FELIP: A local Differentially Private approach to frequency estimation on multidimensional datasets

José S. Costa Filho
Universidade Federal do Ceará
serafim.costa@lsbd.ufc.br

Javam C. Machado
Universidade Federal do Ceará
javam.machado@lsbd.ufc.br

ABSTRACT

Local Differential Privacy (LDP) allows answering queries on users data while maintaining their privacy. Queries are often issued on multidimensional datasets with categorical and numeric dimensions. In this paper, we tackle the problem of answering counting queries over multidimensional datasets with categorical and numeric dimensions under LDP. In the setting without a trusted central agent, the user's private dimensions are firstly perturbed locally to preserve privacy and then sent to an aggregator who will be able to estimate answers to queries. We build our approach on the existing idea of using grids. Mapping users dimensions into grids which are perturbed and sent to the aggregator so it can estimate the real data distributions to answer different queries on the dimensions collected. Finer-grained grids lead to greater error due to noises, while coarser-grained ones result in greater error due to biases. We propose optimizing the construction of grids taking into consideration a number of different factors to obtain better accuracy. Also, we propose to adaptively select the LDP algorithm that based on the grid characteristics will provide the better utility. We conduct experiments on real and synthetic datasets and compare our solution with existing approaches.

1 INTRODUCTION

In the past decade, the evolution of Internet-connected devices (*e.g.*, smart devices, Internet of Things appliances) has accelerated the proliferation of the mobile Internet and spurred a new wave of mobile applications, leading to an unprecedented and ever-increasing amount of data [6]. Organizations are interested in collecting users' data to improve their applications and guide business decisions. However, collecting and analyzing data has incurred serious privacy issues since such data may contain users' sensitive information [22, 49]. Moreover, users' private data are susceptible to attacks and disclosure through advanced data fusion, and analysis techniques [41]. Thus, organizations must provide rigorous privacy guarantees on how users' data is collected and analyzed.

Towards privacy-preserving, differential privacy (DP) [11] has been proposed with strict mathematical proofs capable of providing each user with strong privacy guarantees. DP does not depend on attackers' background knowledge and has been widely adopted in many real-world applications at Google [13], Apple [36], Microsoft [8], and Uber [20]. The centralized setting of DP assumes that there is a trusted data collector who holds users' exact data and adds noise in the analytical process to satisfy DP. However, in many online and crowdsourcing applications, the servers may not be trusted by users who prefer that their data does not leave their devices (*e.g.,* smartphones, wearable devices) unprotected. To address that scenario where central DP is not applicable, Local Differential Privacy (LDP) [24] was proposed as a distributed variant of DP that does not require a trusted third party and provides privacy guarantees for each user locally. In the local setting, a randomized algorithm is used to perturb each user's private data locally so it can then be sent to a data collector. Hence, the data collector will never have access to the true data of each user, and LDP guarantees that the likelihood of any specific output of the randomized algorithm varies little with input, *i.e.,* each user's true data.

Typically, users' data records include multiple dimensions that are often numerical (*e.g.,* age, salary) or categorical (*e.g.,* gender, race) attributes [40]. In this paper, we study the problem of estimating frequencies on multi-dimensional data with categorical and numerical attributes while each user's private data is collected under LDP. For example, consider Table 1 to represent the data of users of a particular application. Our approach enables a service provider to estimate the answer of frequency queries with equality and/or range constraints such as the following query: *SELECT COUNT(*) FROM T WHERE Age BETWEEN 30 AND 60 AND Education IN ('Doctorate', 'Masters') AND Salary <= 80k.*

	Age	Education	Sex	Salary	Capital gain
1	29	Bachelors	Male	60k	2174
2	55	Doctorate	Female	100k	14084
3	48	Masters	Female	80k	5178
4	35	Some-college	Male	50k	1301
5	23	Bachelors	Female	45k	880

Table 1: Example of dataset T with 5 dimensions

There are several challenges regarding solving this problem, including 1) capturing the correlations among different attributes, 2) avoiding the curse of dimensionality, and 3) dealing with attributes with large domains. A solution must be able to deal with all these aspects simultaneously; otherwise, it will have poor utility [45]. The main idea to tackle this problem is to use two-dimensional grids to map, using binning, two-dimensional domains of all combinations of two data attributes. Once all these grids are constructed, one can answer any higher dimensional queries from them. When dealing with range constraints from a query, there is the possibility that a grid's cell is partially included in the answer. One approach is to assume that the data distribution is uniform, which may incur higher errors. Another approach is to combine the two-dimensional grids with other one-dimensional grids [45]. The purpose of one-dimensional grids is to capture finer-grained information on the distribution of each attribute. In this work, we experiment with both approaches and make further enhancements to each strategy. The total number of grids depends on the number of attributes in the dataset. To improve the utility, we choose to divide the population of users into groups, and each group will be responsible for collecting information from one grid. Each user reports a perturbed grid under

LDP to the aggregator. With all reports, the aggregator estimates the frequency of each grid cell. After that, in the post-processing phase, we remove the negative frequency estimations since it is known that all individual frequencies should be non-negative, and summing all of them, the result should be one. Also, in the post-processing phase, since each attribute is involved in various grids, we make the estimations consistent for each attribute across respective grids, which helps to improve utility.

The most challenging task here is determining the best size for each grid. A finer-grained grid leads to more significant errors due to noises since, under LDP, one has to perturb each cell. The more cells, the higher the total noisy error. However, coarser-grained grids will lead to high error due to bias. Choosing the size of the grids can be viewed as a bias-variance tradeoff. To decide on the exact size of each grid, we thoroughly analyze the error involved when answering a query. The decision will be based on which attributes are in the grid (grids can be categorical, numerical, numerical × numerical, categorical × categorical, numerical × categorical), the query selectivity on each dimension, the number of users, privacy budget and a data distribution property. In this work, we also study different frequency oracle (FO) protocols when estimating the frequencies of grids' cells. Once we have determined the exact size of the grid, we dynamically choose what FO will lead to a better utility for that specific grid. By using 2-D grids, we are able to capture correlation among attributes and avoid the curse of dimensionality. By carefully binning for each grid, we avoid the problem of dealing with large domains.

Other strategies have used grids [30, 45]. However, we propose a novel approach when using grids. Specifically, 1) We can support a broader range of applications and analysis by enabling querying with range and point constraints. 2) We leverage different frequency oracle protocols with different utility characteristics on a per-grid strategy since each grid may have different sizes. 3) We allow the aggregator to incorporate the knowledge of query selectivity prior to building the grids. 4) We calculate grid dimensions on a per-grid strategy. Using the same grid size for all grids as in [45] will diminish utility. Also, we avoid splitting the privacy budget as in [30]. Finally, 5) We allow cells to have different sizes avoiding the limitation of having to choose a sub-optimal dimension size in order to make it divisible by the domain (Explained in Section 3.2). Therefore, our approach not only improves utility but attends a wider range of applications and analysis.

Contributions. In summary, this paper makes the following contributions:

- We study the use different frequency oracle protocols when collecting data under LDP using grids. We propose a new framework that adaptively selects the best-performing frequency oracle protocol for each data grid to achieve lower estimation errors.
- We propose a new approach to answer frequency queries with point and range constraints, under LDP, on multi-dimensional datasets. By studying the sources of error and incorporating prior knowledge on queries selectivity, our strategy design enables capturing data characteristics more precisely, resulting in high utility.
- We extensively experiment with our solution under different query scenarios and datasets (synthetic and real ones), comparing our results against existing approaches.

Organization. Section 2 presents the background and essential definitions used throughout the work. In Section 3, we present the related work and discuss how the main existing approaches work and their limitations. Section 4 formalizes the problem tackled in this work. Section 5 introduces the step-by-step of our approach while carefully justifying decisions taken. Experimental results are shown in Section 6. Finally, Section 7 concludes this work.

2 BACKGROUND AND DEFINITIONS

2.1 Local Differential Privacy

We consider the scenario where users do not trust the server and require formal privacy guarantees before sharing their data. In this work, we adopt the local model of differential privacy (LDP) [10], also known as the randomized response model. Under LDP, sensitive information v from each user is encoded by a randomized algorithm Ψ, and its output $\Psi(v)$ is sent to the aggregator, which is responsible for collecting all users' reports. Intuitively, LDP guarantees that, no matter what the value of $\Psi(v)$ is, it is approximately equally as likely to be a result of perturbing v as any other v' differing from v. Therefore, if $\Psi(v)$, instead of v , is collected, the users never reveal their private value v. The user's private v degree of privacy is controlled by the privacy budget ϵ. More formally,

DEFINITION 1 (Local Differential Privacy) An algorithm $\Psi(\cdot)$ satisfies ϵ-local differential privacy (ϵ-LDP), where $\epsilon \geq 0$, if and only if for any pair of inputs (v, v'), and any set R of possible outputs of Ψ, we have

$$Pr[\Psi(v) \in R] \leq e^{\epsilon} Pr[\Psi(v') \in R]$$

2.2 Frequency Oracles

A *frequency oracle (FO)* protocol can be used to estimate the frequency of any value $v \in D$ under LDP, where D is the domain. A FO consists of two algorithms. The first one is Ψ, which users use locally to perturb their private data. The second one is Φ, and the aggregator uses it to estimate the frequencies based on the perturb data received. A FO can be used in many different LDP tasks, and most problems can be modeled as a frequency estimation problem. We present below two FO used in this work.

2.2.1 Generalized Randomized Response (GRR). Randomized Response [43] was introduced for binary responses but it can easily be generalized to larger domains. In the GRR, each user with private value $v \in D$ sends their true value v with probability p. Otherwise, with probability $1 - p$, users sends a randomly chosen value $v' \in D$. Formally, the perturbation algorithm is

$$\forall_{x \in D}\ Pr\left[\Psi_{GRR_{(\epsilon)}}(v) = x\right] = \begin{cases} p = \frac{e^{\epsilon}}{e^{\epsilon}+|D|-1} & \text{if x = v} \\ q = \frac{1-p}{|D|-1} = \frac{1}{e^{\epsilon}+|D|-1} & \text{if x} \neq \text{v} \end{cases}$$

GRR satisfies ϵ-LDP since $\frac{p}{q} = e^{\epsilon}$. From a population of n users, the aggregator receives a vector $\mathbf{x} = \langle x_1, x_2, ...x_n \rangle$ of length $|\mathbf{x}| = n$ where $x_i \in D$ is the reported value of the i-th user. Then, it can estimate the frequency of $v \in D$, which consists of the ratio of users with private value v among all n users. Considering $C(n)$ as the number of times v appears in vector $\mathbf{x}$, the algorithm for estimating the frequency of $v \in D$ is

$$\Phi_{GRR_{(\epsilon)}}(v) := \frac{C(v)/n - q}{p - q} = \frac{\frac{C(v)}{n} - \frac{1}{e^{\epsilon}+|D|-1}}{\frac{e^{\epsilon}-1}{e^{\epsilon}+|D|-1}} \qquad (1)$$

It is shown, in [38], that this is an unbiased estimation of the true count, and the variance for this estimation is

$$Var\left[\Phi_{GRR_{(\epsilon)}}(v)\right] = \frac{e^{\epsilon} + |D| - 2}{n(e^{\epsilon} - 1)^2} \quad (2)$$

Since variance 2 is linear in $|D|$, the accuracy of this protocol decreases when the domain size $|D|$ increases.

2.2.2 Optimized Local Hashing (OLH). The Optimized Local Hashing [38] protocol tackles the problem of dealing with large domains observed in GRR by adopting a hash function to map an input value into a smaller domain of size g. It applies randomized response to the hashed value. In this protocol, it is crucial to notice that both the hashing step and the randomized step result in information loss. There is a tradeoff when choosing the value of g. One must choose between losing information during the randomization step or the hashing step. In [38], to minimize the variance, the value of g should be $\lceil e^{\epsilon} + 1 \rceil$. The hashing process uses a universal hash function family called $\mathbb{H}$ such that each $H \in \mathbb{H}$ takes as input a value in D and outputs a value in $1...g$. The protocol used to report a value using OLH is

$$\Psi_{OLH_{(\epsilon)}}(v) := \left\langle H, \Psi_{GRR_{(\epsilon)}}(H(v)) \right\rangle$$

Let $\left\langle H^i, x^i \right\rangle$ be the report from the i-th user. To compute the the frequency of each $v \in D$, the aggregator first computes $C(v) = |\{j | H^j(v) = x^j\}| = \sum_{j \in n} \mathbb{1}_{\{H^j(v) = x^j\}}$, which is the number of user reports that supports the value v. After that, the aggregator transforms $C(v)$ to its unbiased estimation

$$\Phi_{OLH_{(\epsilon)}}(v) := \frac{C(v) - n/g}{p - 1/g}$$

In [38], the variance found for that estimation is

$$Var\left[\Phi_{OLH_{(\epsilon)}}(v)\right] = \frac{4e^{\epsilon}}{n(e^{\epsilon} - 1)}$$

3 RELATED WORK

Wang et al. [37] proposed LDP mechanisms for collecting multi-dimensional data and estimating the frequency and mean values. In their mechanism, each user submits k (instead of d) attributes with privacy budget ϵ/d. Arcolezi et al. [3] focused on estimating frequencies on longitudinal multi-dimensional data collections. Their strategy randomly samples a single attribute and chooses the best frequency oracle protocol to apply in the two rounds of sanitization. Li et al. [25] propose the Square Wave (SW) approach for reconstructing the distribution of an ordinal attribute. Ren et al. [34] generalize the Expectation Maximization algorithm for estimating the joint distribution of two attributes. Cormode et al. [7] refine and analyze how to release marginals via transformations under LDP. Zhang et al. [48] proposed an approach for marginal release under LDP that adapts the ideas of consistency enforcement and maximum entropy estimation from PriView [31]. Liu et al. [27] proposed an approach that collects data under LDP to generate synthetic datasets that have an approximate joint distribution with the actual dataset. The work introduced an incremental learning-based probabilistic graphical model construction method and a new marginal calculation method for the large cliques in the context of LDP.

Gursoy et al. [16] introduced a Bayesian adversary to analyze the privacy relationships of LDP protocols under varying settings. Linkang et al. [9] dynamically control the build of a tree structure so that the injected noise is well controlled for maintaining high utility when answering range queries. Wang et al. [42] selectively collect a few prefix-sums in a data-dependent way, and answer multidimensional range queries over prefix-sum-based cubes. Alnemari et al. [2] proposed a Multi-Attribute DisAssembly Mechanism for answering multidimensional range queries on healthcare data.

Qardaji et al. [30] proposed an adaptive grid method that lays a coarse-grained grid over the dataset, and then further partitions each cell according to its noisy count. Both levels of partitions are then used in answering queries over the dataset. Our work also uses grids to map users' multidimensional data. However, unlike our work, [30] is developed to work in the central setting of differential privacy, and it works for only two dimensions. Applying the same idea in the local setting is not trivial. Moreover, it proposes to split the privacy budget among cell-splitting phases which could add excessive noise in the local setting.

LDP has also been applied to many different tasks. Estimate frequency and mean on key-value datasets [15, 26, 44, 46], answer linear queries [5, 12, 28], collect and answer queries on geospatial data [14, 19, 47], evolving data [21], continuously answer queries based on stream [33, 39]. However, their problems are different from ours. Thus, their solution is unsuitable for answering multi-dimensional queries in the proposed setting. Below in detail the works that are most similar to ours.

3.1 HIO

HIO [40] is a hierarchy-based approach designed to answer multi-dimensional analytical queries under LDP. It supports frequency queries with categorical and numerical attributes. In HIO, given k attributes with domain d, the aggregator is responsible for constructing a 1-D hierarchy for each attribute. More specifically, a 1-D hierarchy is a hierarchical set of intervals with a branching factor b. Level 0 corresponds to the root, which covers the entire domain d, and it is recursively partitioned into b equally sized subintervals until the leaves (*i.e.,* smaller intervals) only contain one value. There are b^j intervals on level j, each covering d/b^j values. There is a total of $h = \log_b d + 1$ levels, which are called one-dim levels in a 1-D hierarchy. In HIO, a k-dim level is a group of k one-dim levels $(l_1, l_2, ..., l_k)$. Each level corresponds to one d 1-D hierarchies. Also, consider that a k-dim interval is a group of k intervals. Each one of these intervals corresponds to one d 1-D hierarchies. The aggregator is responsible for constructing a k-dimensional hierarchy with the d 1-D hierarchies. A level in the k-dimensional hierarchy is said to be a k-dim level. Therefore, a k-dimensional hierarchy has a total of $(h+1)k$ k-dim levels. Since there are b^j subintervals in a one-dim level l in a 1-D hierarchy, a k-dim level $(l_1, l_2, ..., l_k)$ includes $\prod_{i=1}^{k} b^{j_i}$ k-dim intervals. In the case of a categorical dimension, there are basically two levels; the first one represents all values. In the second level, we have all the individual values in the domain d. All other intermediate levels are unnecessary.

HIO also uses the notion of dividing users instead of dividing the privacy budget ϵ. More specifically, the aggregator divides users into $(h+1)k$ groups, which accounts for one group reporting for one k-dim level. Optimized local hashing protocol is used to get the frequency estimation of all k-dim intervals in all k-dim levels. Next, the aggregator answers a multidimensional query by expanding a query q to a new k-dimensional query q' that is interested in all k attributes by assigning a specified interval $[1, d]$ for each attribute that is not in the query. Next, for each attribute k, the aggregator finds the least number of subintervals that can make up its specified interval in q' from its corresponding 1-D

hierarchy. The last step sums up the noisy frequencies of all the k-intervals to get the answer of q.

Limitations. Since users are divided into $(h+1)k$ groups where $h = \log_b d$. The number of users in each group decreases when the attributes' domain or the number of dimensions increases. That translates to a high amount of noise in the frequencies of k-dim intervals, which results in low utility. Therefore, HIO fails to handle the curse of dimensionality and attributes with large domains.

3.2 TDG & HDG

Two-Dimensional Grid (TDG) [45] is a grid-based approach focused on answering multi-dimensional range queries under LDP. TDG's idea is to use binning to partition the 2-D domains of all attribute pairs into 2-D grids that can answer all 2-D range queries and then estimate the answer of a higher dimensional range query from the answers of corresponding 2-D range queries. In TDG, given k attributes, the aggregator first generates all attribute pairs. Then, the aggregator divides m users into $\binom{k}{2}$ groups. Each group corresponds to a pair of attributes. Next, for each attribute pair $\langle a_i, a_j \rangle$ where $1 \leq i < j \leq k$, the aggregator assigns the same granularity (*i.e.,* grid size of one dimension) g_2 to construct a 2-D grid $G(i, j)$ by partitioning the 2-D domain $d \times d$ into $g_2 \times g_2$ 2-D cells of equal size. It assumes that all attributes have the same domain. In particular, each 2-D cell specifies a 2-D subdomain of $d \times d$. Each user reports their private value on one of these grids, and the frequencies of each cell are estimated using OLH. When answering queries using TDG, the aggregator assumes that values in a cell are uniformly distributed, which may incur additional error.

Hybrid-Dimensional Grid (HDG) introduces finer-grained 1-D grids that, in combination with the information of 2-D grids, provide better utility when compared to TDG. In HDG, the aggregator constructs k 1-D grids for the k attributes, respectively. Thus, there are a total of $k + \binom{k}{2}$ grids in HDG, with users being divided into $m = k + \binom{k}{2}$ groups. As in TDG, one user group reports values on one specific grid. The aggregator calculates the granularity g_1, which is the size of all 1-D grids. Each 1-D cell specifies a 1-D subdomain of d. Finally, the aggregator uses OLH to obtain noisy frequencies of every cell of 1-D and 2-D grids. On the aggregator side, negativity is removed from the estimates. Also, since an attribute is related to multiple grids, inconsistencies among estimates are removed. After that, $\lambda - D$ queries are split into $\binom{\lambda}{2}$ related 2-D range queries and then from all answers of these $\binom{\lambda}{2}$ 2-D queries the answer of the $\lambda - D$ is estimated.

Limitations. We identify three main limitations with TDG and HDG. First, they only support range queries, limiting the analysis one may be interested in performing. Also, it assumes that queries will correspond to half the interval of attributes and that assumption impacts the construction of grids, which limits the utility gains. Additionally, it uses the same size for every grid. Moreover, to ensure that g_1 and g_2 are divisible by domain size d simultaneously, they take the power of two closest to its derived value as the final value. These aspects negatively impact accuracy since the actual size of the grids may be very different from the optimal one. For instance, suppose the optimal granularity is 25 for 1-D grid, the granularity of the actual grid will be 32, which is 20% larger than the optimal value. Also, suppose that the optimal value of a 2-D grid is 11×11, totaling 121 cells; then the actual granularity will be 8×8 totaling 64 cells. Then the total number of cells will be almost half of what it should be.

4 PROBLEM STATEMENT

Consider there are k attributes $a_1, a_2, ..., a_k$, and those attributes can be ordinal and categorical. Let each attribute have a domain $d = d_1, d_2, ..., d_k$. Let n be the total number of users. The i-th user's record is a k-dimensional vector expressed by $v_i = \langle v_i^1, v_i^2, ..., v_i^k \rangle$ where v_i^t means the value of attribute a_t in record v_i. Our solution tackles the problem of answering multi-dimensional queries under LDP. Moreover, a multi-dimensional query is a conjunction of multiple predicates for attributes of interest. A λ-dimensional query q is defined as

$$q = (a_{t_1}, o_{t_1}, v_{t_1}) \wedge (a_{t_2}, o_{t_2}, v_{t_2}) \wedge ... \wedge (a_{t_\lambda}, o_{t_\lambda}, v_{t_\lambda})$$

where $o_{t_i} \in \{in, between\}$ specifies an operator, v_{t_i} is either a set of size $0 < |v_{t_i}| \leq d_{t_1}$ of categorical values or a range $[l_{t_i}, r_{t_i}]$. Also, $1 \leq t_i \leq k$, and $t_i \neq t_j$ when $i \neq j$. Let $A_q = \{a_{t_i} | 1 \leq i \leq \lambda\}$ represent the set of attributes specified in query q. That means that a query q selects all records whose value of attribute a_{t_i} is a specific value when o_{t_i} is the = (Equals) operator, or a_{t_i} is in the set of categorical values when $o_{t_i} = in$ or a_{t_i} is between the range when $o_{t_i} = between$. The answer to query q is equal to the count of all records that satisfy all conditions divided by the number of users n. Formally, the real answer of query q can be expressed as

$$\tilde{f_q} = \frac{|\{v_i | v_i^t \in v_t, \forall a_t \in A_q\}|}{n}$$

Consider the example dataset on Table 1 with $n = 5$ users. User 1, for example, has the following 5-dimensional vector $v_3 = \langle 48, Masters, Female, 80k, 5178 \rangle$. The example query *SELECT COUNT(*) FROM T WHERE Age BETWEEN 30 AND 60 AND Education IN ('Doctorate', 'Masters') AND Salary <= 80k* on Table 1, is interested in attributes $A_q = \{Age, Education, Salary\}$ and it is represented as $q = (a_{t_1} = Age, o_{t_1} = BETWEEN, v_{t_1} = [l_{t_i} = 30, r_{t_i} = 60]) \wedge (a_{t_2} = Education, o_{t_2} = IN, v_{t_2} = \{`Doctorate', `Masters'\}) \wedge (a_{t_3} = Salary, o_{t_3} = BETWEEN, v_{t_3} = [l_{t_i} = 0, r_{t_i} = 80000])$. The answer to q is 1 and $\tilde{f_q} = \frac{1}{5} = 0.2$. Table 2 lists all notations used throughout this work.

Notations	Meaning
n	The total number of users
k	Total number of attributes
k_n	The number of numerical attributes
k_c	The number of categorical attributes
d	The domain size of an attribute
m	The number of user groups
q	A query
A_q	The set of attributes in q
λ	The query dimension
L	Total number of cells in a grid
l_x	Number of cells in the x axis of the grid
l_y	Number of cells in the y axis of the grid

Table 2: Notations

5 THE FELIP APPROACH

This section presents FELIP, designed to answer, under LDP, multi-dimensional queries LDP over datasets with numerical and categorical attributes. FELIP maps users answers to 1-D and 2-D

grids which are perturbed to satisfy LDP. The first step of FELIP consists of the aggregator determining the number of grids to be estimated and dividing the users into groups where each group is responsible for one grid. In the second step, the aggregator calculates the dimensions' size of each grid, and it can leverage the information it has about the queries' selectivity. Note that the grid construction happens on a per-data collection basis. With one collection, the aggregator can answer all queries. So when choosing the selectivity, the aggregator can use a precise number of one specific query it wants to answer, or it can use the average selectivity of a set of queries. After that, the aggregator sends to each user one grid configuration. On the user side, each user projects their answer in the grid, perturbs it under LDP, and sends it to the aggregator. Next, the aggregator collects all answers and estimates the frequencies for each cell of every grid. After that, the post-processing is done. Finally, the aggregator estimates the answers of all high-dimensional queries.

Within FELIP, we develop two strategies called Optimized Hybrid Grid (OHG) and Optimized Uniform Grid (OUG). Since FELIP collects multiple grids, we argue that the utility can be improved by dividing users into groups and show the proofs for that when using OLH and GRR protocols (Section 5.1). Then, we discuss one of the most important aspects of collecting the data, that is, how we can optimize the construction of LDP grids by minimizing the total error while using OLH and GRR protocols (Section 5.2). Next, we motivate why combining GRR and OLH protocols can benefit our problem, presenting the adaptive frequency oracle (Section 5.3). After that, we detail how the aggregator can improve utility by achieving consistency among grids and removing negative estimations in the post-processing stage (Section 5.4). Next, we show the algorithms used for building the response matrix (Section 5.5) and estimating the multi-dimensional query from its related sub-queries (Section 5.6). Finally, we discuss the privacy guarantees of our approach and specify the different sources of error (Section 5.7).

5.1 Population Partitioning

In our approach, we collect, under LDP, information on m different grids, and each grid has L cells. One strategy is to divide the privacy budget into m parts, and the entire population of n users sends reports for all m grids, using the sequential composition [29]. The other strategy is to divide the population into m groups; each group is responsible for reporting information about one specific grid.

Theorem 5.1. *The variance of GRR and OLH is smaller when dividing n users into m groups instead of dividing the privacy budget ϵ into ϵ/m.*

Proof. The variance when answering a query dividing the users with GRR is:

$$Var[\Phi_{GRR_{du}}] = m \cdot \frac{e^{\epsilon} + L - 2}{n(e^{\epsilon} - 1)^2}$$

and the variance when splitting the budget is:

$$Var[\Phi_{GRR_{d\epsilon}}] = \frac{e^{\epsilon/m} + L - 2}{n(e^{\epsilon/m} - 1)^2}$$

Then, if we do $Var[\Phi_{GRR_{d\epsilon}}] - Var[\Phi_{GRR_{du}}]$, we have

$$\begin{aligned}
&= \frac{1}{n}\left[\frac{e^{\epsilon/m} + L - 2}{(e^{\epsilon/m} - 1)^2} - m\frac{e^{\epsilon} + L - 2}{(e^{\epsilon} - 1)^2}\right] \\
&= \frac{L-2}{n}\left[\frac{1}{e^{\epsilon/m} - 1)^2} - \frac{m}{(e^{\epsilon} - 1)^2}\right] + \\
&\quad \frac{e^{\epsilon/m}}{n}\left[\frac{1}{e^{\epsilon/m} - 1)^2} - \frac{me^{\epsilon - \epsilon/m}}{(e^{\epsilon} - 1)^2}\right] \\
&= \frac{(L-2)}{n(e^{\epsilon/m} - 1)^2(e^{\epsilon} - 1)^2}\left[(e^{\epsilon} - 1)^2 - m(e^{\epsilon/m} - 1)^2\right] + \\
&\quad \frac{e^{\epsilon/m}}{n(e^{\epsilon/m} - 1)^2(e^{\epsilon} - 1)^2}\left[(e^{\epsilon} - 1)^2 - me^{\epsilon - \epsilon/m}(e^{\epsilon/m} - 1)^2\right]
\end{aligned}$$

Denoting $e^{\epsilon/m}$ as z and since $\epsilon > 0, z > 1$, the two first terms will always be greater than zero and for the two second terms we have that

$$\begin{aligned}
&(e^{\epsilon} - 1)^2 - m(e^{\epsilon/m} - 1)^2 \\
&= (z^m - 1)^2 - m(z - 1)^2 \\
&= (z - 1)^2\left[(1 + z^2 + \ldots + z^{m-1})^2 - m\right] > 0
\end{aligned}$$

Also, we have that

$$\begin{aligned}
&(e^{\epsilon} - 1)^2 - me^{\epsilon - \epsilon/m}(e^{\epsilon/m} - 1)^2 \\
&= (z^m - 1)^2 - mz^{m-1}(z - 1)^2 \\
&= (z - 1)^2\left[(1 + z^2 + \ldots + z^{m-1})^2 - mz^{m-1}\right] > 0
\end{aligned}$$

Therefore, $Var[\Phi_{GRR_{d\epsilon}}] - Var[\Phi_{GRR_{du}}] > 0$. That is, the variance of the approach that divides the privacy budget $Var[\Phi_{GRR_{d\epsilon}}]$ is always greater than the variance of the approach that divides users $Var[\Phi_{GRR_{du}}]$. The proof for the OLH protocol is similar to GRR, and we have that $Var[\Phi_{OLH_{d\epsilon}}] - Var[\Phi_{OLH_{du}}] > 0$. That is, the approach of dividing the privacy budget $Var[\Phi_{OLH_{d\epsilon}}]$ is always greater than the approach of dividing users $Var[\Phi_{OLH_{du}}]$. Then, we can conclude that for either protocol, OLH and GRR, dividing users will achieve better utility than dividing the budget. □

5.2 Constructing Grids with OLH and GRR

This section discusses how grids are constructed for OUG and OHG while using the OLH and GRR protocols. We design our strategies to handle numerical and categorical attributes with different domain sizes. Also, we incorporate knowledge of the queries' selectivity. By considering these aspects, we can calculate more accurate grid sizes, which translates to a higher utility by balancing the bias-variance tradeoff well. The construction of grids varies in complexity depending on the grid type. It can be O(1) for 1-D grids, or it can involve solving an equation system numerically, which can be done with several different algorithms. In this work, we use the bisection method in all scenarios, and it has converged quickly in all of them.

In OUG, given k attributes, a total of $\binom{k}{2}$ pairs of attributes are generated, representing all the possible attribute combinations. The n users' population is randomly divided into $\binom{k}{2}$ groups. The size of a grid is represented by l_x and l_y, which refers to the size of the grid along the x and y axis, respectively. For each pair of attributes $\langle a_i, a_j \rangle$ where $1 \leq i < j \leq k$, a 2-D grid $G(i, j)$ is constructed partitioning the 2-D domain $[d_i]$ x $[d_j]$ into $l_x \times l_y$ 2-D cells.

In the OHG strategy, 2-D and 1-D grids are constructed. 1-D grids are constructed for numerical attributes only. Later, we utilize this information to refine the final answer grid (Section 5.5).

As in OUG, 2-D grids are constructed for each pair of attributes. From a total of k attributes, k_n represents the number of numerical attributes. The population of n users is divided into $k_n + \binom{k}{2}$ groups, and each group is responsible for one grid. 2-D grids are constructed the same way as in OUG. For each numerical attribute $a_i (1 \leq i \leq k)$, 1-D grid $G(i)$ is constructed with size l_x. Finally, users are asked to report which cell of their assigned 2-D or 1-D grid their private value is in using the AFO protocol.

While constructing grids, we have to consider the effect of non-uniformity error in the numerical dimensions. The non-uniformity error accounts for the difference caused by cells that intersect the query rectangle but are not contained in it. Since we do not have access to the actual data distribution, we need to estimate how many data points are in the intersected cells. A common approach assumes that the data points are uniformly distributed in each cell [30, 45]. The magnitude of this error in any intersected cell, in general, depends on the number of data points in that cell, and is bounded by it. Thus, the finer the grid's granularity, the lower the non-uniformity error. When computing the size of the grids, we approximate the non-uniformity error and control the degree of this error by using constants α_1 and α_2 for 1-D and 2-D grids, respectively.

Calculating the size of grids for OLH and GRR.

Numerical 1-D Grids. We consider that the ratio of interval to the attribute's domain size *i.e.,* query selectivity is r_x and the number of cells in the grid is l_x. The squared noise and sampling error for OLH is

$$(l_x \cdot r_x) \cdot \frac{4me^\epsilon}{n(e^\epsilon - 1)^2} = \frac{4l_x r_x m e^\epsilon}{n(e^\epsilon - 1)^2}$$

For GRR is

$$(l_x \cdot r_x) \cdot \frac{m(e^\epsilon + l_x - 2)}{n(e^\epsilon - 1)^2} = \frac{l_x r_x m(e^\epsilon + l_x - 2)}{n(e^\epsilon - 1)^2}$$

The non-uniformity error is $\left(\frac{\alpha_1}{l_x}\right)$. The sum of the two errors for OLH is

$$\left(\frac{\alpha_1}{l_x}\right)^2 + \frac{4l_x r_x m e^\epsilon}{n(e^\epsilon - 1)^2} \tag{3}$$

and for GRR is

$$\left(\frac{\alpha_1}{l_x}\right)^2 + \frac{l_x r_x m(e^\epsilon + l_x - 2)}{n(e^\epsilon - 1)^2} \tag{4}$$

We can minimized the sum by taking the partial derivative with respect to l_x. For OLH, we have

$$\frac{\partial\left[\left(\frac{\alpha_1}{l_x}\right)^2 + \frac{4l_x r_x m e^\epsilon}{n(e^\epsilon - 1)^2}\right]}{\partial l_x} = \frac{4e^\epsilon m r_x}{n(e^\epsilon - 1)^2} - \frac{2\alpha_1^2}{l_x^3} = 0$$

$$l_{xOLH} = \sqrt[3]{\frac{n\alpha_1^2 \cdot (e^\epsilon - 1)^2}{2mr_x e^\epsilon}} \tag{5}$$

and for GRR, we take the partial derivative of Equation 4 with respect to l_x, we have

$$\frac{\partial\left[(4)\right]}{\partial l_x} = -\frac{2\alpha_1}{l_x^3} + \frac{l_x r_x m}{n(e^\epsilon - 1)^2} + \frac{(l_x + e^\epsilon - 2)ms}{n(e^\epsilon - 1)^2} \tag{6}$$

Then, we find the roots of the Equation 6 and determine l_{xGRR}. Once we have l_{xOLH} and l_{xGRR}, we determine the size of the grid (*i.e.* number of cells).

Categorical 1-D Grids. The size of the grid is equal to the domain of the attribute $l_{xOLH} = l_{xGRR} = d$.

Numerical × Numerical 2-D Grids. We consider that the ratio of interval to the attribute's domain size *i.e.,* query selectivity is r_x and r_y along the x and y axis respectively. The number of cells included in the query rectangle is $l_x r_x \cdot l_y r_y$. Thus, the squared noise and sampling error for OLH is

$$l_x r_x l_y r_y \cdot \frac{4me^\epsilon}{n(e^\epsilon - 1)^2} = \frac{4l_x r_x l_y r_y m e^\epsilon}{n(e^\epsilon - 1)^2} \tag{7}$$

and for GRR is

$$(l_x r_x \cdot l_y r_y) \cdot \frac{m(e^\epsilon + l_x l_y - 2)}{n(e^\epsilon - 1)^2} = \frac{l_x r_x l_y r_y m(e^\epsilon + l_x l_y - 2)}{n(e^\epsilon - 1)^2} \tag{8}$$

The number of cells included in the four edges of the query rectangle is $2(l_x r_x + l_y r_y)$. The expected sum of frequencies of values included in these cells is $2(l_x r_x + l_y r_y) \cdot \frac{1}{l_x \times l_y} = \frac{2(l_x r_x + l_y r_y)}{l_x l_y}$. The non-uniformity error is $\left(\frac{2\alpha_2(l_x r_x + l_y r_y)}{l_x l_y}\right)^2$. The sum of the two errors for OLH is

$$\left(\frac{2\alpha_2(l_x r_x + l_y r_y)}{l_x l_y}\right)^2 + \frac{4l_x r_x l_y r_y m e^\epsilon}{n(e^\epsilon - 1)^2} \tag{9}$$

and for GRR is

$$\left(\frac{2\alpha_2(l_x r_x + l_y r_y)}{l_x l_y}\right)^2 + \frac{l_x r_x l_y r_y m(e^\epsilon + l_x l_y - 2)}{n(e^\epsilon - 1)^2} \tag{10}$$

Taking the partial derivative of Equations 9 and 10 with respect to l_x and l_y, we get a system of polynomial equations for each one. Then, we solve them to find the values l_x and l_y for each protocol. The grid's size for GRR and OLH is $l_{xGRR} \times l_{yGRR}$ and $l_{xOLH} \times l_{yOLH}$ for GRR and OLH respectively.

Categorical x Numerical grid 2-D Grids. For 2-D grids with one categorical attribute, the dimension with the categorical values l_y will have the size of the attribute's domain. Then, the task is to calculate the length of the other dimension l_x. We consider that the ratio of the interval to the numerical attribute's domain size and the categorical attribute's domain size is r_x and r_y respectively. Assuming that there are $l_x r_x \cdot l_y r_y$ cells in the query. The squared noise and sampling error for OLH is Equation 7 and for GRR is Equation 8. However, in this type of grid, the number of cells in the border of the query rectangle is $2l_y r_y$. The expected sum of frequencies of values included in these cells is $2l_y r_y \cdot \frac{1}{l_x \times l_y} = \frac{2r_y}{l_x}$. Then the squared error from non-uniformity is $\left(\frac{2\alpha_2 r_y}{l_x}\right)^2$. The sum of the two errors for OLH is

$$\left(\frac{2\alpha_2 r_y}{l_x}\right)^2 + \frac{4l_x r_x l_y r_y m e^\epsilon}{n(e^\epsilon - 1)^2} \tag{11}$$

and for GRR is

$$\left(\frac{2\alpha_2 r_y}{l_x}\right)^2 + \frac{l_x r_x l_y r_y m(e^\epsilon + l_x l_y - 2)}{n(e^\epsilon - 1)^2} \tag{12}$$

Taking the partial derivative with respect to l_x, we find the value of l_{xOLH} and l_{xGRR}.

Categorical × Categorical 2-D Grids. The size of grids with 2 categorical attributes a_i, a_j corresponds to the size of the product of the attributes' domains $l_{xGRR} = l_{xOLH} = d_i$ and $l_{yGRR} = l_{yOLH} = d_j$.

5.3 Adaptive Frequency Oracle

After we calculate each grid size using the GRR strategy and OLH strategy (Section 5.2), we need to decide which protocol to choose. We propose selecting the protocol with the lowest variance for reporting each specific grid. In the AFO, we make use of two frequency oracle protocols: GRR and OLH. the variance of AFO is the following:

$$Var[\Phi_{AFO_{(\epsilon)}}] = min\Big(Var[\Phi_{GRR_{(\epsilon)}}], Var[\Phi_{OLH_{(\epsilon)}}]\Big)$$
$$Var[\Phi_{AFO_{(\epsilon)}}] = min\left(\frac{e^{\epsilon}+L-2}{(e^{\epsilon}-1)^2}, \frac{4e^{\epsilon}}{(e^{\epsilon}-1)^2}\right)\cdot\frac{m}{n} \quad (13)$$

5.4 Post-Processing

Answering queries using the information in the grids obtained from the adaptive frequency oracle is an estimation problem. The utility can be improved in the post-processing phase by removing negative estimations and making grids that have attributes in common consistent.

Removing Negative Estimations. When using the AFO, it is common to have negative estimations for many values in the domain. Moreover, the sum of all frequencies may be different from one. To minimize the error in estimations in the post-processing step, one can make every negative estimation have a value of zero and make all frequencies sum up to 1. Algorithm 1 shows how we can remove all negative estimations values and make the sum of all positive frequencies 1. The algorithm's input is the grid's estimations in the form of an estimation vector $\tilde{f}$. First, we make every negative estimation have a value of 0 (line 5). Then, we sum all remaining estimates and take the average difference by dividing by the number of positive estimates (line 6 and 7). Next, we add the difference value to every positive estimate (line 10). We repeat the whole process until all values in the output estimation vector $\tilde{f}_v'$ are non-negative, and all the frequencies sum up to 1. We call this algorithm for each estimated grid.

Algorithm 1: Algorithm for removing negative estimations

Input : Estimation vector $\tilde{f}_v$
Output: Estimation vector $\tilde{f}_v'$

1 $\tilde{f}_v' \leftarrow \tilde{f}_v$;
2 **while** *any negative value in* $\tilde{f}_v'$ *or* $\sum \tilde{f}_v' > n$ **do**
3 **for** $i \leftarrow 0$ **to** $|\tilde{f}_v|$ **do**
4 **if** $\tilde{f}_v'[i] < 0$ **then**
5 $\tilde{f}_v'[\mathrm{i}] \leftarrow 0$;
6 sum $\leftarrow \sum_{j=0}^{j=|\tilde{f}_v|} \tilde{f}_v'(j)$;
7 diff $\leftarrow \frac{n-sum}{|\tilde{f}_v|}$;
8 **for** $i \leftarrow 0$ **to** $|\tilde{f}_v|$ **do**
9 **if** $\tilde{f}_v'[i] > 0$ **then**
10 $\tilde{f}_v'[\mathrm{i}] \leftarrow \tilde{f}_v'[\mathrm{i}]$ + diff;
11 **return** $\tilde{f}_v'$;

Consistency Step. When different grids have some attributes in common, those attributes are estimated multiple times. The utility will increase if these estimates are utilized together [45, 48]. We apply consistency techniques similar to [18]. The consistency of estimates among grids can be achieved with Algorithm 2. Each attribute a is related to w grids in total, which includes one 1-D grid and $w-1$ 2-D grids. Let $G_{(a,w)}$ be the w-th grid that is related to attribute a and assume each grid has a set of cells $L_{G_{(a,w)}}$. We define $S_{G_{(a,w)}}(i)$ to be the sum of frequencies of $G_{(a,w)}$'s cells that are in a subdomain value $D_{G(a,w)}(i)$ of attribute a. For example, suppose an attribute a with domain $d = 50$ is related to a 1-D grid $G_{(a,0)}$, which has 5 cells ($|L_{G_{(a,0)}}| = 5$). $L_{G_{(a,0)}}(0)$ represents the first cell of this 1-D grid, and it defines a subdomain value $[0, 9]$ (range) or 0 (categorical value). The goal is to make all $S_{G_{(a,w)}}(j)$ consistent. we compute their weighted average as $S_{(a,i)} = \sum_{j=1}^{w} \theta_j \cdot S_{G_{(a,w)}}(i)$, where θ_j is the weight of $S_{G_{(a,w)}}(i)$. The values of the weights θ impact the variance of $S_{(a,i)}$. To minimize the variance of $S_{(a,i)}$

$$Var[S_{(a,i)}] = \sum_{j=1}^{w} \theta_j^2 \cdot Var[S_{G_{(a,w)}}(i)] = \sum_{j=1}^{w} \theta_j^2 \cdot |L_{G_{(a,w)}}(j)| \cdot Var_0$$

where Var_0 is the basic variance for estimating a single cell. $|L_{G_{(a,w)}}(j)|$ for 2D grids equals the length of grids $G_{(a,w)}$ in the axis of attribute a and for 1D grids equals the length of the corresponding 1D grid of a divided by the length of grids $G_{(a,w)}$ in the axis of attribute a. Based on the analysis in [48], we calculate in Line 4 each grid's weight $\theta_j = \frac{\frac{1}{|L_{G_{(a,w)}}(j)|}}{\sum_{i=j}^{w} \frac{1}{|L_{G_{(a,w)}}(j)|}}$.

Next, for each subdomain value in the grid $G_{(a,w)}$ (Line 5), we calculate (Line 8) the optimal weighted average

$$S_{(a,i)} = \frac{\sum_{i=j}^{w} \frac{1}{|L_{G_{(a,w)}}(j)|} \cdot S_{G_{(a,w)}}(i)}{\sum_{j=1}^{w} \frac{1}{|L_{G_{(a,w)}}(j)|}}$$

The next step (Line 11) is to make every sum of frequencies $S_{G_{(a,w)}}(i)$ have value $S_{(a,i)}$. To achieve that, we need to update its frequency as

$$S_{G_{(a,w)}}(i) = S_{G_{(a,w)}}(i) + \big(S_{(a,i)} - S_{G_{(a,w)}}(i)\big)/|L_{G_{(a,w)}}(i)|$$

It is essential to mention that a later consistency step does not invalidate the consistency resulting from the previous steps [32]. Since a consistency step may introduce negative estimations, Algorithm 1 has to be called after the consistency algorithm finishes. Removing negative estimations as the last step in post-processing.

Note that applying the consistency step may incur negativity and vice versa. Thus in the post-process, we interchangeably invoke these two steps multiple times. Since we need to ensure non-negativity for the response matrix generation (Section 5.5), we end the post-process with the non-negativity step. While the last step may again introduce inconsistency, it tends to be very small.

5.5 Response Matrix

The response matrix generation phase uses 1-D and 2-D grids to build a matrix for each pair of attributes. The resulting matrices will be used during the phase of answering queries. This idea was also explored in [45]. However, in our case, the selection of the set of related grids Γ is different since we have grids with categorical dimensions. Each pair of attributes (a_i, a_j) has a corresponding response matrix $M(i, j)$ of size $d_i \times d_j$, where the element $M_{(i,j)}[x, y]$ represents the estimated frequency of 2-D value (x, y) in the $[d_i] \times [d_j]$ 2-D domain of the attribute pair (a_i, a_j). If a_i and a_j are categorical attributes, the estimated grid $G(i, j)$ is already the response matrix $M(i, j)$. Otherwise, we need to build $M_{(i,j)}$ by invoking an efficient estimation method called

Algorithm 2: Consistency Algorithm

Initialize vector θ with w zeros;
for $a \leftarrow 0$ **to** $|A|$ **do**
  for $j \leftarrow 0$ **to** w **do**
    $\theta_j = \frac{\frac{1}{|L_{G_{(a,w)}}(j)|}}{\sum_{j=0}^{w} \frac{1}{|L_{G_{(a,w)}}(j)|}}$
  for $i \leftarrow 0$ **to** $|D_{G_{(a,w)}}|$ **do**
    $S_{(a,i)} \leftarrow 0$;
    for $j \leftarrow 0$ **to** w **do**
      $S_{(a,i)} \leftarrow S_{(a,i)} + \theta_j \cdot S_{G_{(a,w)}}(i)$
    for $j \leftarrow 0$ **to** w **do**
      for *each cell* $c \in L_{G_{(a,w)}}(j)$ **do**
        $f_c \leftarrow f_c + S_{(a,i)}$

Weighted Update [4, 17]. If a_i and a_j are numerical, we build $M(i,j)$ from its related grids $\Gamma = \{G(i), G(j), G(i,j)\}$ which represent the grids of $\{a_i, a_j, (a_i, a_j)\}$, respectively. Note that, If one of the attributes is categorical, say a_i, we build $M(i,j)$ using only the grids $\{G(j), G(i,j)\}$, which corresponds to $\Gamma = \{a_j, (a_i, a_j)\}$, respectively. Algorithm 3 shows the step-by-step of building a response matrix. First, select the set, denoted by $\delta(c)$, of all 2-D (x,y) values that contribute to the frequency f_c of cell c from $G \in \Gamma$. For example, suppose that the pair of numerical attributes (a_i, a_j) have domains $d_i = 100, d_j = 50$. Its corresponding grid $G(i,j)$ has size 10×5 cells. Each cell c defines a partition $[l_i, r_i]$ of d_i and $[l_j, r_j]$ of d_j. The first cell, in the position $[0,0]$, from $G(i,j)$, for instance, may define the subdomain $[0,9] \times [0,9]$. Then, in line 4, we would collect all 2-D values in the subdomain $[0,9] \times [0,9]$ *i.e.,* values that contribute to the frequency of c. Once we have all the values $\delta(c)$, the elements in the response matrix are updated (Lines 8-10). Algorithm 3 is set to converge when the sum of the changes of all elements in $M_{(i,j)}$ after each update process is smaller than a certain threshold. It is shown in [45] that the threshold should be smaller than $\frac{1}{n}$.

Algorithm 3: Building Response Matrix

Input : Set of related grids Γ, domain sizes d_i, d_j
Output: Response matrix $M_{(i,j)}$

1 Initialize all $d_i \times d_j$ elements in matrix $M_{(i,j)}$ as $\frac{1}{d_i \cdot d_j}$
while *not convergence* **do**
2 **for** *each grid* $G \in \Gamma$ **do**
3 **for** *each cell* $c \in G$ **do**
4 Find the set of 2-D values $\delta(c)$ from c;
5 $S = 0$;
6 **for** *each 2-D value* $(x,y) \in \delta(c)$ **do**
7 $S = S + M_{(i,j)}[x,y]$;
8 **if** $S \neq 0$ **then**
9 **for** *each 2-D value* $(x,y) \in \delta(c)$ **do**
10 $M_{(i,j)}[x,y] \leftarrow \frac{M_{(i,j)}[x,y]}{S} \cdot f_c$;
11 **return** $M_{(i,j)}$;

5.6 λ-D Query Estimation

To estimate the answer of a high dimensional query we leverage the idea of answering low dimensional 2-D queries and use those values to estimate the answer of λ-D query. This idea has been successfully employed in different works [9, 32, 45, 48]. To estimate the answer f_q of a $\lambda - D$ of a query

$$q = (a_{t_1}, o_{t_1}, v_{t_1}) \wedge (a_{t_2}, o_{t_2}, v_{t_2}) \wedge \ldots \wedge (a_{t_\lambda}, o_{t_\lambda}, v_{t_\lambda})$$

where $o_{t_i} \in \{=, in, between\}$ specifies an operator, v_{t_i} is either a set of size $0 < |v_{t_i}| \leq d_{t_1}$ of categorical values or a range $[l_{t_i}, r_{t_i}]$. $A_q = \{a_{t_\psi} | 1 \leq \psi \leq \lambda\}$. The first step is splitting q into $\binom{\lambda}{2}$ associated 2-D queries

$$\left\{ q_{(i,j)} = (a_i, o_i, v_i) \wedge (a_j, o_j, v_j) | a_i, a_j \in A_q \right\}$$

and then get their answers $\left\{ f_q^{i,j} | a_i, a_j \in A_q \right\}$. Next, one should use the $\binom{\lambda}{2}$ corresponding 2-D queries' answers to estimate f_q. Algorithm 4 shows in detail how estimating the answer of a λ-D query q works. The input of the algorithm is the answers of $\binom{\lambda}{2}$ associated 2-D queries. As output, it returns a estimated answer vector z. In particular, the vector z consists of 2^λ elements that are in one-to-one correspondence with the answers of $2^\lambda \lambda$-D queries in $Q(q) = \wedge_t (a_t, o_t, v_t) \vee (a_t, o_t, v_t') | a_t \in A_q$, where v_t' is the complement of v_t on the domain of a_t. In Algorithm 4, for each $f_q^{(i,j)} \in \left\{ f_q^{(i,j)} | a_i, a_j \in A_q \right\}$, the aggregator performs the following update process on z. The aggregator first finds the set of $\lambda - D$ queries $Q(q)^{(i,j)}$ corresponding to the 2-D query $q^{(i,j)}$, which means that $Q(q)^{(i,j)}$ consists of all those $\lambda - D$ queries whose answers can contribute to $f_q^{(i,j)}$. In particular, $Q(q)^{(i,j)}$ contains $2^{\lambda-2}$ λ-D queries from $Q(q)^{(i,j)}$ and is defined as $\left\{ \wedge_t (a_t, o_t, v_t) \vee (a_t, o_t, v_t') | a_t \in A_q / \{a_i, a_j\} \right\}$. Then, the aggregator calculates the sum Y of $z[q']$ for all $q' \in Q(q)^{(i,j)}$, where $z[q']$ is the element corresponding to the answer of q'. Next, the aggregator uses $f_q^{(i,j)}$ to update the elements in z as Lines 6-8. This process is repeated until convergence. The estimated answer f_q of the λ-D query q equals its corresponding element in z, *i.e.,* $z[q]$. Algorithm 4 is set to converge when the sum of the changes of all elements in $z[q]$ after each update process is smaller than a certain threshold. It is shown in [45] that the threshold should be smaller than $\frac{1}{n}$.

Algorithm 4: Estimating Answer of $\lambda - D$ Query

Input : Associated 2-D queries' answers $\left\{ f_{q^{(i,j)}} | a_i, a_j \in A_q \right\}$
Output: Estimated answer vector **z**

1 Initialize all 2^λ elements in the vector z as $\frac{1}{2^\lambda}$;
2 **while** *not convergence* **do**
3 **for** *each* $f_{q^{(i,j)}} \in \left\{ f_{q^{(i,j)}} | a_i, a_j \in A_q \right\}$ **do**
4 Find the set of queries $Q(q)^{(i,j)}$ corresponding to $q^{(i,j)}$;
5 Calculate $Y = \sum_{q' \in Q(q)^{(i,j)}} z[q']$;
6 **if** $Y \neq 0$ **then**
7 **for** *each query* $q' \in Q(q)^{(i,j)}$ **do**
8 $z[q'] \leftarrow \frac{z[q']}{Y} \cdot f_{q^{(i,j)}}$;
9 **return** z;

5.7 Privacy and Error Analysis

Privacy Assurance. The Adaptive strategy satisfies ϵ-LDP because all the information collected from each user goes through either OLH or GRR with privacy budget ϵ.

The error analysis has to consider all sources of error. They are the following: noise error, sampling error, non-uniformity error, and estimation error.

Noise error. Due to the use of LDP frequency oracles, we have the noise error since, to satisfy ϵ-LDP, an error is added to each cell. These noises are independently generated noise and have the same standard deviation. When summing the noisy frequency of cells to answer a query, the noise error is the sum of the corresponding noises. To generate these independent noises, zero-mean random variables are used; thus, the noises cancel each other out to a certain degree. We know that, for independent random variables X and Y, the variance of their sum is the sum of their variances. Hence, the more partitioned the domain, the more cells are included in a query, and the larger the noise error is.

Sampling Error. In our strategy, we select a fraction n/m of the total number of users to answer the frequency of each cell. As the fraction of users may have a different distribution from the global (*i.e.,* the group of all users), this error must be considered. Let $\bar{f}_v$ represent the true frequency, and f_v is the estimated one. D_n is a fraction of dataset D and m is the number of groups that the total n users are divided. The expected squared error for estimating one value is

$$\mathbb{E}\left[\left(f_v(D_n) - \bar{f}_v\right)^2\right] = \mathbb{E}\left[\left((f_v(D_n) - \bar{f}_v(D_n)) + (\bar{f}_v(D_n) - \bar{f}_v)\right)^2\right]$$
$$= \mathbb{E}\left[\left(f_v(D_n) - \bar{f}_v(D_n)\right)^2\right] + \mathbb{E}\left[\left(\bar{f}_v(D_n) - \bar{f}_v\right)^2\right]$$
$$+ 2\mathbb{E}\left[\left(f_v(D_n) - \bar{f}_v(D_n)\right) \cdot \left(\bar{f}_v(D_n) - \bar{f}_v\right)\right] \quad (14)$$

The expected squared noise and sampling error is basically the first part of Equation 14 since the third part of the equation equals zero and the second part is a constant whose value is much smaller when compared to the first part. In OLH, $p = 1/2$ and $p' = 1/(e^\epsilon + 1)$. Thus, the noise and sampling error is expressed as

$$\mathbb{E}\left[\left(f_v(D_n) - \bar{f}_v(D_n)\right)^2\right] = \frac{4me^\epsilon}{n(e^\epsilon - 1)^2} + \frac{m}{n} \cdot \bar{f}_v$$

and GRR with $p = e^\epsilon/e^\epsilon + d - 1$ and $p' = 1/(e^\epsilon + d - 1)$, we have

$$\mathbb{E}\left[\left(f_v(D_n) - \bar{f}_v(D_n)\right)^2\right] = \frac{m(e^\epsilon + L - 2)}{n(e^\epsilon - 1)^2} + \frac{m}{n} \cdot \bar{f}_v$$

Non-Uniformity Error. This error occurs when we have numerical attributes in the grid. In numerical dimensions, we may have the query rectangle intersecting cells. Since we do not have access to the actual data distribution inside the cell, we have to compute the approximate non-uniformity error. More details are in Section 5.2).

Estimation Error. When estimating the answer of a λ-dimensional query where $\lambda > 2$ from the associated answers of 2-D range queries, estimation error will occur. Since the estimation error is dataset dependent, there is an exact way of estimating it [45].

5.8 Discussion

FELIP and TDG/HDG [45] propose the use of grids to answer multi-dimensional queries. However, there are several differences between the two, and they are summarized as follows:

- Unlike TDG/HDG, the focus of FELIP is to answer queries with point and range constraints.
- TDG/HDG is designed around OLH protocol. FELIP analyzes GRR and OLH. Moreover, it proposes a framework to adaptively choose the protocol that offers the best utility on a per-grid strategy.
- Unlike TDG/HDG, FELIP considers attributes with different domain sizes. Moreover, in TDG/HDG grids' dimensions are not optimal since they have to be divisible by domain size (detailed explanation in Section 3.2). We overcome that limitation by enabling cells to have different sizes within a grid.
- In TDG/HDG, all 1-D grids share the same size g_1. Similarly, 2-D grids all have the same size g_2. FELIP calculates the size of each grid individually while considering each grid's unique aspects.
- In TDG/HDG, the grids assume that the query selectivity is always 50%. In FELIP, the aggregator can utilize the knowledge it has about queries selectivity during the construction of grids.

6 EXPERIMENTAL EVALUATION

In this section, we aim to experiment our approach on different settings and datasets while comparing its results to competitors.

6.1 Setup

We implemented our solution in Python 3.10. HIO and TDG/HDG implementations are available online [1]. All the experiments are conducted on a server with Ubuntu 20.04, Intel Core i7-7820X 3.60GHz, and 128GB memory.

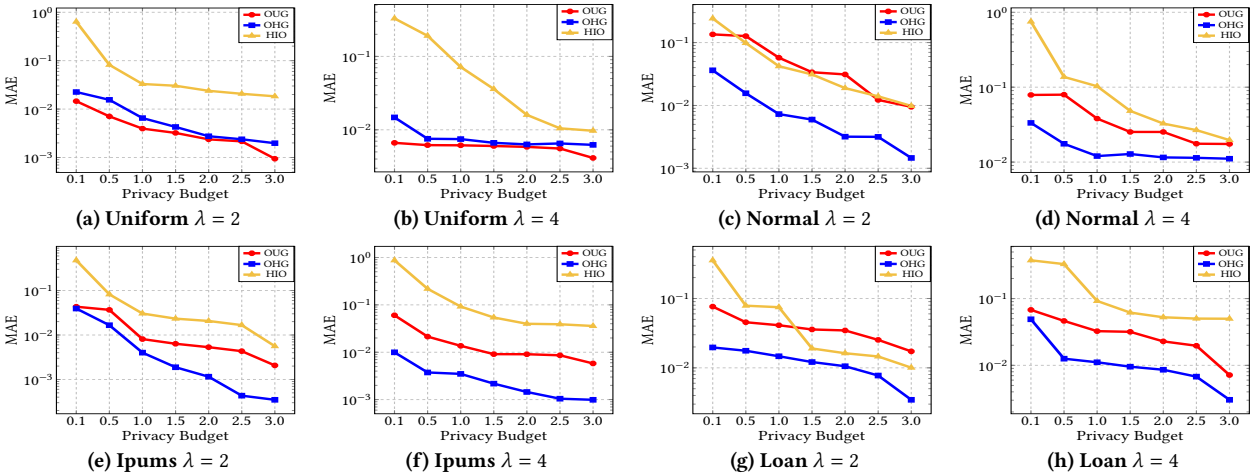
Datasets. We experiment each scenario with four different datasets, two synthetic and two real-world data from different kinds of applications.

- Uniform: Synthetic dataset with all attribute values sampled uniformly. The values in the attributes' domain have roughly the same frequency.
- Normal: Synthetic dataset with attribute values taken from the normal distribution. The distribution is set to cover all the domains of each attribute. The mean is the middle value of the attribute's domain. In this dataset, we experiment with how the strategies behave with more skewed data distribution.
- Ipums [35]: Dataset consists of US census microdata samples from 2014 to 2018. It has information about individuals, such as age, income, and education. We sample 10 million user records with ten attributes (5 categorical and 5 numerical) with different distributions.
- Loan [23]: Dataset from the Lending Club, a peer-to-peer lending company. The dataset includes information about accepted loans and attributes such as credit scores, income, and interest rates. In this dataset, we sample 2 million (from a total of 2.2) user records with ten attributes (5 categorical and 5 numerical) with different distributions.

Error measures. We use the Mean Absolute Error (MAE) to measure the accuracy of estimated answers. Given a set Q queries, it is computed as $MAE = \frac{1}{|Q|}\sum_{q\in Q}|f_q - \bar{f}_q|$, where f_q and $\bar{f}_q$ are the estimated and true answers of query q, respectively.

6.2 Evaluation Scenarios

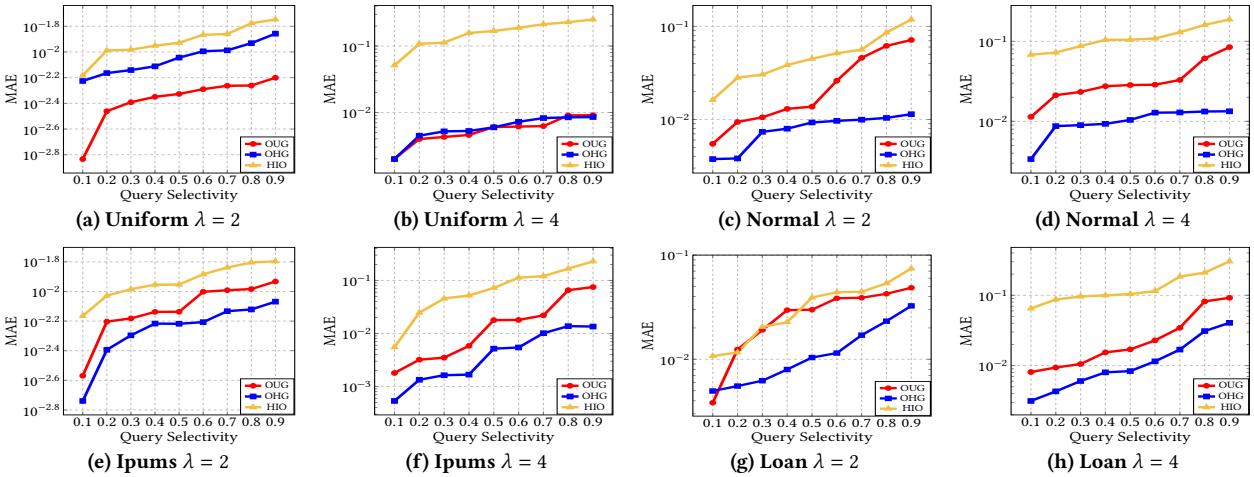
We vary the privacy budget ϵ, the query selectivity s, the number of attributes $|A|$, the domain d of attributes, the query dimension λ, and the number of users n. To experiment with different numbers of users, we generate multiple test datasets from the two synthetic datasets with the number of users ranging from 100k



(a) **Uniform** $\lambda = 2$ (b) **Uniform** $\lambda = 4$ (c) **Normal** $\lambda = 2$ (d) **Normal** $\lambda = 4$

(e) **Ipums** $\lambda = 2$ (f) **Ipums** $\lambda = 4$ (g) **Loan** $\lambda = 2$ (h) **Loan** $\lambda = 4$

Figure 1: Results on different datasets varying the privacy budget ϵ



(a) **Uniform** $\lambda = 2$ (b) **Uniform** $\lambda = 4$ (c) **Normal** $\lambda = 2$ (d) **Normal** $\lambda = 4$

(e) **Ipums** $\lambda = 2$ (f) **Ipums** $\lambda = 4$ (g) **Loan** $\lambda = 2$ (h) **Loan** $\lambda = 4$

Figure 2: Results on different datasets varying the query selectivity s

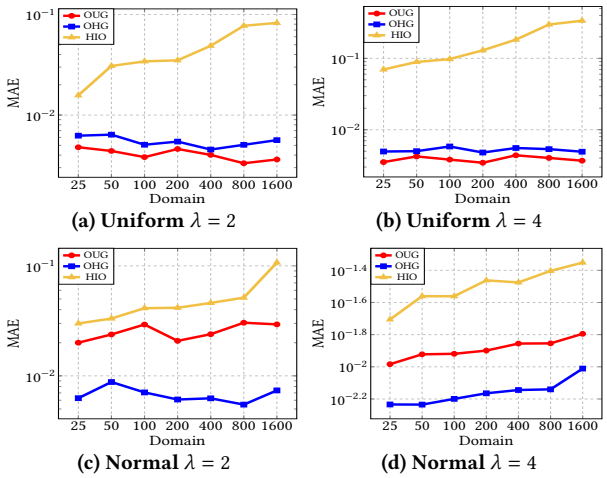
to 1000k. For evaluation varying different numbers of attributes and domain sizes, we generate multiple versions of these four datasets with the number of attributes ranging from 3 to 10, their domain sizes range from 2^4 to 2^{10}.

To the best of our knowledge, the most recent work to support queries with range and point constraints in the local setting, which is the focus of our work, is HIO [40]; thus we compare extensively to HIO. We set the branching factor $b = 4$. We set $\alpha_1 = 0.7$ and $\alpha_2 = 0.03$. To evaluate the performance of our approach, we randomly generate a set Q of λ-D queries and calculate their MAE. We generate queries with different selectivity, denoted by s, which means the ratio of the specified interval to the domain size for each queried numerical attribute. In all subsequent experiments, unless explicitly stated, the privacy budget has default value $\epsilon = 1.0$, the query selectivity $s = 0.5$, the number of attributes $d = 6$, the domain of each numerical attribute $d = 100$, the domain of each categorical attribute $|A| = 6$, the number of users $n = 10^6$, the query dimensions $\lambda = 2, 4$ and $|Q| = 10$.

6.2.1 Privacy budget. Figure 1 shows the results when varying the privacy budget ϵ. The strategy OUG performs better, as expected, in the uniform dataset, achieving better results than even OHG, especially for larger ϵ. In all other datasets, OHG is superior. Due to its refined estimation, which uses auxiliary 1-D grids, it obtains the best utility among all approaches. HIO had the larger MAE in all datasets.

6.2.2 Query Selectivity. In this experiment, we vary how selective the query is. We start with queries that select 10% of the domain and increase that to 90%. Figure 2 shows the results. For OHG and UOG, the broader the query, the greater the number of cells included in the answer. That impacts the utility since, to satisfy LDP, noise is added to each cell. The error increases on all strategies as the query becomes less selective. OHG and UOG perform better than HIO in all configurations. UOG has greater utility on uniform datasets, especially when $\lambda = 2$. For all other selectivity experiments, OHG is consistently the most accurate.

6.2.3 Domain size. Figure 3 shows the results when varying the domain size of all attributes. The numerical attributes vary from 25 to 1600, and the categorical vary from 2 to 8. The error of OUG/OHG tends to stay, within a certain margin, the same as the domain increases, unlike HIO which clearly increases considerably. In general, we found that there is no significant trend

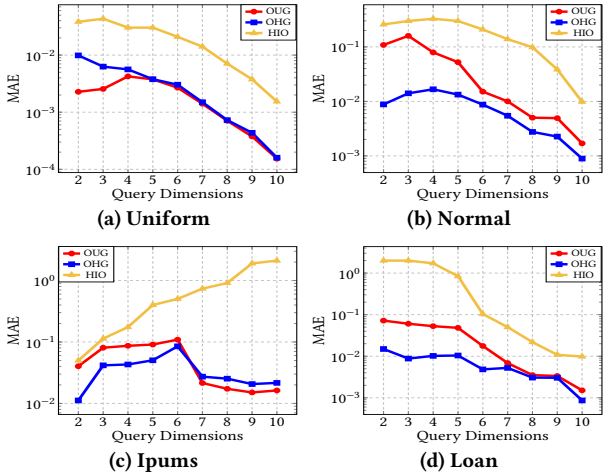


(a) Uniform $\lambda = 2$ (b) Uniform $\lambda = 4$

(c) Normal $\lambda = 2$ (d) Normal $\lambda = 4$

Figure 3: Results on different datasets varying the attributes' domain d

(increasing or decreasing) for the error as the domain increases in OUG/OHG. We can see that OHG performs the best on all datasets except the uniform, in which OUG has some advantages.



(a) Uniform (b) Normal

(c) Ipums (d) Loan

Figure 4: Results on different datasets varying the number of query dimensions λ

6.2.4 Query Dimension. Figure 4 shows the results when varying the number of query dimensions. We considered queries with 2-10 dimensions because it is the same setting that previous work [45] has used. We noticed that the more dimensions a query has, the closer the true answer gets to zero (*i.e.*, the query becomes very restrictive). In general, while we increase the number of query dimensions, the true answer value becomes very small, and the estimated frequencies as well (also due to post-processing). Thus, with both values approaching zero, the error tends to become smaller. In the IPUMS Figure 4c, we found that the error grows until a certain point where the true answer is not so small. We notice that from 2 to 6 dimensions, there is still a considerable amount of users that answer true to all predicates. From 7 to 10, the true answer value becomes very small with the queries becoming too restrictive from that point and the estimated frequencies as well, thus with both values, true and estimated, approaching zero the error tends to become smaller.

6.2.5 Number of attributes. Figure 5 shows the results when varying the number of attributes in the dataset from 4 to 10. The error on all strategies increases as the number of attributes becomes larger. To collect data under LDP, the total population is divided into a greater number of groups; the higher the number of attributes, the fewer users per group. Thus, the utility is reduced. HIO showed the more significant error in all configurations. UOG, as in other experiments, has good performance on uniform datasets. OHG has the lowest error on all the other datasets.

6.2.6 Number of Users. Figure 6 shows the results when varying the population size n. In the Loan dataset, we vary n from $10k$ to $1m$, and in all other datasets, n varies from $100k$ to $10m$. As expected, by increasing the number of users participating in the data collection, we increase the number of users per group, which leads to a better utility of all strategies under LDP. OUG has lower MAEs in the uniform distributed dataset. HIO shows to have the lowest utility compared to the optimized grid approaches. OHG achieves the best results among all the strategies.

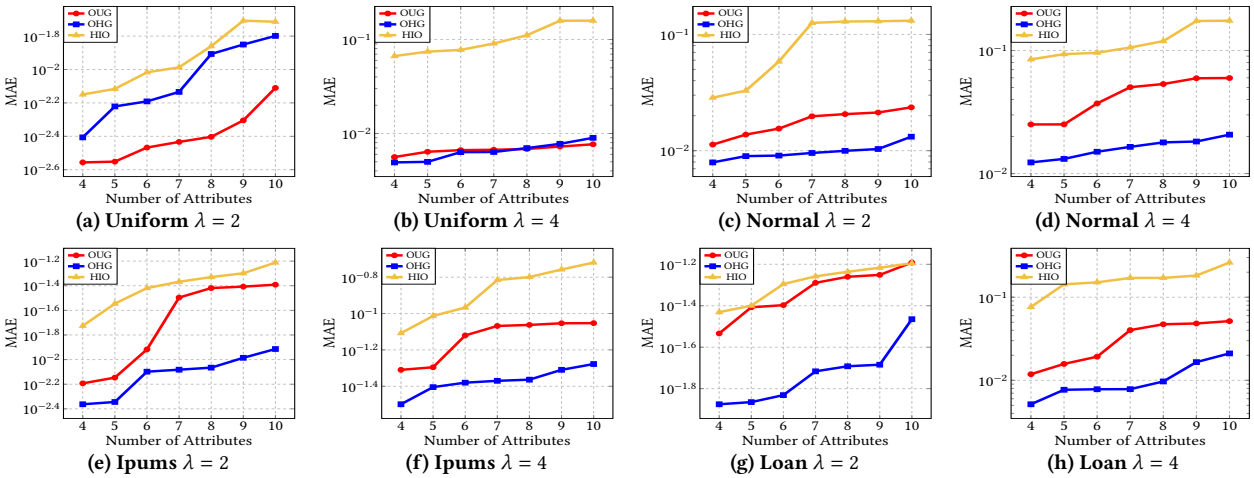
6.3 Adaptive Protocol Evaluation

In this experiment section, unlike Section 6.2, we consider queries with range constraints only, even though FELIP is designed to be a broader solution. We compare our solution to TDG/HDG [45]. We also evaluate if the adaptive protocol can reduce error in this scenario. So we compare our strategy with and without the adaptive protocol to the baselines TDG and HDG. We call OUG-OLH and OHG-OLH our optimized strategies that use the OLH protocol only. OUG and OHG are the proposed strategies with the adaptive protocol. TDG/HDG is designed around the OLH protocol. In this evaluation, we set the parameter values the same for all strategies. We set $\alpha_1 = 0.7$, $\alpha_2 = 0.03$, the number of users to 1000000, the query selectivity to 0.5, domain size to 100 for all attributes, query dimension to 3, and both datasets (normal, uniform) have six attributes.

Figure 7 (a) and (b) shows the results when using the uniform grid strategies. In both datasets, we notice that even without the adaptive protocol, OUG-OLH still performs better than TDG. That is explained by the fact that we construct more accurate sized grids than TDG. Also, we verify that the best uniform grid strategy is OUG, which chooses the best protocol for each grid. We can see that the non-uniformity error is an essential aspect since all uniform grid strategies have a more significant error on the normal dataset when compared to the uniform dataset. Figure 7 (c) and (d) shows the results when comparing the hybrid grid strategies. In this experiment, we note a more significant difference between our proposed approach OHG and the baseline HDG in both uniform and normal datasets. That is because, in hybrid grids, we are constructing more accurate sized 2-D grids and optimized 1-D grids which significantly impact the overall utility. The OHG-OLH still has higher utility when compared to HDG, but OHG that has the adaptive protocol achieves the best result among all strategies.

7 CONCLUSION

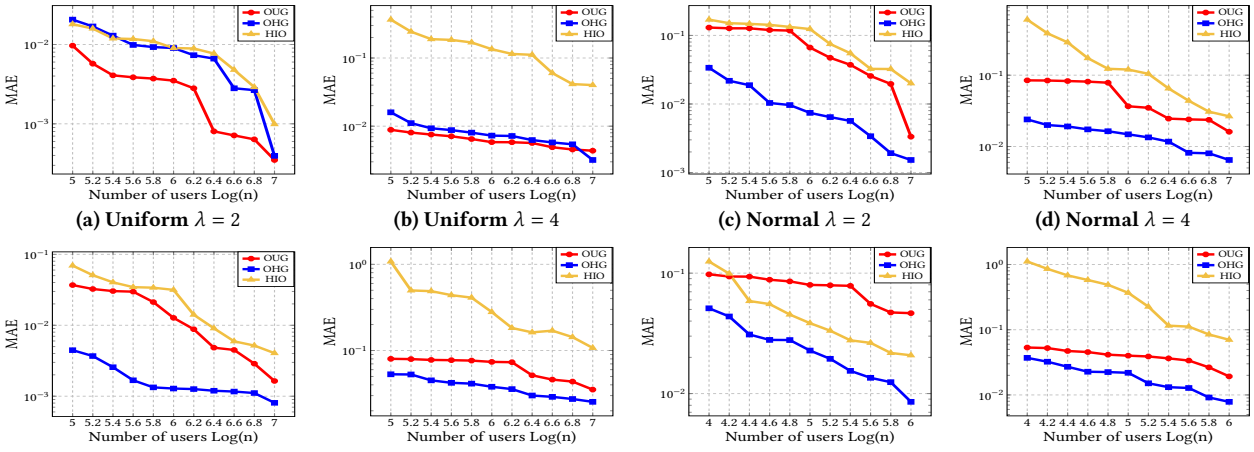
In this work, we study the frequency estimation problem, under LDP, over multidimensional data with categorical and numerical attributes. We carefully analyze the different factors that



(a) Uniform $\lambda = 2$ (b) Uniform $\lambda = 4$ (c) Normal $\lambda = 2$ (d) Normal $\lambda = 4$

(e) Ipums $\lambda = 2$ (f) Ipums $\lambda = 4$ (g) Loan $\lambda = 2$ (h) Loan $\lambda = 4$

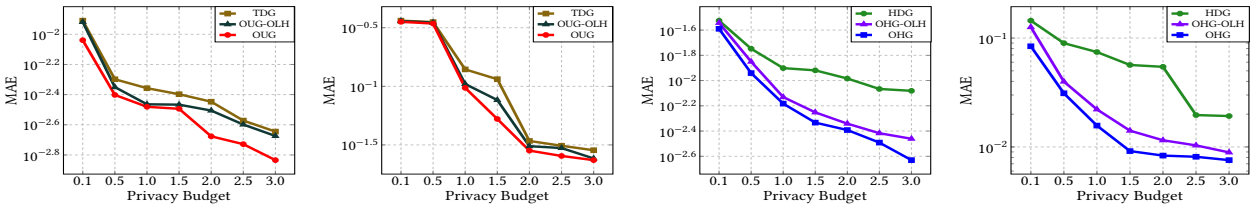
Figure 5: Results on different datasets varying the number of attributes



(a) Uniform $\lambda = 2$ (b) Uniform $\lambda = 4$ (c) Normal $\lambda = 2$ (d) Normal $\lambda = 4$

(e) Ipums $\lambda = 2$ (f) Ipums $\lambda = 4$ (g) Loan $\lambda = 2$ (h) Loan $\lambda = 4$

Figure 6: Results on different datasets varying the number of users Log(n)



(a) Uniform grids and Uniform dataset **(b) Uniform grids and Normal dataset** **(c) Hydrid grids and Uniform dataset** **(d) Hydrid grids and Normal dataset**

Figure 7: Results from queries with range constraints only when varying the privacy budget

impact the estimation error when using 1-D and 2-D grids. We propose optimizing the grids' construction, enabling users to report their private data by mapping the domain of the attributes using binning. Also, based on each grid characteristic, we propose to adaptively select the frequency oracle protocol that will offer the best accuracy. Our approach achieves high utility through extensive evaluation across various datasets and query scenarios.

There are several potential extensions to our work. First is how to enhance data decomposition to avoid cells with low true counts, so the noise does not dominate the estimation. Another research direction is studying how the aggregator can incorporate prior public knowledge about the dataset to modify the data collection process and improve utility. Finally, one can further investigate how to leverage low-dimensional grids to answer queries over data streams.

ACKNOWLEDGEMENTS

This work was partially supported by Lenovo, as part of its R&D investment under Brazil's Informatics Law, by a grant from Capes/Brazil and also by LSBD/UFC.

REFERENCES

[1] [n.d.]. *HIO.* https://github.com/vvv214/LDP_Protocols

[2] Asma Alnemari, Rajendra K. Raj, Carol J. Romanowski, and Sumita Mishra. 2021. Interactive Range Queries for Healthcare Data under Differential Privacy. In *2021 IEEE 9th International Conference on Healthcare Informatics (ICHI)*. 228–237.

[3] Héber H. Arcolezi, Jean-François Couchot, Bechara Al Bouna, and Xiaokui Xiao. 2022. Improving the utility of locally differentially private protocols for longitudinal and multidimensional frequency estimates. *Digital Communications and Networks* (2022).

[4] Sanjeev Arora, Elad Hazan, and Satyen Kale. 2012. The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. *Theory of Computing* 8, 6 (2012), 121–164.

[5] Raef Bassily. 2019. Linear queries estimation with local differential privacy. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 721–729.

[6] Xiang Cheng, Luoyang Fang, Liuqing Yang, and Shuguang Cui. 2017. Mobile Big Data: The Fuel for Data-Driven Wireless. *IEEE Internet of Things Journal* 4, 5 (2017), 1489–1516.

[7] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2018. Marginal Release Under Local Differential Privacy. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA.

[8] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 3574–3583.

[9] Linkang Du, Zhikun Zhang, Shaojie Bai, Changchang Liu, Shouling Ji, Peng Cheng, and Jiming Chen. 2021. AHEAD: Adaptive Hierarchical Decomposition for Range Query under Local Differential Privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21)*. Association for Computing Machinery, New York, NY, USA, 1266–1288.

[10] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2014. Local Privacy, Data Processing Inequalities, and Statistical Minimax Rates. arXiv:math.ST/1302.3203

[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, Shai Halevi and Tal Rabin (Eds.). Springer Berlin Heidelberg, 265–284.

[12] Alexander Edmonds, Aleksandar Nikolov, and Jonathan Ullman. 2020. The Power of Factorization Mechanisms in Local and Central Differential Privacy. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. Association for Computing Machinery, New York, NY, USA.

[13] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, New York, NY, USA.

[14] Xiaolan Gu, Ming Li, Yang Cao, and Li Xiong. 2019. Supporting Both Range Queries and Frequency Estimation with Local Differential Privacy. In *2019 IEEE Conference on Communications and Network Security (CNS)*. 124–132.

[15] Xiaolan Gu, Ming Li, Yueqiang Cheng, Li Xiong, and Yang Cao. 2020. PCKV: Locally Differentially Private Correlated Key-Value Data Collection with Optimized Utility. In *29th USENIX Security Symposium*. USENIX Association, 967–984.

[16] M. Emre Gursoy, Ling Liu, Ka-Ho Chow, Stacey Truex, and Wenqi Wei. 2022. An Adversarial Approach to Protocol Analysis and Selection in Local Differential Privacy. *IEEE Transactions on Information Forensics and Security* 17 (2022), 1785–1799.

[17] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A Simple and Practical Algorithm for Differentially Private Data Release. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'12)*. Curran Associates Inc., Red Hook, NY, USA, 2339–2347.

[18] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the Accuracy of Differentially Private Histograms through Consistency. *Proc. VLDB Endow.* 3, 1–2 (sep 2010), 1021–1032.

[19] Daeyoung Hong, Woohwan Jung, and Kyuseok Shim. 2021. Collecting Geospatial Data with Local Differential Privacy for Personalized Services. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 2237–2242.

[20] Noah Johnson, Joseph P. Near, and Dawn Song. 2018. Towards Practical Differential Privacy for SQL Queries. 11, 5 (2018).

[21] Matthew Joseph, Aaron Roth, Jonathan Ullman, and Bo Waggoner. 2018. Local Differential Privacy for Evolving Data. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 2381–2390.

[22] Martin Victor K, J. Immanuel Johnraja, Getzi Jeba Leelipushpam, J. Jebaveerasingh Jebadurai, and I. Bildass Santhosam. 2021. Security and Privacy Issues in the Internet of Things – A Survey. In *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*.

[23] Kaggle. [n.d.]. *All Lending Club loan data.* https://www.kaggle.com/datasets/wordsforthewise/lending-club

[24] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2008. What Can We Learn Privately?. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*. 531–540.

[25] Zitao Li, Tianhao Wang, Milan Lopuhaä-Zwakenberg, Ninghui Li, and Boris Škoric. 2020. Estimating Numerical Distributions under Local Differential Privacy. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA.

[26] Xiaojun Ye Shuo Feng Teng Wang Tao Bai Lin Sun, Jun Zhao. 2019. Conditional Analysis for Key-Value Data with Local Differential Privacy. *CoRR* abs/1907.05014 (2019).

[27] Gaoyuan Liu, Peng Tang, Chengyu Hu, Chongshi Jin, and Shanqing Guo. 2023. Multi-Dimensional Data Publishing With Local Differential Privacy. In *Proceedings 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28-31, 2023*. OpenProceedings.org, 183–194.

[28] Ryan McKenna, Raj Kumar Maity, Arya Mazumdar, and Gerome Miklau. 2020. A Workload-Adaptive Mechanism for Linear Queries under Local Differential Privacy. (2020).

[29] Frank D. McSherry. 2009. Privacy Integrated Queries: An Extensible Platform for Privacy-Preserving Data Analysis *(SIGMOD '09)*. Association for Computing Machinery, New York, NY, USA, 19–30.

[30] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Differentially private grids for geospatial data. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. 757–768.

[31] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2014. PriView: Practical Differentially Private Release of Marginal Contingency Tables. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*. Association for Computing Machinery, New York, NY, USA.

[32] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2014. PriView: Practical Differentially Private Release of Marginal Contingency Tables. Association for Computing Machinery, New York, NY, USA.

[33] Xuebin Ren, Liang Shi, Weiren Yu, Shusen Yang, Cong Zhao, and Zongben Xu. 2022. LDP-IDS: Local Differential Privacy for Infinite Data Streams. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 1064–1077.

[34] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A. McCann, and Philip S. Yu. 2018. LoPub : High-Dimensional Crowdsourced Data Publication With Local Differential Privacy. *IEEE Transactions on Information Forensics and Security* 13, 9 (2018), 2151–2166.

[35] Ronald Goeken Megan Schouweiler Steven Ruggles, Sarah Flood and Matthew Sobek. 2022. *IPUMS USA: Version 12.0.* Minneapolis, MN, USA. https://doi.org/10.18128/D010.V12.0

[36] Apple Differential Privacy Team. 2017. Learning with Privacy at Scale. (2017). https://machinelearning.apple.com/research/learning-with-privacy-at-scale

[37] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and Analyzing Multidimensional Data with Local Differential Privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 638–649.

[38] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally Differentially Private Protocols for Frequency Estimation. In *26th USENIX Security Symposium*. USENIX Association, Vancouver, BC, 729–745.

[39] Tianhao Wang, Joann Qiongna Chen, Zhikun Zhang, Dong Su, Yueqiang Cheng, Zhou Li, Ninghui Li, and Somesh Jha. 2021. Continuous Release of Data Streams under Both Centralized and Local Differential Privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21)*. Association for Computing Machinery, New York, NY, USA.

[40] Tianhao Wang, Bolin Ding, Jingren Zhou, Cheng Hong, Zhicong Huang, Ninghui Li, and Somesh Jha. 2019. Answering Multi-Dimensional Analytical Queries under Local Differential Privacy *(SIGMOD '19)*. ACM, New York, 159–176.

[41] Teng Wang, Xuefeng Zhang, Jingyu Feng, and Xinyu Yang. 2020. A Comprehensive Survey on Local Differential Privacy toward Data Statistics and Analysis. *Sensors* 20, 24 (2020).

[42] Yufei Wang and Xiang Cheng. 2022. PRISM: Prefix-Sum based Range Queries Processing Method under Local Differential Privacy. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 433–445.

[43] Stanley L. Warner. 1965. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *J. Amer. Statist. Assoc.* 60, 309 (1965), 63–69.

[44] Gewei Zheng Hui Li Fenghua Li Xiaoguang Li, Haonan Yan. 2022. Key-Value Data Collection with Distribution Estimation under Local Differential Privacy. *Security and Communication Networks* (2022).

[45] Jianyu Yang, Tianhao Wang, Ninghui Li, Xiang Cheng, and Sen Su. 2020. Answering Multi-Dimensional Range Queries under Local Differential Privacy. *Proc. VLDB Endow.* 14, 3 (2020).

[46] Qingqing Ye, Haibo Hu, Xiaofeng Meng, Huadi Zheng, Kai Huang, Chengfang Fang, and Jie Shi. 2021. PrivKVM*: Revisiting Key-Value Statistics Estimation with Local Differential Privacy. *IEEE Transactions on Dependable and Secure Computing* (2021).

[47] Yutong Ye, Min Zhang, and Dengguo Feng. 2021. Collecting Spatial Data Under Local Differential Privacy. In *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*. 120–127.

[48] Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, and Jiming Chen. 2018. CALM: Consistent Adaptive Local Marginal for Marginal Release under Local Differential Privacy *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 212–229.

[49] Tianqing Zhu, Gang Li, Wanlei Zhou, and Philip S. Yu. 2017. Differentially Private Data Publishing and Analysis: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 29, 8 (2017), 1619–1638.