to be larger when the grid cell size is also large. Consequently, the utility loss is much larger when this event occurs near the root of the index, compared to the case when it occurs at the leaf level. Intuitively, the impact on utility of the event occurring at level i is g times higher than the same event occurring at level $i+1$. Consider a stochastic obfuscation mechanism that satisfies geo-indistinguishability, and denote its probability of mapping location (i.e., cell) x to z by $Pr[z|x]$. The probability density function of the output is indeed the distribution $K(x)(z)$, as discussed in Section 2. The proposed budget allocation strategy aims to precisely control $Pr[x|x]$, i.e. the probability of reporting cell x when the actual location is also within x. Based on the earlier observation, to reduce utility loss it is important to maintain $Pr[x|x]$ high at the upper levels of the index.

To calculate $Pr[x|x]$ precisely, we can solve the $|\mathcal{X}|^3$ constraints in the linear optimization program of Eq. (3), according to the input prior Π. However, estimating $Pr[x|x]$ in isolation poses a challenge, since it is not feasible to narrow down the effect on a single location of all other cells (as per Eq. (4)). Hence, we devise a method to estimate this value to an arbitrary precision. We denote the approximation of $Pr[x|x]$ as $\Phi(x)$, and compute it by estimating its complement, i.e., the probability of mapping x to another grid cell.

$$\Phi(x) = \frac{1}{\sum_n \exp\left(-\frac{\varepsilon L}{g}\sqrt{n}\right)} \qquad n = a^2 + b^2, (a,b) \in \mathbb{Z}^2 \tag{7}$$

where $\mathbb{Z}^2$ denotes the 2-dimensional infinite integer lattice whose points are tuples of integers. Switching to coordinates with origin at zero, and defining $\hat{0} \triangleq x$, the denominator can be understood to be the sum over the exponents of the *multiplicative distances* between $Pr[\hat{0}|\hat{0}]$ and $Pr[\hat{0}|\hat{n}]$. $\Phi(x)$ is constrained to the integer lattice to ensure that the probability mass of the entire cell is assigned to the center of the cell, giving us a close estimate of our desired probability.

While $\Phi(x)$ needs to be summed over the infinite lattice, the function $T(\varepsilon, g) \triangleq \sum_n \exp(-\varepsilon L/g\sqrt{n})$ converges quickly. In order to approximate this value within a factor of $\exp(-N)$ for any arbitrarily large N, we need only $O(N^2 f/\varepsilon L)$ terms. However, when the value of ε is small, which is often the case in differential privacy literature (with values of $\varepsilon \leq 0.5$ being the most common settings to achieve a reasonable privacy protection), straightforward computation of this value can be prohibitive. To this end, we can apply the two-dimensional Poisson summation to expand the series, followed by taking its Fourier transform, to obtain an efficient approximation when $0 \leq \varepsilon \leq \frac{2\pi g}{L}$:

$$T(\epsilon, g) = \frac{2\pi g^2}{\varepsilon^2 L^2} + \sum_{k=1}^{\infty} c_{2k-1} \left(\frac{\varepsilon L}{g}\right)^{2k-1} \tag{8}$$

where the coefficients $c_1, c_3, c_5, \ldots$ are given by

$$c_{2k-1} = 4\binom{-3/2}{k-1}(2\pi)^{-2k}\zeta\left(k+\frac{1}{2}\right)L\left(k+\frac{1}{2}, \chi_4\right). \tag{9}$$

where ζ is the Riemann zeta function [23] and $L(\cdot, \chi_4)$ is the Dirichlet L-series[25] of the non-principal Dirichlet character $\chi_4 \pmod 4$, which is known to converge absolutely as

$$L(s, \chi_4) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)^s} = 1 - \frac{1}{3^s} + \frac{1}{5^s} - \frac{1}{7^s} + - \cdots. \tag{10}$$

Since we can approximate $T(\varepsilon, g)$ efficiently, the same is true of $\Phi(x)$. In order to configure the budget at each level of the grid, we formulate our problem as follows:

PROBLEM 1. *Given a desired value of $Pr[x|x]$ as ρ, and the grid parameters: side length L, and the granularity g, estimate the minimum budget ε_i that ensures at least ρ percent chance of remaining within the boundaries of the current cell at level i. The budget ε_i can be calculated according to the following optimization problem:*

Minimize:

$$\text{Privacy Budget } \varepsilon \tag{11}$$

Subject to:

$$\begin{aligned} \left(\sum_n \exp\left(-\frac{\varepsilon L}{g}\sqrt{n}\right)\right)^{-1} - \rho \geq 0, \\ \varepsilon > 0, \\ 0 \leq \rho \leq 1 \end{aligned} \tag{12}$$

Since $T(\varepsilon, g)$ can not be written in a closed form, solving directly for ε is not achievable. Nevertheless, the expression in Eq. (12) is monotonic in ε, hence we can utilize a simple branch-and-bound technique [10] to efficiently get an answer with arbitrary precision.

Algorithm 2 summarizes the functionality of the proposed budget allocation strategy. Procedure *getGridParameters* calculates the minimum budget required for each level of the grid, such that with probability at least ρ a location enclosed within cell x is mapped to the same grid cell. For every iteration of the while loop, the procedure determines if there is any budget remaining for additional levels of the multi-step mechanism. Finally, the process halts when it expends all the budget, returning the height h and the budget allocated at each level of the grid.

Algorithm 2 Grid Configuration

Input: ε, g
Output: height h, budgets $\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_h$

1: **procedure** GETGRIDPARAMETERS
2: $v \leftarrow \varepsilon$ ▷ stores the remaining budget
3: $i \leftarrow 1$ ▷ denotes the current grid level
4: $\mathcal{B} \leftarrow \emptyset$ ▷ stores the budget for each level
5: **while** true **do**
6: $\mathcal{B}[i] = \varepsilon_i \leftarrow \max\{\text{solution to Problem 1}, v\}$
7: $v = v - \varepsilon_i$
8: $i = i + 1$
9: **if** $v \leq 0$ **then** ▷ budget expended
10: **return** $\mathcal{B}$

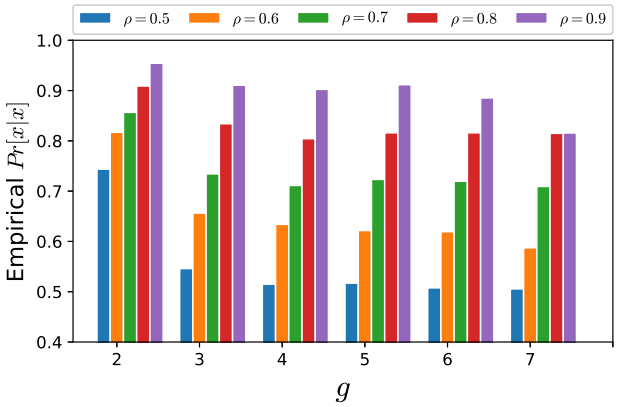


Figure 5: Accuracy of estimated Φ for varying g

We conclude this section by providing a numerical result to validate our analytical model. We illustrate the precision with which our budget allocation algorithm estimates an effective configuration of the grid for MSM (we defer the details of the experimental setup to Section 6.1). We plot the probability $Pr[x|x]$ for various levels of ρ and granularity g using the Gowalla dataset. Figure 5 shows the results, assuming a uniform global prior as input. Excluding the case when the granularity is 2, the predicted value of Φ is within ±5 percent of $Pr[x|x]$ reported in $K(\mathcal{X}, \mathcal{Z})$, thus validating the efficacy of the budget allocation scheme.

6 EXPERIMENTAL EVALUATION

In this section we evaluate the performance of our proposed Multi-Step Mechanism (MSM) in terms of utility and computational overhead. In Section 6.1 we present the details of the experimental setup. We provide the results of comparison with benchmarks in Section 6.2, followed by an in-depth analysis of MSM behavior when varying system parameters in Section 6.3.

6.1 Experimental Setup

Datasets. We use two real datasets collected from the operation of two prominent geo-social apps, namely *Gowalla* and *Yelp*. Each dataset consists of a set of user check-ins. Every check-in is described by a record consisting of user identifier, the latitude and longitude of the check-in location. The *Gowalla* dataset is a subset of the user check-ins collected by the SNAP project [9] from a location based social networking website. In our experiments, to simulate a realistic environment of a city and its suburbs, we focus on check-ins within a single urban area, namely Austin, Texas. In particular, we consider a large geographical region covering a $20\times 20\text{km}^2$ area bounded to the South and North by latitudes 30.1927 and 30.3723, and to the West and East by longitudes -97.8698 and -97.6618. The selected data contains a total of $265,571$ check-ins from $12,155$ unique users during a time period between February 2009 and October 2010. The *Yelp* dataset was made available to the public by Yelp, Inc [3] as part of a dataset challenge, and contains user ratings for various points of interest. Again, to simulate a typical urban area, we utilize location data within the city of Las Vegas, NV and its surroundings. The filtered dataset contains $81,201$ check-ins from $7,581$ unique users within a $20\times 20\text{km}^2$ area bounded between latitudes 36.0645, 36.2442 and longitudes -115.291, -115.069.

Implementation. All algorithms were implemented in C++ on a Ubuntu Linux 14.04 LTS operating system, and executed on an Intel Core 2 Duo 2.0 GHz CPU with 4GB RAM. All data and indices are stored in main memory. For the computation of the linear optimization problem, we utilize the C++ interface of the state-of-the-art commercial linear program solver in the Gurobi Optimization Suite [22]. We used the dual simplex method of solving linear programs throughout our evaluation since it consistently outperformed the primal simplex and interior-point methods in terms of numerical stability.

Prior Modeling. We compute a *global prior* for the Gowalla and Yelp dataset partitions with the help of a regular grid superimposed on top of the data domains. The grid granularity varies to match the grid structure used in each specific experiment. For a given granularity, we count the number of check-ins in every cell relative to total number of check-ins in the entire grid (a similar approach was taken in [2, 5]). The global prior Π describes the behavior of an average user (as a vector of probabilities for each grid cell), and is used in the computation of the optimal mechanism. We store a global prior on the finest effective granularity grid used in the experiments and aggregate this information to obtain priors on coarser grids. This procedure mimics the scenario where the aggregate information about past users of the service is made available from the service provider.

6.2 Comparison with Benchmarks

We evaluate our proposed MSM approach in comparison with two benchmarks: the basic optimal mechanism *OPT* introduced in [2] and described in detail in Section 3.2; and the planar Laplace (PL) mechanism, for which we also include a post-processing (or re-mapping) step by projecting its output to the grid (as discussed in [5]).

All evaluated mechanisms are constructed to satisfy GeoInd with privacy budget ε ranging from 0.1 to 0.9, which is a common range used in the majority of differential privacy work [11]. Recall from Section 2 that a smaller ε value corresponds to stronger privacy. Values higher than 1.0 are typically considered insufficient in terms of privacy (i.e., the attacker's gain in distinguishing among candidate locations becomes significant). We consider several granularities of the grid index structure, ranging from $g = 2$ up to $g = 6$ (recall that for our method, the effective fanout is $g \times g$ at each level). We consider a value range for the probability of hashing the user location in the same cell at a certain level (see Section 5 for details) between $\rho = 0.5$ and $\rho = 0.9$ (default value is set to $\rho = 0.8$). In all experiments, we measure the utility loss experienced by a user of a location-based service over a set of $3,000$ requests randomly selected from the set of check-ins corresponding to each dataset. We consider in turn both Euclidean ($d_Q = d$) and squared Euclidean ($d_Q = d^2$) utility metrics. The default value of ε is set to 0.5.

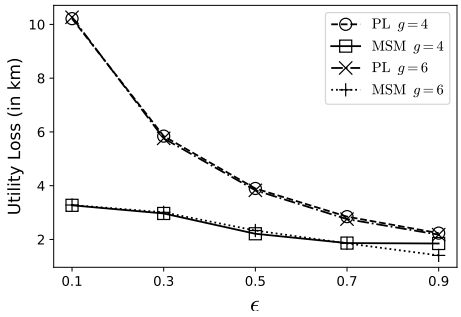
Table 2: MSM comparison against *OPT*, Gowalla dataset

Granularity		Utility Loss (in km)		Time (in sec)	
OPT	MSM	*OPT*	MSM	*OPT*	MSM
4	2	2.29	2.63	0.04	0.008
9	3	1.97	2.22	205.7	0.009
16	4	—	2.02	72hrs+	0.53

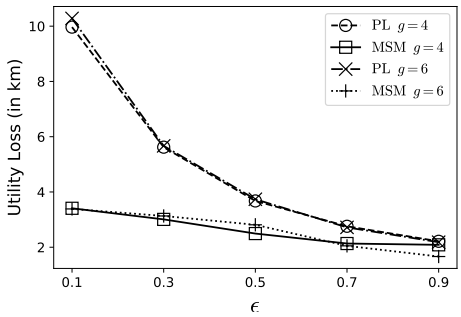
First, we compare the proposed MSM approach with the optimal mechanism OPT. Due to the high overhead of OPT, we are only able to perform the comparison in a restricted setting, for which the linear program completes within reasonable time. Table 2 presents the utility loss and execution time values of OPT and MSM for the same effective grid granularity (e.g., 9×9 granularity for *OPT* corresponds to a 3×3 granularity for our method, where the second level of MSM will have the same number of cells as the OPT grid). We focus on the Gowalla dataset for this experiment. We were not able evaluate the performance of OPT for the granularity of 16 (i.e., 256 locations) because the program did not complete within 72 hours. For 121 locations (i.e., 11×11 grid) OPT took 3.3 hours to complete (see Figure 3). Even for a granularity of 9, the optimal method would be completely unfeasible for online queries, given its running time of 205 seconds, whereas our approach achieves sub-second processing times in all runs. In terms of utility, for the same granularity at the leaf level OPT does outperform MSM. This result is not surprising, since our execution time gain is obtained by pruning the search space. However, the additional loss of MSM compared to OPT is not large. Furthermore, in practice, our approach is able to increase utility by using much finer-grained grids (as we show in the next section), whereas OPT can only function for coarse-grained grids. In the rest of our evaluation, we no longer consider OPT, since it is clearly not feasible in practical settings.

Next, we focus on the comparison between MSM and the planar Laplace mechanism (PL). Figures 6a and 6b plot the utility loss of MSM and PL for varying levels of privacy on both Gowalla and Yelp datasets, using d as utility metric. As expected, utility loss decreases for both mechanisms as ε grows: a larger budget corresponds to a weaker privacy requirement, hence less noise is added. The proposed MSM approach clearly outperforms PL in terms of utility, especially at low ε values, which are more important because they provide appropriate privacy. As ε approaches 1, the utility obtained by the two methods becomes similar, but as mentioned earlier, $\varepsilon = 1$ or higher does not provide sufficient protection. The gain of our approach over PL is more pronounced at low ε settings. For $\varepsilon = 0.1$, MSM utility loss is three times better than the one achieved by PL.

A similar trend can be observed when comparing MSM and PL using the squared Euclidean distance as utilty metric (Figures 7a

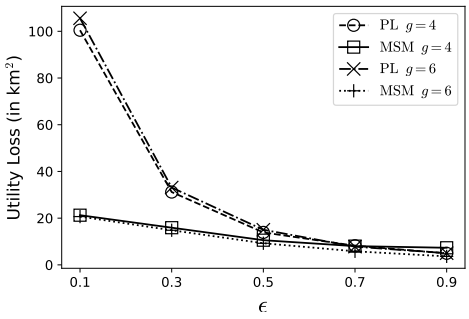


(a) Gowalla dataset

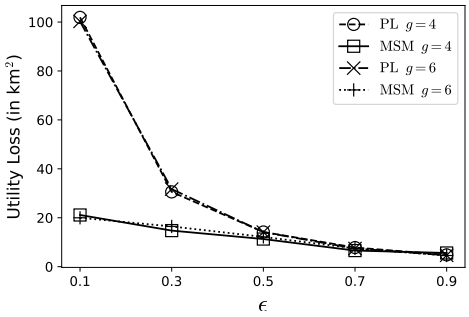


(b) Yelp dataset

Figure 6: Effect of ϵ on utility loss (Euclidean utility metric).

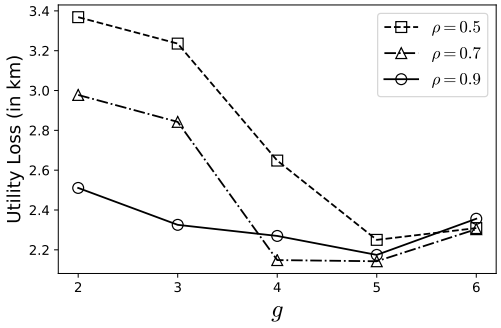


(a) Gowalla dataset

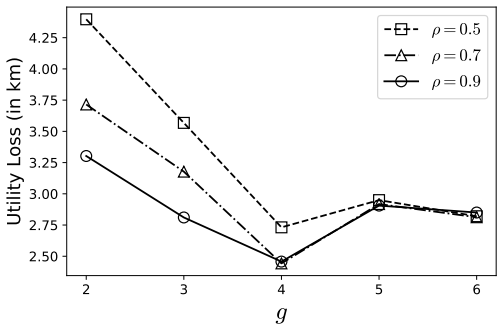


(b) Yelp dataset

Figure 7: Effect of ϵ on utility loss (squared Euclidean utility metric).



(a) Gowalla dataset



(b) Yelp Dataset

Figure 8: Effect of varying granularity on the utility loss (Euclidean utility metric).

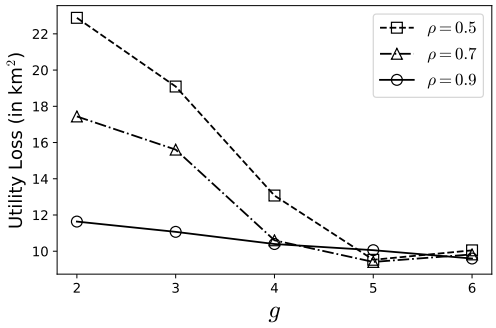
and 7b). This time, the gap between the two approaches is even larger, with MSM outperfoming PL by a factor of 5 at the low end of the privacy budget range. PL does however catch up with our method earlier than in the case of Euclidean utility metric (around the $\varepsilon = 0.5$ threshold). Still, MSM remains clearly superior in the range of tight privacy settings.

Our results show that MSM outperforms significantly the PL benchmark in terms of utility loss. As discussed in Section 2, PL is very fast in terms of execution time, and it takes on average 10 milliseconds to complete. In contrast, our method is more expensive. Still, the execution time is always below 1 second in the worst case (which occurred for the highest considered granularity case $g = 6$ and a large ε value). In most cases, the average runtime of our method is in the range of $100 - 200$ milliseconds. Even though this is higher than PL, $100 - 200$ milliseconds per query is a small price to pay in order to increase utility significantly. Furthermore, recall that this cost will be incurred at the client side, so there is no danger of the server being overloaded due to a spike in request. In terms of user experience, the additional time required for sanitizing locations is barely noticeable.

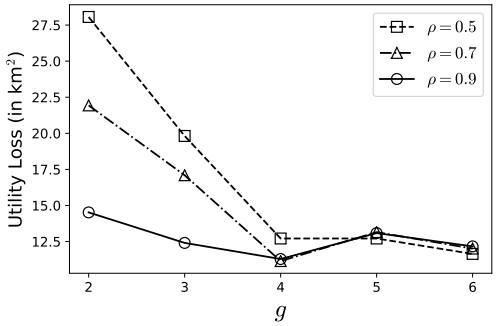
For the rest of the evaluation, we no longer consider PL, and we focus on analyzing the performance of the proposed MSM method when varying its system parameters values.

6.3 MSM System Parameter Analysis

Next, we evaluate the utility of MSM when varying grid granularity, while also considering several settings of ρ (shown as distinct lines in each graph). Recall that, for MSM the fanout at each level is $g \times g$, hence the first level (below the single virtual root node) has granularity $g \times g$, the second level granularity $g^2 \times g^2$, and

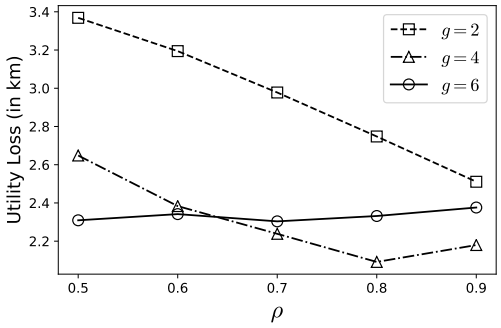


(a) Gowalla dataset

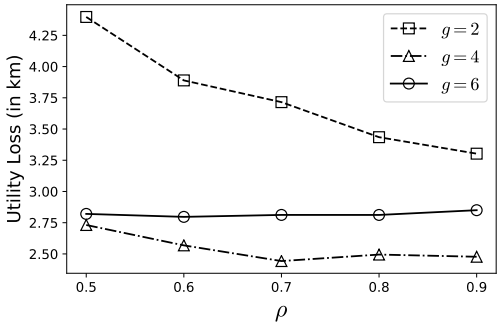


(b) Yelp Dataset

Figure 9: Effect of varying granularity on the utility loss (squared Euclidean utility metric).



(a) Gowalla dataset



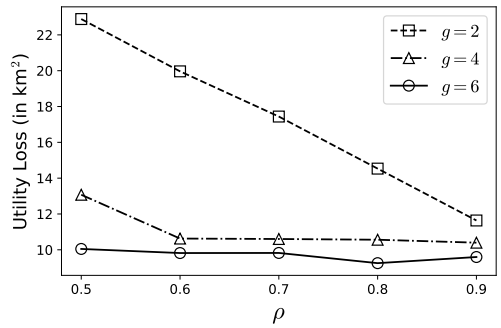
(b) Yelp Dataset

Figure 10: Effect of varying probability ρ on the utility loss (Euclidean utility metric).

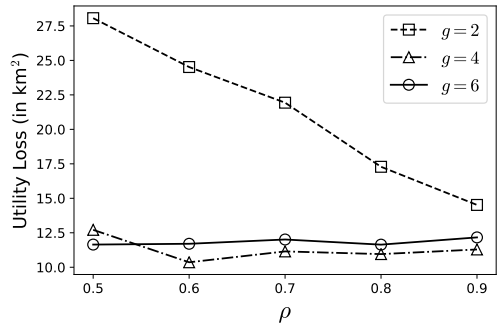
so on. Figures 8a and 8b show the obtained results for the two datasets under Euclidean distance utility metric. The general trend is that of a "U"-shaped dependency: utility loss decreases initially as granularity increases, which intuitively is a result of a higher precision in reporting locations. However, after a certain point, the utility loss starts to increase, as a high granularity will determine more cases where the reported and actual locations are in different cells. As a side effect, as the area of each grid cell decreases, more budget is required at a certain level to maintain the same required probability ρ, which can starve lower index levels of privacy budget. We notice that the ideal granularity may also vary with the dataset, as data density typically affects accuracy. For the Gowalla dataset the best-performing granularity is at $g = 5$, whereas for the Yelp dataset it is $g = 4$. For a given granularity, we note that a higher value of ρ typically results in better utility, although there are some exceptions: for instance, at $g = 4$, the $\rho = 0.7$ case outperforms the $\rho = 0.9$ setting, for the reason explained above, namely the top level uses most of the budget, and the lower levels do not receive sufficient budget to accurately execute the linear program.

A similar trend is observed when using squared Euclidean distance as utility metric, as shown in Figures 9a and 9b. A higher ρ results in better utility in the majority of cases.

Finally, we measure the effect of varying parameter ρ, with several distinct settings of granularity (shown as different lines in each graph). Figures 10a and 10b summarize the results for Euclidean distance, whereas results for the squared Euclidean distance are shown in Figures 11a and 11b. For the lowest granularity $g = 2$, we note a clear decreasing trend in utility loss. As the grid



(a) Gowalla dataset



(b) Yelp Dataset

Figure 11: Effect of varying probability ρ on the utility loss (squared Euclidean utility metric).

granularity grows gradually from one level to another, the algorithm for budget allocation is able to allocate budget in a smoother fashion, leading to steady progress as ρ grows (we emphasize that although the trend is more pronounced, the absolute value of utility is worse for the $g = 2$ setting compared to the rest). For the other settings of g, due to the fact that the transition from one level to the other is more abrupt, the net effect of ρ exhibits a not-so-well defined trend. The $g = 4$ case still shows a decreasing trend initially, but then utility loss starts to increase as ρ grows, as a result of budget starvation at lower index levels. In the case of $g = 6$, the budget starvation seems to manifest even at lower ρ values, so the trend is constant to slightly increasing. Note that, starvation is not necessarily a negative effect, in the sense that the utility can still be better than the other settings. Starving lower levels of budget may be a worthwhile choice, as long as the higher levels keep the reported cell close to the actual cell, and in effect the utility loss is low. We insist on the starvation concept mostly to describe the observed trends. We do, however, note that excessive starvation can sometime increase utility loss: in the case of the Yelp dataset for instance, we observe that the utility loss obtained at a finer granularity ($g = 6$) is higher than that obtained with $g = 4$.

7 RELATED WORK

In the past decade, a vast amount of research focused on preserving location privacy. The earliest approaches used *dummy generation* [18, 27] to protect locations of users who issue location-based queries. In [18], the user issues a number of redundant (fake) queries at random locations, thus decreasing the probability that an adversary guesses which is the real location. The work in [27] chooses an *anchor* around the real location, and focuses on how query processing can be done around the anchor in such a way that precise results are obtained with respect to the real location (e.g., exact nearest-neighbor queries). Both approaches are vulnerable to background-knowledge attacks: an adversary who knows the map features or the patterns of user movement can filter out the fake locations and reveal the real one.

The *spatial k-anonymity (SKA)* concept has been introduced in [16] to address the limitations of dummy generation. The main idea behind SKA is to construct a *cloaking region (CR)* that encloses at least k *real* users. This way, it is more difficult for the adversary to filter out locations and narrow down the query source. The work in [21] showed how SKA can be implemented on top of a spatial index, but may be subject to reverse engineering attacks because the CR generation algorithm is deterministic and takes as seed the location of the querying user. Later on, [17] introduced the *reciprocity* property, which shows that as long as the same CR is generated for all k users in an anonymity set, the adversary's probability of identifying the user issuing a query is bounded above by $1/k$. However, all SKA approaches no longer provide protection when users move, or when some users in the anonymity sets disconnect. In addition, they may reveal the exact user locations: for instance, if k users are situated inside a hospital, it is possible for the CR to be completely enclosed by the hospital area. Even if the attacker cannot identify which user issued the query, s/he learns that all users are in a hospital, which is a serious privacy breach. Some later work considered *spatial diversity* [12, 26], which attempted to enlarge cloaking regions such that association with sensitive features (e.g., hospitals, nightclubs) is reduced. However, the limitations of SKA remain in place for spatial diversity when users move or disconnect.

The introduction of the novel semantic model of *differential privacy (DP)* changed the landscape of location privacy approaches. DP is a statistical model designed to address the scenario of *aggregate queries*, where the presence of an individual in a dataset must be hidden. Therefore, it is not directly applicable in the context of online location reporting, which is our problem setting. However, DP can be used in the context of publishing historical datasets of locations or trajectories. For instance, the work in [11] shows how spatial indexes can be used to release geospatial datasets at different granularities. The authors consider quadtrees and k-d-trees, and propose a cost model to help allocate budget across different structure levels. While we also consider multiple index levels, our context is a completely different one (that of geo-indistinguishability). Furthermore, their model shows that it is best to allocate smaller budgets towards the root of the index, and preserve a larger proportion of the budget for the leaf level, whereas our findings for the GeoInd setting are exactly the opposite. Due to the fact that an error at the root of the structure has more impact on data utility, we found that it is important to allocate a higher proportion of the budget at the higher index structure levels. Some other work in the context of DP [7, 8] focuses on releasing trajectories using noisy counts of prefixes or n-grams in a trajectory, but similar to [11], the results apply to the offline setting only.

Geo-indistinguishability [1, 2, 5] is a novel and promising semantic model that inherits the powerful guarantees of DP, but adapts them to the setting of online location protection. Since its introduction in 2013, several research efforts focused on finding efficient and utility-preserving techniques for implementing GeoInd. Closest to our work is the research in [5], which proposed several mechanisms and post-processing algorithms to improve utility. However, as we show in our extensive performance evaluation, those techniques are either fast and inaccurate (planar Laplace in conjunction with remapping to grid), or they only work for very small sets of candidate locations (as is the case with the optimal mechanism). In contrast, our multiple-step approach is able to achieve high utility with very reasonable computational overhead.

8 CONCLUSION

We proposed a multiple-step algorithm for protecting location privacy according to the geo-indistinguishability model. By using a multi-level index structure, we are able to prune the solution search space of expensive GeoInd mechanisms, and achieve high utility with low performance overhead. We also derived cost models that show how to judiciously allocate the available privacy budget, and how to choose important index parameters to improve accuracy. The resulting approach outperforms significantly existing benchmarks. In future work, we plan to investigate more advanced cost models to better capture prior information, and thus further improve the accuracy of our approach. We will also investigate more complex index structures (e.g., k-d-trees and R^+ trees) which can adjust better to skewed distributions of priors.

Acknowledgements. The authors thank Noam D. Elkies for his guidance with the computation of the infinite series of exponents in Section 5. This work has been supported in part by NSF grants IIS-1320149 and CNS-1461963, the USC Integrated Media Systems Center, and unrestricted cash gifts from Oracle. The opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors such as NSF.

REFERENCES

[1] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, pages 901–914, 2013.

[2] N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Optimal geo-indistinguishable mechanisms for location privacy. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 251–262, 2014.

[3] Y. D. Challenge. Yelp dataset challenge, 2019.

[4] K. Chatzikokolakis, M. E. Andrés, N. E. Bordenabe, and C. Palamidessi. Broadening the scope of differential privacy using metrics. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 82–102. Springer, 2013.

[5] K. Chatzikokolakis, E. ElSalamouny, and C. Palamidessi. Efficient utility improvement for location privacy. In *Proceedings on Privacy Enhancing Technologies (PoPETS)*, pages 308–328, 2017.

[6] K. Chatzikokolakis, C. Palamidessi, and M. Stronati. Constructing elastic distinguishability metrics for location privacy. In *Proceedings on Privacy Enhancing Technologies (PoPETS)*, pages 156–170, 2015.

[7] R. Chen, G. Acs, and C. Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *ACM CCS*, pages 638–649, 2012.

[8] 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. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 213–221, 2012.

[9] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090, 2011.

[10] J. Clausen. Branch and bound algorithms-principles and examples. *Department of Computer Science, University of Copenhagen*, pages 1–30, 1999.

[11] G. Cormode, C. Procopiuc, E. Shen, D. Srivastava, and T. Yu. Differentially private spatial decompositions. In *ICDE*, pages 20–31, 2012.

[12] M. L. Damiani, E. Bertino, and C. Silvestri. The probe framework for the personalized cloaking of private locations. *Trans. Data Privacy*, 3(2):123–148, Aug. 2010.

[13] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. of Theory of Cryptography Conference (TCC)*, pages 265–284, 2006.

[14] S. R. Finch. *Mathematical constants*, volume 93. Cambridge university press, 2003.

[15] 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 International Conference on Management of Data (ACM SIGMOD)*, 2008.

[16] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proc. of USENIX MobiSys*, 2003.

[17] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preserving Location-based Identity Inference in Anonymous Spatial Queries. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(12), 2007.

[18] H. Kido, Y. Yanagisawa, and T. Satoh. An Anonymous Communication Technique using Dummies for Location-based Services. In *IEEE International Conference on Pervasive Services (ICPS)*, 2005.

[19] F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 627–636. ACM, 2009.

[20] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30, 2009.

[21] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In *Proceedings of International Conference on Very Large Data Bases (VLDB)*, 2006.

[22] G. Optimization. Inc., gurobi optimizer reference manual. *URL: http://www. gurobi. com*, 2017.

[23] B. Riemann. about the number of primes under a given size. *Ges. Math. Works and Scientific Nachla ss*, 2:145–155, 1859.

[24] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec. Protecting location privacy: Optimal strategy against localization attacks. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pages 617–627, 2012.

[25] E. W. Weisstein. Dirichlet l-series. *Wolfram Research, Inc.*, 2003.

[26] M. Xue, P. Kalnis, and H. K. Pung. Location diversity: Enhanced privacy protection in location based services. In *Proceedings of the 4th International Symposium on Location and Context Awareness*, pages 70–87, 2009.

[27] M. L. Yiu, C. Jensen, X. Huang, and H. Lu. SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services. In *International Conference on Data Engineering (ICDE)*, pages 366–375, 2008.