Answering Multi-Dimensional Analytical Queries under Local Differential Privacy

Tianhao Wang*
Purdue University
tianhaowang@purdue.edu

Bolin Ding
Alibaba Group
bolin.ding@alibaba-inc.com

Jingren Zhou
Alibaba Group
jingren.zhou@alibaba-inc.com

Cheng Hong
Alibaba Group
vince.hc@alibaba-inc.com

Zhicong Huang
Alibaba Group
zhicong.hzc@alibaba-inc.com

Ninghui Li
Purdue University
ninghui@cs.purdue.edu

Somesh Jha
University of Wisconsin
jha@cs.wisc.edu

ABSTRACT

Multi-dimensional analytical (MDA) queries are often issued against a fact table with predicates on (categorical or ordinal) dimensions and aggregations on one or more measures. In this paper, we study the problem of answering MDA queries under local differential privacy (LDP). In the absence of a trusted agent, sensitive dimensions are encoded in a privacy-preserving (LDP) way locally before being sent to the data collector. The data collector estimates the answers to MDA queries, based on the encoded dimensions. We propose several LDP encoders and estimation algorithms, to handle a large class of MDA queries with different types of predicates and aggregation functions. Our techniques are able to answer these queries with tight error bounds and scale well in high-dimensional settings (*i.e.*, error is polylogarithmic in dimension sizes). We conduct experiments on real and synthetic data to verify our theoretical results, and compare our solution with marginal-estimation based solutions.

ACM Reference Format:
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. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3299869.3319891

*Work done at Alibaba Group.

SIGMOD '19, June 30-July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5643-5/19/06…$15.00
https://doi.org/10.1145/3299869.3319891

1 INTRODUCTION

Large volumes of users' data about their profiles and activities are collected by enterprises to make informed business decisions. In order to meet users' expectation of their privacy, applications and services must provide rigorous privacy guarantees on how their data is collected and analyzed. Differential privacy (DP) [14] has emerged as the *de facto* standard for privacy guarantees, and is being used by, *e.g.*, Apple [32], Google [16], Uber [23], and Microsoft [9].

A well-studied DP model is in the centralized setting, where a *trusted data collector* obtains exact data from users, and injects noise in the analytical process to guarantee DP. In the absence of such a trusted party, users prefer not to have their private data leave their devices in an unprotected form, and thus, the centralized setting of DP is no longer applicable. In such scenarios, one can adopt the *local differential privacy* model (LDP) [12]. Each user's private data is encoded by a randomized algorithm before being sent to the data collector. LDP guarantees that the likelihood of any specific output of the algorithm varies little with input, *i.e.*, the private data. In this way, users do not need to trust the data collector.

Data model and application scenarios. LDP fits the class of analytical applications considered in this paper well. Suppose a number of individuals use a service in the cloud, and each user generates some *multi-dimensional data* during the service. Some dimensions, called *measure attributes*, are *naturally known* to the service provider, *e.g.*, active time and purchase amount for the billing purpose; some other dimensions are *sensitive*, *e.g.*, income and location, and users prefer to have them collected only in a privacy-preserving way; the remaining ones are *non-sensitive*. On the other side, the provider wants to analyze how the service performs by issuing analytical queries that aggregate measure attributes under constraints on sensitive dimensions. While all the dimensions will never be released to the public and the analytics are conducted internally by the provider, the provider needs to guarantee that the sensitive dimensions are handled properly by providing an LDP data collection algorithm that runs on each user's device. A motivating example follows.

	Age	Salary	State	OS	ActiveTime	Purchase
t_1	30	50K	NY	Win	1.6h	\$120
t_2	60	80K	WA	iOS	1.2h	\$100
t_3	50	90K	NY	Win	1.0h	\$100
t_4	40	70K	NY	iOS	1.8h	\$100

Table 1: A relational table T with sensitive dimensions

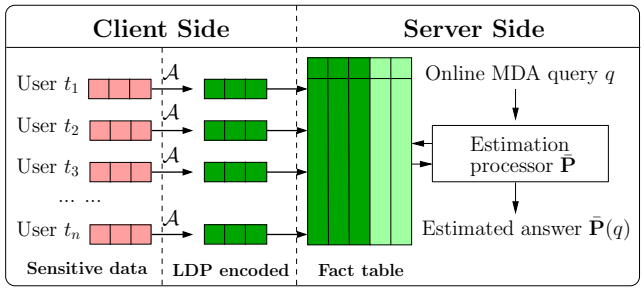


Figure 1: Answering MDA queries under LDP

Example 1.1. *The multi-dimensional data model of users in an online shopping app is shown in Table 1. Users are anonymous. Measure attributes* ActiveTime *(how much time a user spent in the app) and* Purchase *(amount of money spent in the app), are inherently known to the service provider.* Age, Salary, *and* State *are sensitive dimensions, and attribute* OS *is non-sensitive. The service provider wants to analyze how much money is spent by a specific group of users using a query:*

SELECT SUM(Purchase) FROM T (1)
WHERE Age $\in$ [30, 40] AND Salary $\in$ [50K, 150K].

In this paper, we study how to (approximately) answer a class of *multi-dimensional analytical (MDA) queries*, while each user's sensitive data is collected under LDP. An *MDA query* is a SQL query with aggregation (*e.g.*, COUNT, SUM, or AVG) on measure attributes (accessible by service provider), and a predicate with equality and range constraints on sensitive dimensions (to be collected under LDP).

Overview of our solution (Figure 1). On the client side, each user holds some multi-dimensional data, and runs an LDP encoding algorithm $\mathcal{A}$ on the sensitive dimensions. The output is sent to the server. A *fact table* on the server side is a combination of LDP encoded dimensions collected from users and dimensions/measures that are non-sensitive or known to the server. An arbitrary number of MDA queries can be issued as the privacy is guaranteed for each user before her/his data is collected. Since sensitive dimensions are encoded with random noise injected, we need an algorithm $\bar{\mathbf{P}}$ to estimate the query answers (with bounded errors).

Challenges. Answering MDA queries under LDP is closely related to the recent line of works on releasing marginals under LDP [8, 30, 41]. A marginal table records the (empirical) probability distribution between a set of dimensions. Each row in a marginal is equivalent to a COUNT query. In Table 1, *e.g.*, a row in the 2-way marginal (Age, Salary) is:

SELECT Age, Salary, COUNT(*) FROM T (2)
WHERE Age = 30 AND Salary = 50K.

LDP marginals can be adapted to answer an MDA query by summing up marginal rows such as (2) covered by the query. For example, to answer a COUNT query with Age $\in$ [30, 40] AND Salary $\in$ [50K, 150K], assuming Salary takes values in thousands, we need to add up 11×101 rows in the 2-way marginal on (Age, Salary), with the error variance potentially amplified 11×101 times. More generally, suppose there are d ordinal dimensions, each with m distinct values, the worst-case squared error is proportional to $O(m^d)$, as we may need to sum up m^d such rows under range constraints. The worst-case error could be exorbitant when m is large.

LDP frequency oracles [4, 5, 9, 13, 35] are in another related line. Here, each user has a private value and encodes it with noise in an LDP way. Each LDP encoded value is sent to the server, and on the server, we want to estimate the frequency of a given value (*i.e.*, how many users hold this value). Some very simple MDA queries, *e.g.*, in Table 1,

SELECT COUNT(*) FROM T WHERE State = NY

can be translated into frequency queries, *e.g.*, the frequency of "NY". However, it was unknown how to handle other aggregations, *e.g.*, SUM, with frequency oracles and how to handle complicated predicates, *e.g.*, range constraints, on multiple dimensions without blowing up estimation errors.

Contributions. We aim to process MDA queries with both privacy and accuracy guarantees. To this end, we first revisit LDP frequency oracles, which are building blocks in our solution. We propose a weighted frequency oracle: each user is associated with a public weight and holds a private value; an LDP version of the value is sent to the server. We want to estimate, for a given value, the total weight of users holding this value. It will be used to handle more general aggregations, *e.g.*, SUM, AVG, and STDEV, of measures attributes – each measure is regarded as the weight of each user.

Our solution $(\mathcal{A}, \bar{\mathbf{P}})$ relies on a hierarchical decomposition of the ordinal dimensions into sub-intervals, and can reduce the worst-case squared error from $O(m^d)$ in the marginal-based solution to $\log^{O(d)} m$ (keeping other terms that are dependent on data size and privacy budget the same).

The decomposition schema on each single dimension is not new, which has been used to answer range COUNT queries in the centralized DP setting firstly by [22]. Our new contribution is a novel way to incorporate weighted frequency oracles in the hierarchy, so that the error in the estimated answer is polylogarithmic in the cardinality of dimensions (instead of polynomial if we adopt LDP marginals in a naive way). A more important contribution is that we

extend the decomposition schema for d dimensions so that the error is still polylogarithmic, *i.e.*, $\log^{O(d)} m$.

The main idea behind the hierarchical decomposition is that the privacy budget is partitioned across a polylogarithmic number of levels in the hierarchy, and the query is partitioned into a polylogarithmic number of sub-queries on sub-intervals in the hierarchy. An alternative is that each user encodes and sends only one *randomly* selected level in the hierarchy, so that less LDP noise is added to that level but sampling noise is introduced in the estimation. By carefully exploring the properties of our weighted frequency oracles, a bit surprisingly, this alternative can boost the accuracy of estimated answers to MDA queries by orders of magnitude, which we will demonstrate theoretically and empirically.

Finally, we consider the case when the total number of private dimensions is large, but the number of dimensions in the query is small (which is often true in practice). By exploring variants of our weighted frequency oracles, we propose an LDP mechanism whose estimation error is only exponential in the number of dimensions in the query, but less heavily dependent on the total number of dimensions.

Organization. Section 2 presents the data model and the privacy guarantee. Section 3 introduces our weighted LDP frequency oracle. Section 4 gives our solution for one dimension, which will be generalized and optimized for multiple dimensions in Section 5. Experimental results are reported in Section 6. We present extensions to our solution in Section 7, and discuss related work in Section 8.

2 PRELIMINARIES

We first introduce MDA queries and the privacy guarantee.

2.1 Multi Dimensional Model and Analytics

Each *user* contributes a *tuple* t to a table with a set of attributes, called *dimensions* or *measures*. A dimension, denoted by D, appears in predicates, and a measure, denoted by M, is aggregated in analytical questions. We also use D or M to denote the *domain* (the set of possible values) of an attribute, and $t[D]$ and $t[M]$ are the attribute values in a tuple.

Multi-dimensional analytical (MDA) queries. Let T be the relational table, called *fact table*. We focus on the following class of *multi-dimensional analytical queries*:

$$Q_T(\mathsf{F}(M), \mathsf{C}): \ \text{SELECT } \mathsf{F}(M) \text{ FROM } T \text{ WHERE } \mathsf{C} \qquad (3)$$

- *Aggregation* $\mathsf{F}(M)$ is $\mathsf{COUNT}(*)$, $\mathsf{SUM}(M)$, or $\mathsf{AVG}(M)$. We focus on $\mathsf{SUM}(M)$ in the main text (COUNT is a special case and AVG can be derived from the other two).
- *Predicate* C consists of *point constraints* "$D_i = v_i$" for *categorical dimensions*, and *range constraints* "$D_i \in [l_i, r_i]$" for *ordinal dimensions*. We will first focus on conjunctions (AND-only) of one or more such constraints.

We will introduce how to generalize our solution for the other aggregate functions and AND-OR predicates in Section 7.

2.2 Local Differential Privacy (LDP)

A *server* collects tuples from users into T. In the application scenarios introduced in Section 1, some dimensions are considered *sensitive* by users, and thus need to be collected in a privacy-preserving way; measures are *public* or *known to the server* (*e.g.*, how much time a user spends on a service is known to the service provider for the billing purpose).

Trust model of LDP. Users *do not trust* the server and require formal privacy guarantees before they are willing to send their dimension values. We adopt the *local model of differential privacy* (LDP) [13], which is also called randomized response model [39], γ-amplification [17], or FRAPP [2]. Under LDP, sensitive dimensions in a tuple t from each user are encoded by a randomized algorithm $\mathcal{A}$, and the output $\mathcal{A}(t)$, called an *LDP report*, is sent to the server. Intuitively, LDP guarantees that, no matter what $\mathcal{A}(t)$ is, it is approximately equally as likely to have come from t as any other t' differing from t in one or more sensitive dimensions. Hence, if $\mathcal{A}(t)$, instead of t, is collected, t's information on sensitive dimensions is protected (to some degree measured by the privacy budget ϵ). More formally, suppose $D_1, \ldots, D_d$ are sensitive dimensions, we provide the following LDP guarantee:

Definition 1 (Local Differential Privacy [13]). *A randomized algorithm $\mathcal{A}$ is ϵ-locally differentially private or ϵ-LDP, if for any pair of different tuples t and t', with $t[D_i] \neq t'[D_i]$ for at least one $i \in \{1, \ldots, d\}$, and any $O \subseteq \text{Range}(\mathcal{A})$,*

$$\Pr[\mathcal{A}(t) \in O] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(t') \in O].$$

In comparison to previous work on LDP, we identify a separation between sensitive (but users volunteer to contribute under LDP) and non-sensitive (*e.g.*, measures that server will always know), and utilize the non-sensitive information.

Example 2.1. *The multi-dimensional data model in Table 1 has six attributes. D_1 and D_2 are ordinal dimensions; D_3 and D_4 are categorical dimensions. There are also two numeric measures M_1 and M_2. The query in Example 1.1 is an MDA query with two range constraints on D_1 and D_2.*

D_1-D_3 are sensitive dimensions, and thus need to be collected under LDP. LDP guarantees that we cannot distinguish between two users with (30, 50K, NY) and (40, 70K, NY) based on their LDP reports, and thus, their dimension values are protected.

Sequential composability. An important property of DP, sequential composability [26], also holds for LDP (considering a dataset with one row in the centralized model), and will be used repeatedly in the rest of this paper.

Proposition 2 (Directly from [26]). *Suppose $\mathcal{A}_i$ satisfies ϵ_i-LDP, the algorithm $\mathcal{A}$ which simultaneously releases $\mathcal{A}(t) = \langle \mathcal{A}_1(t), \ldots, \mathcal{A}_k(t) \rangle$ satisfies $\sum_{i=1}^{k} \epsilon_i$-LDP.*

2.3 Private Multi-Dimensional Analytics

We study the task of *private multi-dimensional analytics* in this paper. An *LDP mechanism* for this task is a pair $(\mathcal{A}, \bar{\mathbf{P}})$.

Client side (LDP encoder $\mathcal{A}$). Each user t runs an ϵ-LDP algorithm $\mathcal{A}^\epsilon$ on her/his sensitive dimensions, and sends the output $\mathcal{A}^\epsilon(t)$, *i.e.*, the LDP report, to the server. If ϵ is clear from the context, we simply write $\mathcal{A}^\epsilon$ as $\mathcal{A}$.

Server side (estimation processor $\bar{\mathbf{P}}$). There are n users whose tuples form a fact table $T = \{t_1, \ldots, t_n\}$. The server receives $\mathcal{A}(T) = \{\mathcal{A}(t_1), \ldots, \mathcal{A}(t_n)\}$. An MDA query $q = Q_T(\mathsf{F}(M), \mathsf{C})$, in the form of (3), can be approximately answered on *LDP reports* $\mathcal{A}(T)$ and *other public attributes of* T using an estimation algorithm $\bar{\mathbf{P}}$. Let $\bar{\mathbf{P}}(q)$ be the estimate.

An arbitrary number of MDA queries can be issued on the server, since LDP is preserved for each user t on the report $\mathcal{A}(t)$ and $\bar{\mathbf{P}}$ can be regarded as "post-processing" of reports.

Error metric. Let q also denote the exact answer to an MDA query. Let $\mathrm{Err}(\bar{\mathbf{P}}(q))$ be the *expected (over randomness in $\mathcal{A}$) squared error* $\mathbf{E}[(\bar{\mathbf{P}}(q) - q)^2]$ in $\bar{\mathbf{P}}(q)$. If $\bar{\mathbf{P}}(q)$ is an unbiased estimator of q, $\mathrm{Err}(\bar{\mathbf{P}}(q)) = \mathbf{Var}\left[\bar{\mathbf{P}}(q)\right]$. Our goal is to bound $\mathrm{Err}(\bar{\mathbf{P}}(q))$ for the supported MDA queries.

3 WEIGHTED FREQUENCY ORACLE

We introduce building blocks used in our LDP mechanisms for MDA. We first introduce *weighted frequency queries* and their relationship to MDA. We present an LDP mechanism (*weighted frequency oracle*) for such queries. We introduce how to estimate weighted frequencies if a random sample of users send their LDP reports – this twist will be used (in Sections 4-5) to boost the accuracy of MDA. Finally, a marginal-based solution for answering MDA is presented.

3.1 Weighed Frequency Queries and MDA

In a multi-dim data model, each user t has a private dimension $t[D] \in D$ and a public measure $t[M] \in \mathbb{R}$. A *weighted frequency query* asks, for a set of users S, what is the total measure of users with a given dimension value v, *i.e.*,

$$\mathbf{f}_S^M(v; D) = \sum_{t \in S \wedge t[D]=v} t[M] \tag{4}$$

$$\Leftrightarrow \text{ SELECT SUM}(M) \text{ FROM } S \text{ WHERE } D = v. \tag{5}$$

We write $\mathbf{f}_S^M(v; D)$ as $\mathbf{f}_S^M(v)$ if D is clear from the context.

3.2 An LDP Frequency Oracle (FO)

As $t[D]$ is sensitive, each user uses an ϵ-LDP algorithm $\mathcal{A}_{\mathsf{FO}}$ to encode $t[D]$ as $\mathcal{A}_{\mathsf{FO}}(t[D])$ before sending it to the server. For a user set S, the server obtains an estimator $\bar{\mathbf{f}}_S^M(v)$ of $\mathbf{f}_S^M(v)$ from the LDP reports $\mathcal{A}_{\mathsf{FO}}(S) = \{\mathcal{A}_{\mathsf{FO}}(t[D])\}_{t \in S}$ and the public measure M. An LDP *weighted frequency oracle* refers to a pair of encoder and estimator $(\mathcal{A}_{\mathsf{FO}}, \bar{\mathbf{f}}^M)$.

3.2.1 Existing Unweighted Frequency Oracles $(\mathcal{A}_{\mathsf{FO}}, \bar{\mathbf{f}})$. When $t[M] = 1$ for all users, $\mathbf{f}_S^M(v)$ is equal to the (*unweighted*) frequency of v, $\mathbf{f}_S(v)$, which is equivalent to a COUNT query:

$$\mathbf{f}_S(v) = \sum_{t \in S \wedge t[D]=v} 1 = |\{t \in S \mid t[D] = v\}|. \tag{6}$$

There have been previous works, *e.g.*, [4, 5, 9, 13, 35], on LDP unweighted frequency oracles with asymptotically optimal error. We use OLH (optimal local hashing) from [35]. It maps $t[D]$ from D to a smaller domain using a hash function H *randomly* picked from a universal family, and *randomly perturb* $H(t[D])$ to a different value y in the domain with certain probability. $\mathcal{A}_{\mathsf{FO}}(t[D]) = \langle H, y \rangle$ is the LDP report from t. It is ϵ-LDP as long as the perturbation distribution is "flat" enough. For the completeness, we describe details about $\mathcal{A}_{\mathsf{FO}}$ and the estimator $\bar{\mathbf{f}}$ in Algorithm 3 in Appendix A. We state its asymptotically optimal error bound in Lemma 3.

Lemma 3 (OLH [35]). *There is an ϵ-LDP frequency oracle $(\mathcal{A}_{\mathsf{FO}}, \bar{\mathbf{f}})$ (Algorithm 3). For any dimension value $v \in D$, $\bar{\mathbf{f}}_S(v)$ is an unbiased estimator of $\mathbf{f}_S(v)$, and we can bound its error:*

$$\mathrm{Err}(\bar{\mathbf{f}}_S(v)) = \mathbf{E}\left[\left(\bar{\mathbf{f}}_S(v) - \mathbf{f}_S(v)\right)^2\right] = \frac{4|S|e^\epsilon}{(e^\epsilon - 1)^2} + \mathbf{f}_S(v).$$

3.2.2 Our Weighted Frequency Oracle $(\mathcal{A}_{\mathsf{FO}}, \bar{\mathbf{f}}^M)$. We now show how to generalize an unweighted frequency oracle into a weighted frequency oracle. We use OLH [35] in our work. Those in [4] can be also applied and analyzed similarly.

The idea behind $(\mathcal{A}_{\mathsf{FO}}, \bar{\mathbf{f}}^M)$ is to partition users into groups by their measures. Let $S_x = \{t \in S \mid t[M] = x\}$ be the group of users in S with measure equal to x. For a dimension value v, we establish the relationship between $\mathbf{f}$ and $\mathbf{f}^M$ via S_x:

$$\mathbf{f}_S^M(v) = \sum_{\text{distinct } x} \mathbf{f}_{S_x}^M(v) = \sum_{\text{distinct } x} x \cdot \mathbf{f}_{S_x}(v). \tag{7}$$

We use an unweighted frequency oracle $(\mathcal{A}_{\mathsf{FO}}, \bar{\mathbf{f}}_{S_x})$ to encode $t[D] = v$ in each S_x. We can then approximate $\mathbf{f}_S^M(v)$ with the estimator $\bar{\mathbf{f}}^M$ by combining the frequency estimates:

$$\bar{\mathbf{f}}_S^M(v) = \sum_{\text{distinct } x} x \cdot \bar{\mathbf{f}}_{S_x}(v), \tag{8}$$

Example 3.1. *Consider such a query against T in Table 1:*

SELECT SUM(Purchase) FROM T WHERE State = NY

The answer is $120 + 100 + 100 + \ldots$. *We have defined* $S_{120} = \{t_1\}$ *and* $S_{100} = \{t_2, t_3, t_4\}$. *The frequency of "NY" in* S_{120}, $\mathbf{f}_{S_{120}}(\mathrm{NY}) = 1$, *and, in* S_{100}, $\mathbf{f}_{S_{100}}(\mathrm{NY}) = 2$. *Thus, the answer can be also calculated* $120 \times 1 + 100 \times 2 + \ldots$, *and to estimate the answer, we can estimate* $\mathbf{f}_{S_{120}}(\mathrm{NY})$ *and* $\mathbf{f}_{S_{100}}(\mathrm{NY})$ *instead.*

Proposition 4 (Weighted Frequency Oracle). *Mechanism* $(\mathcal{A}_{\text{FO}}, \bar{\mathbf{f}}^M)$ *is* ϵ*-LDP. For a set of users* S *and a value* $v \in D$, $\bar{\mathbf{f}}_S^M(v)$ *is an unbiased estimator of* $\mathbf{f}_S^M(v)$. *Let* $M_S^2 = \sum_{t \in S} t[M]^2$ *and* $M_S^2(v) = \sum_{t \in S \wedge t[D]=v} t[M]^2$. *The error is*

$$\text{Err}(\bar{\mathbf{f}}_S^M(v)) = \mathbf{E}\left[\left(\bar{\mathbf{f}}_S^M(v) - \mathbf{f}_S^M(v)\right)^2\right] = \frac{4M_S^2 e^\epsilon}{(e^\epsilon - 1)^2} + M_S^2(v)$$
$$\leq \frac{M_S^2(e^\epsilon + 1)^2}{(e^\epsilon - 1)^2} = \text{O}\left(\frac{|S|\Delta^2}{\epsilon^2}\right) \text{ when } \epsilon \text{ is small,}$$

where Δ *is the range of* M, i.e., $\Delta = \max(M) - \min(M)$. *Moreover, estimation errors for two different values are additive:*

$$\mathbf{Var}\left[\bar{\mathbf{f}}_S^M(u) + \bar{\mathbf{f}}_S^M(v)\right] = \mathbf{Var}\left[\bar{\mathbf{f}}_S^M(u)\right] + \mathbf{Var}\left[\bar{\mathbf{f}}_S^M(v)\right].$$

Note that the above result does not depend on how large or how small each S_x is; in an extreme case, even if every distinct value of measure M appears only once ($|S_x| = 1$ for each x), we still have the same expected error.

3.3 Oracle Running on Random Samples

If we ask a random sample of users to report their private values $t[D]$ using $\mathcal{A}_{\text{FO}}$, we can still estimate the weighted frequency of a value v in this sample using $\bar{\mathbf{f}}^M$, and then scale the estimate up for the whole population – what is the accuracy loss in this procedure? This twist will be used in our mechanisms to boost its performance.

More formally, for a set of users S, we first randomly partition S into $S_1, \ldots, S_k$ (each user in S randomly chooses $i \in \{1, \ldots, k\}$, with equal probability $1/k$, and joins S_i). We run the weighted frequency oracle $(\mathcal{A}_{\text{FO}}, \bar{\mathbf{f}}^M)$ only on one sample, say, S_1. For a dimension value v, we can estimate its weighted frequency $\mathbf{f}_S^M(v)$ in S using $\bar{\mathbf{f}}^M$ on S_1. Define

$$\tilde{\mathbf{f}}_{S,1/k}^M(v) = k \cdot \bar{\mathbf{f}}_{S_1}^M(v), \tag{9}$$

where S_1 (or any of $S_1, \ldots, S_k$ generated above) is a random sample of S with sampling rate $1/k$. $\tilde{\mathbf{f}}_{S,1/k}^M(v)$ in (9) is an unbiased estimator of weighted frequency $\mathbf{f}_S^M(v)$, because

$$\mathbf{E}\left[\tilde{\mathbf{f}}_{S_1}^M(v)\right] = \mathbf{E}\left[\mathbf{E}\left[\tilde{\mathbf{f}}_{S_1}^M(v) \,\middle|\, S_1\right]\right] = \mathbf{E}_{S_1}\left[\mathbf{f}_{S_1}(v)\right] = \frac{1}{k} \cdot \mathbf{f}_S(v).$$

The second equality is from the unbiasedness of $\bar{\mathbf{f}}^M$ and the third one is due to the sampling process. The error in $\tilde{\mathbf{f}}_{S,1/k}^M(v)$ comes from two sources, one due to LDP noise and the other due to sampling process. We can bound it as follows.

Proposition 5 (Accuracy Loss on Samples). $\tilde{\mathbf{f}}_{S,1/k}^M(v)$ *is an unbiased estimator of* $\mathbf{f}_S^M(v)$, *and the error is bounded as*

$$\text{Err}(\tilde{\mathbf{f}}_{S,1/k}^M(v)) = \frac{4kM_S^2 e^\epsilon}{(e^\epsilon - 1)^2} + (2k-1)M_S^2(v)$$
$$\leq \frac{2kM_S^2(e^{2\epsilon} + 1)}{(e^\epsilon - 1)^2} = \text{O}\left(\frac{k|S|\Delta^2}{\epsilon^2}\right) \text{ when } \epsilon \text{ is small,}$$

where Δ *is the range of* M, i.e., $\Delta = \max(M) - \min(M)$.

3.4 Answering MDA via LDP Marginals

Mechanisms to estimate LDP marginals [8, 30, 41] focus on COUNT queries. However, they can be adapted to handle MDA queries via a transition that is similar to (7).

$(\mathcal{A}_{\text{MG}}, \bar{\mathbf{P}}_{\text{MG}})$: To answer an MDA query $Q_T(\text{F}(M), \text{C})$ in (3), we first partition T by measure M into sub-tables $T_x = \{t \in T \mid t[M] = x\}$. We use marginals estimated under LDP to count how many tuples in T_x satisfy the predicate C as $\bar{n}_x$ by summing up cells in the marginal on dimensions in C. For a SUM query Q_T, its answer can be estimated as $\sum_x x \cdot \bar{n}_x$.

Let's consider the SUM query in Example 1.1. We partition T by Purchase. In a sub-table, *e.g.*, $T_{\$100}$, estimate the marginal

SELECT COUNT(*) FROM $T_{\$100}$ GROUP BY Age, Salary.

Sum up (11×101) rows in the above 2-way marginal with Age $\in [30, 40] \wedge$ Salary $\in$ [50K, 150K] to obtain $\bar{n}_{\$100}$, which contributes a term ($\$100 \cdot \bar{n}_{\$100}$) in the estimated answer.

Error analysis. We can analyze errors in the above marginal-based solution for data with one sensitive dimension using Proposition 4. Let D be a sensitive ordinal dimension, and we want to handle MDA queries with range constraints,

$$q: \text{SELECT SUM}(M) \text{ FROM } T \text{ WHERE } D \in [l, r].$$

We partition T by M. For each distinct $x \in M$, in the estimated LDP marginal of T_x on the dimension D, we sum up rows with $D \in [l, r]$, each contributing x in the answer. Since a 1-way marginal can be optimally estimated with a frequency oracle, the estimated answer to q is equivalent to:

$$\sum_{\text{distinct } x} \left(\sum_{v \in [l,r]} x \cdot \bar{\mathbf{f}}_{T_x}(v) \right) = \sum_{v \in [l,r]} \bar{\mathbf{f}}_T^M(v). \tag{10}$$

The error now depends on how many distinct values of D we have within $[l, r]$. If D has m distinct values, from Proposition 4, the error of the above estimation is:

$$(r - l + 1) \cdot \text{Err}(\bar{\mathbf{f}}_T^M) = \Theta\left(m|T|\Delta^2/\epsilon^2\right) \tag{11}$$

with a linear dependency on $r - l + 1$ or m.

Suppose there are d sensitive dimensions, each with m distinct values, the worst-case error in the above solution is proportional to m^d, as we may need to sum up m^d marginal rows under range constraints on these dimensions.

In Sections 4-5, we propose new mechanisms to remove the linear/polynomial dependency on m in the error, via careful query decomposition and privacy-budget partitioning. Their error is poly-logarithmically dependent on m.

4 MDA WITH ONE PRIVATE DIMENSION

We first focus on one sensitive dimension, and will generalize our solution for multi-dimensional MDA in Section 5.

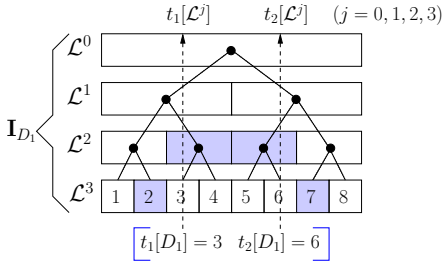


Figure 2: Hierarchy of intervals and HI mechanism

4.1 Hierarchical-Interval (HI) Mechanism

We propose a mechanism $(\mathcal{A}_{\mathsf{HI}}, \bar{\mathbf{P}}_{\mathsf{HI}})$, whose one-dimensional version is inspired by the structure of a binary search tree, to ensure that the error is sublinear in m. Similar structures have also been used by previous work, *e.g.*, [22, 28], to answer range counting queries in the centralized DP setting.

Hierarchy of intervals. Suppose the ordinal dimension D has m distinct values, in the order of $z_1, z_2, \ldots, z_m$. We construct a hierarchical collections of intervals with a *fan-out* b, which can be viewed as a *perfect b-way tree*: each node corresponds to an interval, and has b children (except leaves), corresponding to b equally sized subintervals. We assume $m = b^h$ (if not, we can add some dummy values in D).

Level 0 in the hierarchy is $\mathcal{L}^0 = \{[z_1, z_m]\}$. $[z_1, z_m]$ corresponds to the root, and is recursively partitioned into b equally sized subintervals until we reach the leaves, *i.e.*, intervals with unit length $\mathcal{L}^h = \{[z_1, z_1], \ldots, [z_m, z_m]\}$. There are b^j intervals on *level* j, each covering m/b^j values:

$$\mathcal{L}^j = \{[z_{(i-1)\cdot m/b^j+1}, z_{i\cdot m/b^j}] \mid i = 1, 2, \ldots, b^j\}.$$

Let $\mathbf{I}_D = \{\mathcal{L}^0, \ldots, \mathcal{L}^h\}$ be the whole hierarchy ($h = \log_b m$).

Example 4.1. *Consider an ordinal dimension D_1 with 8 values. Figure 2 shows its hierarchical intervals (with $b = 2$).*

Query rewriting with HI. In a query $q = Q_T(\mathsf{SUM}(M), D \in [l, r])$, the interval $[l, r]$ can be decomposed into $2(b-1)\log_b m$ (or less) disjoint intervals, $I^1, \ldots, I^p$, in the hierarchy $\mathbf{I}_D$. q can be decomposed into sub-queries on these intervals. If every user tells the server whether her/his dimension value is in each interval in $\mathbf{I}_D$, in an LDP way, each sub-query $Q_T(\mathsf{SUM}(M), D \in I^i)$ can be estimated ($i = 1, 2, \ldots, p$). The query q can be answered by assembling estimates for the $p \le 2(b-1)\log_b m$ sub-queries, and thus with a polylogarithmic factor in the error. Following is a rewriting example.

Example 4.2. *Assume that the ordinal dimension D_1 in Table 1 takes values in {1, 2, …, 8}. Consider the query*

$$q_1 : \mathsf{SELECT}\ \mathsf{SUM}(M_1)\ \mathsf{FROM}\ T\ \mathsf{WHERE}\ D_1 \in [2, 7].$$

$[2, 7]$ is decomposed into 4 intervals (the blue ones in Figure 2); correspondingly, q_1 is rewritten as the sum of four queries with "$D_1 \in [2, 2]$", "$D_1 \in [3, 4]$", "$D_1 \in [5, 6]$", and "$D_1 \in [7, 7]$".

In a naive implementation of the above strategy, each user sends $\Theta(m)$ LDP reports (as there are $\Theta(m)$ intervals in $\mathbf{I}_D$). We will show that, in fact, $\Theta(\log m)$ LDP reports suffice.

The main idea is that the dimension value $t[D]$ of a user t belongs to exactly $h + 1$ intervals in $\mathbf{I}_D$, one on each level: suppose $t[D]$ belongs to $I \in \mathcal{L}^j$ on level j, we let $t[\mathcal{L}^j] = I$. We only need h frequency oracles, each encoding and collecting the interval $t[\mathcal{L}^j]$ on level j, for $j = 1, \ldots, h$.

In general, $[l, r]$ is partitioned into p disjoint intervals: $[l, r] = I^1 \cup \ldots \cup I^p$ – suppose I^i is on level $\mathcal{L}^{k_i}$. We rewrite

$$Q_T(\mathsf{SUM}(M), D \in [l, r]) = \sum_{i=1}^{p} Q_T(\mathsf{SUM}(M), D \in I^i) \quad (12)$$

where each sub-query $Q_T(\mathsf{SUM}(M), D \in I^i)$ can be estimated by the weighted frequency oracle on $\mathcal{L}^{k_i}$ as $\tilde{\mathbf{f}}_T^M(I^i)$.

Example 4.3. *In the hierarchy of D_1 in Figure 2, a tuple t_1 with $t_1[D_1] = 3$ belongs to one interval on each level (those crossed by the dashed arrowed line): $t_1[\mathcal{L}^3] = [3, 3]$, $t_1[\mathcal{L}^2] = [3, 4]$, $t_1[\mathcal{L}^1] = [1, 4]$, and $t_1[\mathcal{L}^0] = [1, 8]$. The first three intervals are encoded and collected using frequency oracles.*

As in Example 4.2, q_1 is decomposed into four sub-queries: the one with "$D_1 \in [3, 4]$" can be estimated with $\tilde{\mathbf{f}}_T^M([3, 4])$.

HI mechanism $(\mathcal{A}_{\mathsf{HI}}, \bar{\mathbf{P}}_{\mathsf{HI}})$**.** On the client side, the privacy budget is partitioned evenly for the h levels $\mathcal{L}^1, \ldots, \mathcal{L}^h$, and the interval a tuple t belongs to on each level (*i.e.*, $t[\mathcal{L}^i]$) is encoded using $\mathcal{A}_{\mathsf{FO}}$ in a weighted frequency oracle.

On the server, for a query q, we estimate each sub-query $Q_T(\mathsf{SUM}(M), D \in I^i)$ in (12) using the weighted frequency estimator $\tilde{\mathbf{f}}^M(I^i)$, and sum them up as an estimation to q.

$$\bar{\mathbf{P}}_{\mathsf{HI}}(q) = \sum_{i=1}^{p} \tilde{\mathbf{f}}_T^M(I^i). \quad (13)$$

Theorem 6 (1D-HI). *i) $\mathcal{A}_{\mathsf{HI}}$ satisfies ϵ-LDP. ii) $\bar{\mathbf{P}}_{\mathsf{HI}}(q)$ is an unbiased estimator of q, and the expected squared error*

$$\mathrm{Err}(\bar{\mathbf{P}}_{\mathsf{HI}}(q)) \le 2(b-1)\log_b m \cdot M_T^2 \cdot \frac{(e^{\epsilon/\log_b m} + 1)^2}{(e^{\epsilon/\log_b m} - 1)^2} \quad (14)$$

$$= \mathrm{O}\left(\frac{n\Delta^2 \log^3 m}{\epsilon^2}\right) \text{ when } \epsilon \text{ is small,}$$

where $n = |T|$ is the number of users, Δ is the range of M, $M_T^2 = \sum_{t \in T} t[M]^2$, and the constant b is the fan-out.

4.2 Better Accuracy via Level Partitioning

In $\mathcal{A}_{\mathsf{HI}}$, the privacy budget ϵ is partitioned evenly for the h levels. An alternative is to randomly partition the users into h groups instead of partitioning the privacy budget: users in a group S_j, corresponding to level j, can be regarded as a random sample with sampling rate $1/h$, and spend all the privacy budget on only level j. A bit surprisingly, this

Client side: Encode private dimension $t[D]$.

1: Randomly pick $j \in \{1, 2, \ldots, h\}$ with equal prob.
2: Suppose $t[D]$ is in interval $I \in \mathcal{L}^j$: $t[\mathcal{L}^j] \leftarrow I$;
3: Create LDP report:

$$\mathcal{A}_{\mathsf{HIO}}(t) \leftarrow \left(j, \mathcal{A}^{\epsilon}_{\mathsf{FO}}(t[\mathcal{L}^j])\right). \tag{16}$$

Server side: MDA query $q = Q_T(\mathsf{SUM}(M), D \in [l, r])$.

1: Let user groups $S_j \leftarrow \emptyset$ for $j = 1, \ldots, h$.
2: For each user t. if we get $\left(j, \mathcal{A}^{\epsilon}_{\mathsf{FO}}(t[\mathcal{L}^j])\right)$:
3: $\quad S_j \leftarrow S_j + \{t\}$;
4: Decompose $[l, r]$ into p disjoint intervals $I^1 \in \mathcal{L}^{k_1}$, $I^2 \in \mathcal{L}^{k_2}$, …, $I^p \in \mathcal{L}^{k_p}$ in the hierarchy $\mathbf{I}_D$.
5: For each $i = 1, 2, \ldots, p$:
6: $\quad$ Estimate $\mathbf{f}^M_T(I^i)$ as $\tilde{\mathbf{f}}^M_{T,1/h}(I^i)$ (using (9)):

$$\tilde{\mathbf{f}}^M_{T,1/h}(I^i) = h \cdot \bar{\mathbf{f}}^M_{S_{k_i}}(I^i); \tag{17}$$

7: Output an estimation to q as:

$$\bar{\mathbf{P}}_{\mathsf{HIO}}(q) = \sum_{i=1}^{p} \tilde{\mathbf{f}}^M_{T,1/h}(I^i). \tag{18}$$

Algorithm 1: 1D HI Optimized $(\mathcal{A}_{\mathsf{HIO}}, \bar{\mathbf{P}}_{\mathsf{HIO}})$

alternative has the accuracy boosted by orders of magnitude. The intuition behind this is: we gain accuracy by spending more privacy budget on each level, but lose accuracy as each level is supported for a random sample of users (refer to Proposition 5 in Section 3.3); as long as the accuracy gain overcomes the loss, the overall accuracy can be boosted.

Query $q = Q_T(\mathsf{SUM}(M), D \in [l, r])$ is decomposed in the same way as (12). For a sub-query $Q_T(\mathsf{SUM}(M), D \in I^i)$ in (12), where I^i is on level k_i, we refer to the user group S_{k_i} that corresponds to level k_i. As introduced in Section 3.3, we can run frequency oracles on the random sample S_{k_i}, and scale up the estimation $\bar{\mathbf{f}}^M_{S_{k_i}}(I^i)$ by a factor of h, to approximate the sub-query's answer. Based on the above idea, we propose a mechanism $(\mathcal{A}_{\mathsf{HIO}}, \bar{\mathbf{P}}_{\mathsf{HIO}})$ with details described in Algorithm 1. Its error bound is given in Theorem 7.

Theorem 7 (1D-HIO). *i)* $\mathcal{A}_{\mathsf{HIO}}$ *satisfies* ϵ*-LDP. ii)* $\bar{\mathbf{P}}_{\mathsf{HIO}}(q)$ *is an unbiased estimator of* q *with expected squared error*

$$\mathrm{Err}(\bar{\mathbf{P}}_{\mathsf{HI}}(q)) \le 4(b-1)\log_b^2 m \cdot M_T^2 \cdot \frac{(e^{2\epsilon}+1)}{(e^{\epsilon}-1)^2} \tag{15}$$

$$= \mathrm{O}\left(\frac{n\Delta^2 \log^2 m}{\epsilon^2}\right) \text{ when } \epsilon \text{ is small,}$$

where $n = |T|$, Δ *is the range of* M, *and* $M_T^2 = \sum_{t \in T} t[M]^2$.

The HIO mechanism boosts the accuracy by a factor of $\log_b m$ in comparison to the HI mechanism (14). We use $b = 5$ in our implementation to minimize RHS of (15).

5 MULTIPLE PRIVATE DIMENSIONS

We now introduce how to handle multiple private dimensions in MDA queries. We will first focus on the case when we have multiple ordinal dimensions and range constraints in an MDA query, for which we extend our HI/HIO mechanism in Section 4 to a multi-dimensional one. We will then introduce how to handle a combination of ordinal and categorical dimensions. Finally, when there are many sensitive dimensions in the data model and the worst-case error blows up, we will introduce a *split-and-conjunction* mechanism which is designed to handle low-dimensional queries.

5.1 Multiple Ordinal Dimensions

In order to extend our HI mechanism for multiple dimensions, let's first introduce a multi-dimensional hierarchy of intervals, which naturally generalizes the one-dimensional hierarchy in Section 4.1. An MDA query can be decomposed into a polylogarithmic (in dimension cardinalities) number of sub-queries in this hierarchy, and they together are aggregated to answer the original MDA query without blowing up the worst-case error. Similar user-partitioning techniques as in Section 4.2 can be applied to boost the accuracy.

5.1.1 Multi-dimensional Hierarchical Intervals. Recall that $\mathbf{I}_D = \{\mathcal{L}^0_D, \ldots, \mathcal{L}^h_D\}$ is the hierarchy for dimension D: level 0 is $\mathcal{L}^0_D = \{[z_1, z_m]\}$ and $\mathcal{L}^{j+1}_D$ is obtained by partitioning each interval in $\mathcal{L}^j_D$ into b equally sized subintervals. W.l.o.g., assume each dimension has the same cardinality $m = b^h$ (*i.e.*, # distinct values) for the simplicity of explanation.

Two-dimensional hierarchy. Let's first focus on two dimensions. Define a 2-*dim hierarchy* to be:

$$\mathbf{I}_{D_1} \otimes \mathbf{I}_{D_2} = \left\{ \mathcal{L}^{j_1}_{D_1} \times \mathcal{L}^{j_2}_{D_2} \;\middle|\; 0 \le j_1, j_2 \le h \right\}.$$

Each $\mathcal{L}^{j_1}_{D_1} \times \mathcal{L}^{j_2}_{D_2}$ is called a 2-*dim level*. There are a total of $(h+1)^2$ 2-dim levels in a 2-dim hierarchy. Each pair $\langle I_1, I_2 \rangle \in \mathcal{L}^{j_1}_{D_1} \times \mathcal{L}^{j_2}_{D_2}$ is called a 2-*dim interval*. We will write $I_1 I_2 \ldots I_d$ as a shorthand for $\langle I_1, I_2, \ldots, I_d \rangle$ in the rest part.

Consider a tuple t, for each j_1 and j_2, $t[D_1]$ and $t[D_2]$ belong to exactly one interval $I_1^{j_1} \in \mathcal{L}^{j_1}_{D_1}$ and one $I_2^{j_2} \in \mathcal{L}^{j_2}_{D_2}$, respectively. Conceptually, we *augment* t with *a new dimension* "$\mathcal{L}^{j_1}_{D_1} \times \mathcal{L}^{j_2}_{D_2}$" for the corresponding 2-dim level: let

$$t[\mathcal{L}^{j_1}_{D_1} \times \mathcal{L}^{j_2}_{D_2}] = I_1^{j_1} I_2^{j_2},$$

which means that $t[D_1] \in I_1^{j_1} \wedge t[D_2] \in I_2^{j_2}$. Indeed, we have

$$\begin{aligned} &Q_T(\mathsf{SUM}(M), D_1 \in I_1^{j_1} \wedge D_2 \in I_2^{j_2}) \\ =&Q_T(\mathsf{SUM}(M), \mathcal{L}^{j_1}_{D_1} \times \mathcal{L}^{j_2}_{D_2} = I_1^{j_1} I_2^{j_2}) = \mathbf{f}^M_T(I_1^{j_1} I_2^{j_2}). \end{aligned}$$

In an MDA query $q = Q_T(\mathsf{SUM}(M), D_1 \in [l_1, r_1] \wedge D_2 \in [l_2, r_2])$, $[l_1, r_1]$ can be decomposed into p_1 disjoint intervals in

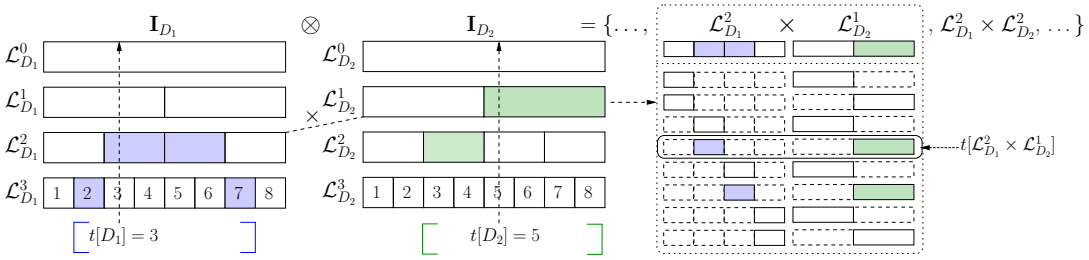


Figure 3: 2D hierarchy of intervals, query decomposition, and HI mechanism

$\mathbf{I}_{D_1}$: $[l_1, r_1] = I_1^1 \cup \ldots \cup I_1^{p_1}$, and similarly $[l_2, r_2] = I_2^1 \cup \ldots \cup I_2^{p_2}$. We can then decompose q into $p_1 \times p_2$ sub-queries:

$$\begin{aligned} & Q_T(\mathsf{SUM}(M), D_1 \in [l_1, r_1] \wedge D_2 \in [l_2, r_2]) \\ = & \sum_{1 \le a \le p_1, 1 \le b \le p_2} Q_T(\mathsf{SUM}(M), D_1 \in I_1^a \wedge D_2 \in I_2^b) \\ = & \sum_{1 \le a \le p_1, 1 \le b \le p_2} \mathbf{f}_T^M(I_1^a I_2^b). \end{aligned} \tag{19}$$

As $p_1, p_2 \le 2(b-1)\log_b m$, there are $O(\log^2 m)$ sub-queries. Each sub-query can be estimated as $\bar{\mathbf{f}}_T^M(I_1^a I_2^b)$ using a weighted frequency oracle on the corresponding 2-dim level, and then we just need to sum up these estimates to answer q.

Example 5.1. *Suppose the two ordinal dimensions D_1 and D_2 take values in $\{1, 2, \ldots, 8\}$. A 2-dim hierarchy on them is shown in Figure 3. Their individual 1-dim hierarchies $\mathbf{I}_{D_1}$ and $\mathbf{I}_{D_2}$ are on the left, each with 4 levels. The 2-dim hierarchy is a Cartesian product of the two, with 4×4 2-dim levels. In particular, the 2-dim level, $\mathcal{L}^2_{D_1} \times \mathcal{L}^1_{D_2}$, depicted on the right, is a Cartesian product of two 1-dim interval sets $\mathcal{L}^2_{D_1}$ and $\mathcal{L}^1_{D_2}$, with 4×2 2-dim intervals, each of which is a pair of 1-dim intervals, with one from $\mathcal{L}^2_{D_1}$ and the other from $\mathcal{L}^1_{D_2}$. For example, the 4th one (top-to-bottom) in the figure is $[3, 4][5, 8]$.*

Consider a tuple t with $t[D_1] = 3$ and $t[D_2] = 5$. It belongs to the above 2-dim interval as $t[D_1] \in [3, 4]$ and $t[D_2] \in [5, 8]$. Thus, the augmented dimension $t[\mathcal{L}^2_{D_1} \times \mathcal{L}^1_{D_2}] = [3, 4][5, 8]$.

Consider the following MDA query:

$$q_2 : \text{SELECT SUM}(M_1) \text{ FROM } T$$
$$\text{WHERE } D_1 \in [2, 7] \text{ AND } D_2 \in [3, 8]$$

As shown in Figure 3, $[2, 7]$ can be partitioned into 4 intervals in $\mathbf{I}_{D_1}$ (blue ones), and $[3, 8]$ partitioned into 2 in $\mathbf{I}_{D_2}$ (green ones). Thus, q_2 can be decomposed into 4×2 disjoint sub-queries that are to be answered using weighted frequency oracles – two are on the 2-dim level $\mathcal{L}^2_{D_1} \times \mathcal{L}^1_{D_2}$: one with "$D_1 \in [3, 4]$ AND $D_2 \in [5, 8]$" and one with "$D_1 \in [5, 6]$ AND $D_2 \in [5, 8]$".

d-dim hierarchy. The construction of a 2-dim hierarchy can be easily extended for more dimensions. Define:

$$\mathbf{I}_{D_1} \otimes \ldots \otimes \mathbf{I}_{D_d} = \left\{ \mathcal{L}^{j_1}_{D_1} \times \ldots \times \mathcal{L}^{j_d}_{D_d} \;\middle|\; 0 \le j_1, j_2, \ldots, j_d \le h \right\}$$

to be a *hierarchy* of *d-dim levels* (there are $(h+1)^d$ levels). Each $I_1 I_2 \ldots I_d \in \mathcal{L}^{j_1}_{D_1} \times \ldots \times \mathcal{L}^{j_d}_{D_d}$ is a *d-dim interval*.

An MDA query $q = Q_T(\mathsf{SUM}(M), D_1 \in [l_1, r_1] \wedge \ldots \wedge D_d \in [l_d, r_d])$ can be thus decomposed into $p_1 \times \ldots \times p_d$ sub-queries:

$$\begin{aligned} & Q_T(\mathsf{SUM}(M), D_1 \in [l_1, r_1] \wedge \ldots \wedge D_d \in [l_d, r_d]) \\ = & \sum_{1 \le i_1 \le p_1, \ldots, 1 \le i_d \le p_d} \mathbf{f}_T^M(I_1^{i_1} I_2^{i_2} \ldots I_d^{i_d}), \end{aligned} \tag{20}$$

where $p_1, p_2, \ldots, p_d \le 2(b-1)\log_b m$.

5.1.2 Multi-dimensional HI Mechanism ($\mathcal{A}_{\mathsf{HI}}, \bar{\mathbf{P}}_{\mathsf{HI}}$). On the client side, an *augmented dimension* tells which interval a user belongs to in each d-dim level. We use $\mathcal{A}_{\mathsf{FO}}$ in a weighted frequency oracle to encode the $(h+1)^d$ augmented dimensions, each of which uses a privacy budget of $\epsilon/(h+1)^d$. On the server side, from how q is rewritten in (20), we can estimate the weighted frequency of each d-dim interval and sum up the estimates to approximate the answer to q:

$$\bar{\mathbf{P}}_{\mathsf{HI}}(q) = \sum_{1 \le i_1 \le p_1, \ldots, 1 \le i_d \le p_d} \bar{\mathbf{f}}_T^M(I_1^{i_1} I_2^{i_2} \ldots I_d^{i_d}). \tag{21}$$

It is formally described in Appendix B (Algorithm 4).

Theorem 8 (HI). *i) $\mathcal{A}_{\mathsf{HI}}$ satisfies ϵ-LDP. ii) $\bar{\mathbf{P}}_{\mathsf{HI}}(q)$ is an unbiased estimator of q with expected squared error*

$$\begin{aligned} \mathrm{Err}(\bar{\mathbf{P}}_{\mathsf{HI}}(q)) & \le (2(b-1)\log_b m)^{d_q} M_T^2 \cdot \frac{(e^{\epsilon/(\log_b m+1)^d} + 1)^2}{(e^{\epsilon/(\log_b m+1)^d} - 1)^2} \\ & = O\left(\frac{n\Delta^2 \log^{d_q+2d} m}{\epsilon^2} \right) \text{ when } \epsilon \text{ is small,} \end{aligned} \tag{22}$$

where $n = |T|$ is the number of users, d (d_q) is the number of sensitive dimensions (in the query q), Δ is the range of M, $M_T^2 = \sum_{t \in T} t[M]^2$, and the constant b is the fan-out.

5.1.3 Boosting Accuracy via User Partitioning. Similar to the 1-dim case in Section 4.2, HI's accuracy can be boosted by randomly partitioning users by levels. On the client, a user picks one of the $(h+1)^d$ d-dim levels randomly, and encodes only the d-dim interval in this level with privacy budget ϵ. On the server, we estimate the weighted frequency $\mathbf{f}_T^M(I_1^{i_1} I_2^{i_2} \ldots I_d^{i_d})$ in (20) with LDP reports in the corresponding level from a random $1/(h+1)^d$ portion of users (as in Section 3.3).

Client side: Encode dimensions $t[D_1], \ldots, t[D_d]$.

1: Randomly pick $(j_1, \ldots, j_d) \in \{0, 1, \ldots, h\}^d$.
2: Suppose $t[D_i]$ is in interval $I_i \in \mathcal{L}_{D_i}^{j_i}$ $(i = 1, \ldots, d)$:
let $t[\mathcal{L}_{D_1}^{j_1} \times \ldots \times \mathcal{L}_{D_d}^{j_d}] \leftarrow I_1 I_2 \ldots I_d$.
3: Create LDP report:

$$\mathcal{A}_{\mathsf{HIO}}(t) \leftarrow \left((j_1, \ldots, j_d), \mathcal{A}_{\mathsf{FO}}^{\epsilon}(t[\mathcal{L}_{D_1}^{j_1} \times \ldots \times \mathcal{L}_{D_d}^{j_d}])\right). \quad (23)$$

Server side: MDA query $q = Q_T(\mathsf{SUM}(M), D_1 \in [l_1, r_1] \wedge \ldots \wedge D_d \in [l_d, r_d])$.

1: Let $S_{(j_1,\ldots,j_d)} \leftarrow \emptyset$ for each $(j_1, \ldots, j_d) \in \{0, 1, \ldots, h\}^d$.
2: For each user t, if we get $((j_1, \ldots, j_d), \mathcal{A}_{\mathsf{FO}}^{\epsilon}(\cdot))$:
3: $S_{(j_1,\ldots,j_d)} \leftarrow S_{(j_1,\ldots,j_d)} + \{t\}$;
4: For $i = 1$ to d do:
5: Decompose $[l_i, r_i]$ into p_i disjoint intervals $[l_i, r_i] \rightarrow I_i^1 \cup I_i^2 \cup \ldots \cup I_i^{p_i}$ in the hierarchy $\mathbf{I}_{D_i}$;
6: For each $(i_1, \ldots, i_d) \in \{1, \ldots, p_1\} \times \ldots \times \{1, \ldots, p_d\}$:
7: Estimate $\mathbf{f}_T^M(I_1^{i_1} I_2^{i_2} \ldots I_d^{i_d})$ as (suppose $I_k^{i_k} \in \mathcal{L}_{D_k}^{j_k}$):

$$\tilde{\mathbf{f}}_{T,1/(h+1)^d}^M(I_1^{i_1} I_2^{i_2} \ldots I_d^{i_d}) = (h+1)^d \cdot \tilde{\mathbf{f}}_{S_{(j_1,\ldots,j_d)}}^M(I_1^{i_1} I_2^{i_2} \ldots I_d^{i_d}); \quad (24)$$

8: Output an estimation to q as:

$$\bar{\mathbf{P}}_{\mathsf{HIO}}(q) = \sum_{1 \le i_1 \le p_1, \ldots, 1 \le i_d \le p_d} \tilde{\mathbf{f}}_{T,1/(h+1)^d}^M(I_1^{i_1} I_2^{i_2} \ldots I_d^{i_d}). \quad (25)$$

Algorithm 2: d-dim HI Optimized $(\mathcal{A}_{\mathsf{HIO}}, \bar{\mathbf{P}}_{\mathsf{HIO}})$

The resulting mechanism $(\mathcal{A}_{\mathsf{HIO}}, \bar{\mathbf{P}}_{\mathsf{HIO}})$ is in Algorithm 2. Theorem 9 shows that the gain from a larger privacy budget spent on the picked level per user overcomes the error due to running weighted frequency oracles on samples. $\bar{\mathbf{P}}_{\mathsf{HIO}}$ has a significant accuracy boost over $\bar{\mathbf{P}}_{\mathsf{HI}}$.

Theorem 9 (HIO). *i)* $\mathcal{A}_{\mathsf{HIO}}$ *satisfies* ϵ*-LDP. ii)* $\bar{\mathbf{P}}_{\mathsf{HIO}}(q)$ *is an unbiased estimator of* q *with expected squared error*

$$\begin{aligned} &\mathrm{Err}(\bar{\mathbf{P}}_{\mathsf{HIO}}(q)) \quad (26)\\ \le &(2(b-1)(\log_b m + 1))^{d_q} (\log_b m + 1)^d M_T^2 \cdot \frac{(e^{2\epsilon}+1)}{(e^{\epsilon}-1)^2}\\ = &\mathrm{O}\left(\frac{n\Delta^2 \log^{d_q+d} m}{\epsilon^2}\right) \text{ when } \epsilon \text{ is small,} \end{aligned}$$

where $n = |T|$ *is the number of users,* d (d_q) *is the number of sensitive dimensions (in the query* q*),* Δ *is the range of* M*,* $M_T^2 = \sum_{t \in T} t[M]^2$*, and the constant* b *is the fan-out.*

5.2 Ordinal and Categorical Dimensions

A categorical dimension D can be regarded as a hierarchy with two levels: $\mathcal{L}_D^0 = \{*\}$ and $\mathcal{L}_D^1 = \{[v_1], [v_2], \ldots, [v_c]\}$, where '$*$' means 'anything', and $v_1, v_2, \ldots, v_c$ are distinct values D. As there are only point constraints on D, *e.g.*, "$D = v_i$", all the intermediate levels are unnecessary. Such a *categorical hierarchy* can be incorporated into the multi-dimensional hierarchy of intervals introduced in Section 5.1.1. Refer to Appendix C for how our HIO mechanism is extended.

5.3 Split-and-Conjunction: When the Dimensionality is High

HI and HIO mechanisms have errors exponentially depending on both the number of dimensions in the query (d_q) and the total number of sensitive dimensions in the data model (d). On the client side, they partition privacy budget ϵ or users into $\Theta(\log^d m)$ portions, one for each d-dim level, and thus may introduce too much LDP noise for large d. When $d_q \ll d$, there is room for improvement: whether it is possible to remove the exponential dependency on d.

We introduce our *split-and-conjunction (SC) mechanism* in this section. Instead of encoding a tuple on the d-dim hierarchy, a user maintains d one-dim hierarchies, on which the d dimensions are encoded and reported independently. The privacy budget ϵ is thus partitioned into $\Theta(d \log m)$ portions (d dimensions each with $\Theta(\log m)$ one-dim levels). The question is, while all dimensions are reported independently, whether we can estimate how many rows have, *e.g.*, $t[D_1] = v_1$ *and* $t[D_2] = v_2$ conjunctively. To this end, we will first introduce a new class of estimators in frequency oracles, called *conjunctive estimators* as a building block of SC.

5.3.1 Conjunctive Estimators $\hat{\mathbf{f}}$ *and* $\hat{\mathbf{f}}^M$*.* Let D_1 and D_2 be two sensitive dimensions. A *conjunctive weighted frequency query* asks, for a set of users S, and $v_1 \in D_1$ and $v_2 \in D_2$, the total measure of users with $t[D_1] = v_1$ and $t[D_2] = v_2$, *i.e.*, weighted frequency $\mathbf{f}_S^M(v_1 v_2) = \sum_{t \in S \wedge t[D_1] = v_1 \wedge t[D_2] = v_2} t[M]$ of $\langle v_1, v_2 \rangle$ on the domain $D_1 \times D_2$, or unweighted frequency $\mathbf{f}_S(v_1 v_2) = |\{t \in S \mid t[D_1] = v_1 \wedge t[D_2] = v_2\}|$.

Given two sets of independently-generated LDP reports $\{\mathcal{A}_{\mathsf{FO}}(t[D_1])\}_{t \in S}$ and $\{\mathcal{A}_{\mathsf{FO}}(t[D_2])\}_{t \in S}$, we want to estimate $\mathbf{f}_S^M(v_1 v_2)$, for any $v_1 \in D_1$ and $v_2 \in D_2$. More generally, we can estimate $\mathbf{f}_S^M(v_1 \ldots v_k)$ from reports on k dimensions. For this purpose, we introduce *conjunctive estimators* $\hat{\mathbf{f}}$ and $\hat{\mathbf{f}}^M$.

States and transition. Recall that an LDP report $\mathcal{A}_{\mathsf{FO}}(t[D_i]) = \langle H, y \rangle$ is a pair of a random hash function H and a randomly perturbed value y (refer to Section 3.2.1 and Appendix A).

We define two indicator variables for the query term v_i:

$$\textit{input state: } B_i(t) = \begin{cases} 0, & \text{if } t[D_i] \neq v_i \\ 1, & \text{if } t[D_i] = v_i \end{cases} \quad \text{and} \quad (27)$$

$$\textit{output state: } A_i(t) = \begin{cases} 0, & \text{if } H(v_i) \neq y \\ 1, & \text{if } H(v_i) = y \end{cases} \quad (\text{for } i = 1, 2). \quad (28)$$

Input state $B_i(t)$ is deterministic and depends on v_i; output state $A_i(t)$ is a random variable and depends on both v_i and randomness in $\mathcal{A}_{\mathsf{FO}}$. We can define the following *transition probabilities* (with their values calculated in Appendix A):

$$P_{b\to a} \triangleq \mathbf{Pr}\left[A_i(t) = a \mid B_i(t) = b\right] \quad (\text{for } a, b \in \{0, 1\}).$$

A tuple with two dimensions has 4 possible 2-*dim input states*: "11" (meaning $B_1(t) = 1 \wedge B_2(t) = 1$), "01", "10", and "00". After applying $\mathcal{A}_{\mathsf{FO}}$, the LDP report has 4 possible 2-*dim output states* "11" (meaning $A_1(t) = 1 \wedge A_2(t) = 1$), "01", "10", and "00". We can derive 2-*dim transition probabilities*:

$$\begin{aligned} &P_{b_1b_2\to a_1a_2} \quad (\text{for } a_1, a_2, b_1, b_2 \in \{0, 1\}) \\ &\triangleq \mathbf{Pr}\left[A_1(t) = a_1 \wedge A_2(t) = a_2 \mid B_1(t) = b_1 \wedge B_2(t) = b_2\right] \\ &= P_{b_1\to a_1} \cdot P_{b_2\to a_2} \text{ (as } A_1(t) \text{ and } A_2(t) \text{ are independent)}, \end{aligned}$$

since dimensions are encoded independently.

Estimation via transition matrix. For a set S of users and their LDP reports $\{\mathcal{A}_{\mathsf{FO}}(t[D_1])\}_{t\in S}$ and $\{\mathcal{A}_{\mathsf{FO}}(t[D_2])\}_{t\in S}$, we observe the frequency of each 2-dim output state. With them, we can estimate the frequencies of input states via transition probabilities, and in particular, the frequency of 2-dim input state "11" is equal to unweighted frequency $\mathbf{f}_S(v_1v_2)$.

The frequencies of input states are, for $b_1b_2 = 11, 01, 10, 00$:

$$\mathbf{b}_S(b_1b_2) = |\{t \in S \mid B_1(t) = b_1 \wedge B_2(t) = b_2\}|.$$

And those of output states are, for $a_1a_2 = 11, 01, 10, 00$:

$$\mathbf{a}_S(a_1a_2) = |\{t \in S \mid A_1(t) = a_1 \wedge A_2(t) = a_2\}|.$$

Consider the corresponding frequency vectors of input and output states, $\mathbf{b} = [\mathbf{b}_S(11), \mathbf{b}_S(01), \mathbf{b}_S(10), \mathbf{b}_S(00)]^\top$ and $\mathbf{a} = [\mathbf{a}_S(11), \mathbf{a}_S(01), \mathbf{a}_S(10), \mathbf{a}_S(00)]^\top$. We can establish the relationship between $\mathbf{b}$ and $\mathbf{a}$ through transition matrix $\mathbf{P}$:

$$\mathbf{P}\cdot\mathbf{b} \triangleq \begin{pmatrix} P_{11\to11} & P_{01\to11} & P_{10\to11} & P_{00\to11} \\ P_{11\to01} & P_{01\to01} & P_{10\to01} & P_{00\to01} \\ P_{11\to10} & P_{01\to10} & P_{10\to10} & P_{00\to10} \\ P_{11\to00} & P_{01\to00} & P_{10\to00} & P_{00\to00} \end{pmatrix} \begin{pmatrix} \mathbf{b}_S(11) \\ \mathbf{b}_S(01) \\ \mathbf{b}_S(10) \\ \mathbf{b}_S(00) \end{pmatrix} = \mathbf{E}[\mathbf{a}],$$

from the property of $\mathbf{P}$ and the linearity of expectation. By observing the frequency vector $\mathbf{a}$, we can estimate $\mathbf{b}$ as:

$$\hat{\mathbf{b}} = [\hat{\mathbf{b}}_S(11), \hat{\mathbf{b}}_S(01), \hat{\mathbf{b}}_S(10), \hat{\mathbf{b}}_S(00)]^\top = \mathbf{P}^{-1} \cdot \mathbf{a} \tag{29}$$

$$\text{and, in particular,} \quad \hat{\mathbf{f}}_S(v_1v_2) = \hat{\mathbf{b}}_S(11). \tag{30}$$

From the linearity of expectation, we have $\mathbf{E}[\hat{\mathbf{b}}] = \mathbf{P}^{-1}\cdot\mathbf{E}[\mathbf{a}] = \mathbf{P}^{-1}\mathbf{P}\cdot\mathbf{b} = \mathbf{b}$, and thus $\mathbf{E}[\hat{\mathbf{f}}_S(v_1v_2)] = \mathbf{b}_S(11) = \mathbf{f}_S(v_1v_2)$.

Similar to (8), for the weighted case, we can derive

$$\hat{\mathbf{f}}^M_S(v_1v_2) = \sum_{\text{distinct } x} x \cdot \hat{\mathbf{f}}_{S_x}(v_1v_2), \tag{31}$$

where $S_x = \{t \in S \mid t[M] = x\}$. Its unbiasedness is from $\hat{\mathbf{f}}$'s.

In order to extend $\hat{\mathbf{f}}$ and $\hat{\mathbf{f}}^M$ from two dimensions to d dimensions, we extend 2-dim input/output states to d-dim input/output states. Correspondingly, their frequencies are:

$$\begin{aligned} &\mathbf{b}_S(b_1 \dots b_d) = |\{t \in S \mid B_1(t) = b_1 \wedge \dots \wedge B_d(t) = b_d\}|, \\ &\mathbf{a}_S(a_1 \dots a_d) = |\{t \in S \mid A_1(t) = a_1 \wedge \dots \wedge A_d(t) = a_d\}|. \end{aligned}$$

The transition matrix $\mathbf{P}$ is a $2^d \times 2^d$ one, as there are 2^d states. The errors in $\hat{\mathbf{f}}$ and $\hat{\mathbf{f}}^M$ can be bounded as follows.

Proposition 10 (Conjunction of Oracles). *Run an ϵ-LDP encoder $\mathcal{A}^\epsilon_{\mathsf{FO}}$ on each of the d dimensions independently (overall, the procedure is $(d\epsilon)$-LDP). For any k dimensions and values $v_1, \dots, v_k$ on them, we have unbiased estimators $\hat{\mathbf{f}}_S$ and $\hat{\mathbf{f}}^M_S$ of conjunctive unweighted and weighted frequencies, respectively,*

$$\begin{aligned} \textit{with error} \quad &\mathrm{Err}(\hat{\mathbf{f}}_S(v_1 \dots v_k)) = \mathrm{O}\Big(|S|/\epsilon^{2k}\Big) \ \textit{and} \\ &\mathrm{Err}(\hat{\mathbf{f}}^M_S(v_1 \dots v_k)) = \mathrm{O}\Big(|S|\Delta^2/\epsilon^{2k}\Big), \end{aligned}$$

where $\Delta = \max(M) - \min(M)$. If we guarantee ϵ-LDP across all the d dimensions, each dimension gets a privacy budget ϵ/d, and thus, the error of $\hat{\mathbf{f}}^M_S$ is $\mathrm{O}\big(|S|\Delta^2/(\epsilon/d)^{2k}\big)$.

5.3.2 Split-and-Conjunction (SC) Mechanism. We describe our SC mechanism $(\mathcal{A}_{\mathsf{SC}}, \bar{\mathbf{P}}_{\mathsf{SC}})$ formally in Algorithm 5 in Appendix D. On the client side, a user reports each one-dim interval s/he belongs to (one per level) in each one-dim hierarchy $\mathbf{I}_{D_i}$ using $\mathcal{A}_{\mathsf{FO}}$ independently, with a privacy budget of $\epsilon/(dh)$. The estimator on the server side is the same as the one in HI mechanism, except that, instead of $\tilde{\mathbf{f}}^M$, the conjunctive estimator $\hat{\mathbf{f}}^M$ (from Section 5.3.1) is used as we have no access to the (LDP version of) d-dim hierarchy. Error in the estimated answer to an MDA query is exponentially dependent on only d_q (number of private dimensions in q).

Theorem 11 (SC). *i) $\mathcal{A}_{\mathsf{SC}}$ satisfies ϵ-LDP. ii) $\bar{\mathbf{P}}_{\mathsf{SC}}(q)$ is an unbiased estimator of q with expected squared error*

$$\mathrm{Err}(\bar{\mathbf{P}}_{\mathsf{SC}}(q)) = \mathrm{O}\left(\frac{n\Delta^2 d^{2d_q}\log^{3d_q} m}{\epsilon^{2d_q}}\right) \ \textit{when } \epsilon \textit{ is small}, \tag{32}$$

where $n = |T|$ is the number of users, d (d_q) is the number of sensitive dimensions (in the query q), and Δ is the range of M.

5.4 Performance Comparison

The accuracy of mechanisms introduced depends on some parameters, *e.g.*, number/sizes of dimensions, in different ways. We can identify the analytical turning points of their performance, which will be verified experimentally later.

Marginal/FO v.s. HIO. Worst-case errors in the marginal or FO-based solution (introduced in Section 3.4) depend on ϵ, $|T|$, and Δ in the same way as errors in HIO asymptotically.

Consider the marginal with all the dimensions in the predicate. Define the *volume* $\mathrm{vol}(q)$ of a query q to be the ratio of

marginal rows satisfying its predicate to all marginal rows. A 1-dim query with a range constraint $D \in [l, r]$ has volume $\text{vol}(q) = (r-l+1)/m$, From (11) in Section 3.4 and Theorem 7, HIO is better than the marginal/FO-based solution if

$$(r - l + 1) \geq \Theta(\log^2 m) \Leftrightarrow \text{vol}(q) \geq \Theta(\log^2 m/m). \quad (33)$$

If there are d sensitive dimensions, from Section 3.4 and Theorem 9, HIO is better than the marginal-based solution if

$$\text{vol}(q) \geq \Theta(\log^{2d} m/m^d) \quad (\text{when } d_q = d). \quad (34)$$

HIO v.s. SC. Comparing Theorem 11 to Theorem 9, SC removes d from the power of $\log m$ in the error, but incurs an additional term d^{2d_q}. Only when (from Theorems 9 and 11)

$$(d \log m/\epsilon)^{2d_q} \leq \Theta(\log^d m/\epsilon^2), \quad (35)$$

i.e., d_q is small enough relative to d, SC is better than HIO.

Please refer to Appendix F for a comparison of comlexity of different mechanisms we have introduced so far.

6 EVALUATION

We evaluate our mechanisms in various settings. HIO performs the best most of time, with a normalized absolute error less than 5% in most queries, and a relative error less than 5% if the predicate is not too selective; SC performs better in high-dimensional settings if the number of dimensions in the query is much less than the total number of dimensions. We also conduct a case study in an e-commerce application. Mechanisms are implemented in Python and evaluated on an Intel Xeon E5 2682 v4 PC with 64GB memory.

Datasets. We conduct experiments on three datasets:
- *Adult [11]:* A dataset from the UCI ML repository with around 45 thousand tuples after removing missing values.
- *IPUMS [31]:* A US census dataset from the IPUMS repository. It contains around 3 million records.
- A real dataset with 150 million records from an e-commerce application. Details are deferred to Section 6.2.3.

Experiment settings. We compare the four mechanisms:
- *MG:* Processing MDA queries with the state-of-the-art LDP marginal-releasing technique [41] (described in Section 3.4).
- *HI:* Hierarchical-interval mechanism ($\mathcal{A}_{\text{HI}}, \bar{\mathbf{P}}_{\text{HI}}$).
- *HIO:* HI Optimized ($\mathcal{A}_{\text{HIO}}, \bar{\mathbf{P}}_{\text{HIO}}$).
- *SC:* Split-and-conjunction (SC) mechanism ($\mathcal{A}_{\text{SC}}, \bar{\mathbf{P}}_{\text{SC}}$).

We use fan-out $b = 5$ for HI, HIO, and SC.
We test SUM/COUNT/AVG queries (SUM by default).

Error measures. We use two error measures:
- *Mean Normalized Absolute Error* $\text{MNAE} = \text{AVG}_q(\frac{|\bar{\mathbf{P}}(q)-q|}{\Sigma_S})$, where the absolute error is normalized by $\Sigma_S = \sum_t |t[M]|$ to $[0, 1]$. It is used for SUM queries, and measures how large errors are relative to the maximum possible answer (Σ_S).
- *Mean Relative Error* $\text{MRE} = \text{AVG}_q(\frac{|\bar{\mathbf{P}}(q)-q|}{|\bar{\mathbf{P}}(q)|})$. It is used for SUM/COUNT/AVG queries, and measures how large the error is relative to the true answer for each query.

In Section 6.1, we use MNAE for SUM queries to verify theoretical results and compare the mechanisms, as they all have theoretical guarantees about absolute errors, which are independent of (or not dominated by) query sizes/answers.

In Section 6.2, we use relative error MRE to demonstrate the utility of our mechanisms for SUM/COUNT/AVG queries. Queries are partitioned into groups by selectivity of their predicates: in each group, queries have similar sizes and answers, and thus MRE's for different queries are comparable.

For each data point in every figure, we test 30 random queries and plot their MNAE or MRE (Y-axis) with 1-std.

6.1 Experimental Comparison

We compare different mechanisms for varying factors that potentially influence the accuracy: i) query volume $\text{vol}(q)$ (defined in Section 5.4), ii) number of dimensions, iii) domain sizes (cardinalities of dimensions), iv) data size, and v) ϵ.

6.1.1 One Ordinal Dimension. We start from MDA queries with one sensitive ordinal dimension: $q = Q_T(\text{SUM}(M), D \in [l, r])$. We create the ordinal dimension with size $m = 1024$ by bucketizing a numeric column of the table.

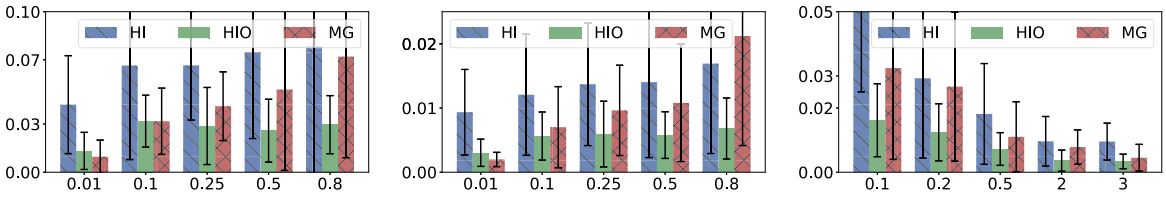
Varying query volume. Figures 4(a)-4(b) show the MNAE for different query volumes on Adult and IPUMS (1M sample). The interval $[l, r]$ is generated randomly with $r - l + 1 = m \cdot \text{vol}(q)$. The accuracy of MG deteriorates quickly as $\text{vol}(q)$ increases. Only when $\text{vol}(q)$ is as small as 0.01, MG is better than HIO (the errors of both are small, too); $\text{vol}(q) = 0.1$ is the break-even point, which conforms with our analysis in Section 5.4; when $\text{vol}(q) = 0.8$, error of MG is $\sim 3\times$ that of HIO. The performance of different methods on IPUMS is better than that on Adult, because the data size is larger.

Varying data size. We sample 0.1, 0.2, 0.5, 2 and 3 million rows from IPUMS (without replacement), and test queries with volume 0.25. As can be seen in Figure 4(c), the larger the dataset, the better the estimation accuracy, which is consistent with our theorems. HIO always performs best.

Varying privacy budget ϵ. Figure 5 shows performance of different mechanisms on IPUMS for varying ϵ. All methods benefit from larger ϵ, with HIO performing the best.

In the rest of Section 6.1, we will use $\text{vol}(q) = 0.25$, data size $|T| = 1$ million, and $\epsilon = 2$, as their default values.

6.1.2 Two Ordinal Dimensions. We create two ordinal dimensions on IPUMS with sizes m_1 and m_2 by bucketizing numeric columns. Two configurations of $m_1 \times m_2$ are tested:



(a) Adult: vary vol(q) (b) IPUMS 1M: vary vol(q) (c) IPUMS: vary $|T|$ (million)

Figure 4: Comparing different mechanisms: vary query volume and data size ($\epsilon = 2$ and $d = 1$)

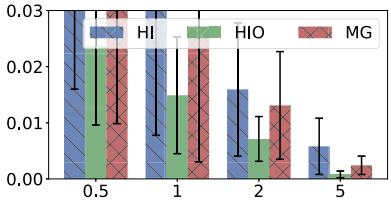
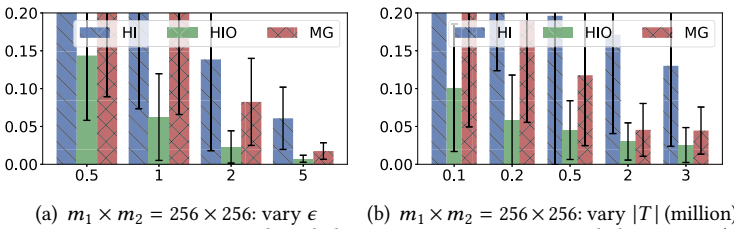
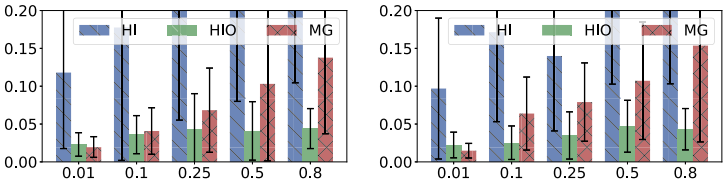


Figure 5: IPUMS 1M: vary ϵ ($d = 1$)



(a) $m_1 \times m_2 = 256 \times 256$: vary ϵ (b) $m_1 \times m_2 = 256 \times 256$: vary $|T|$ (million)

Figure 6: Two sensitive ordinal dimensions: vary ϵ and data size ($d = 2$)



(a) $m_1 \times m_2 = 256 \times 256$: vary vol($q$) (b) $m_1 \times m_2 = 1024 \times 64$: vary vol($q$)

Figure 7: Two sensitive dimensions: vary query volume ($\epsilon = 2$ and $d = 2$)

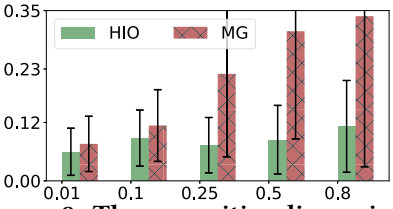


Figure 8: Three sensitive dimensions: vary query volume ($\epsilon = 2$ and $d = 3$)

256×256 and 1024×64. The query predicate is the conjunction of two range constraints: $D_1 \in [l_1, r_1] \wedge D_2 \in [l_2, r_2]$, with volume $\text{vol}(q) = (r_1 - \ell_1 + 1) \times (r_2 - \ell_2 + 1)/(m_1 \times m_2)$.

Figures 6-7 shows the results of different mechanisms for the two configurations, when varying ϵ, $|T|$, and vol(q). When $\text{vol}(q) \leq 0.01$, MG is better, which is consistent with our analysis in Section 5.4; otherwise, MG is much worse than HIO for varying ϵ and $|T|$, as an MDA query with two range constraints can be decomposed into too many marginal cells, whose errors accumulate when being aggregated.

6.1.3 Three Ordinal Dimensions. We create three sensitive ordinal dimensions on IPUMS with sizes $256 \times 256 \times 64$. The query predicate is the conjunction of the two range constraints. Errors in HI are much larger than those in HIO, so we omit it here. Figure 8 shows the results of HIO and MG, when varying vol(q). Errors in MG highly depend on vol(q), which is consistent with our analysis in Sections 3.4 and 5.4. On the other hand, although more challenging, HIO can still work, with consistently better estimations than MG. When $\text{vol}(q) \geq 0.5$, HIO is at least 3× more accurate.

6.2 Relative Error and Practical Usage

We now focus on the utility of our mechanisms in more real settings and conduct a case study. We will report their relative errors (MRE). We consider data models with four or more sensitive dimensions, but MDA queries may or may not contain all of them. When there are more than two sensitive dimensions, HI and MG already give much worse accuracy than HIO does. Thus, their performances are not reported here. We will evaluate HIO and SC, as their benefit is more pronounced when there are more sensitive attributes.

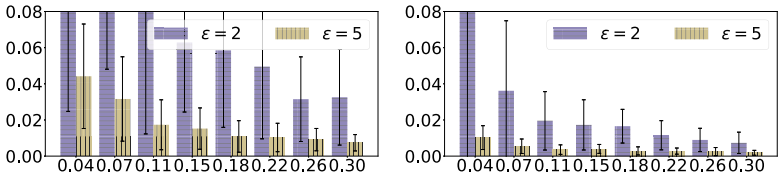
6.2.1 Two Ordinal and Two Categorical Dimensions. We start with four dimensions in IPUMS ($m = 5^4$ by default – we can change domain sizes via finer or looser bucketization).

Sample queries on with varying ϵ. Three sample queries $Q1$-$Q3$ (listed in Appendix G) are processed using HIO, and their estimated/true answers shown in Figure 9. The relative errors of estimations are within 5% most of the time. $Q3$ has the most selective predicate, and error in its estimated answers is also the largest among the three.

Varying selectivity of predicate. Since our theorems give guarantees on absolute errors for COUNT and SUM queries (*e.g.*, Theorem 9), relative errors are highly impacted by sizes of query answers, which in-turn depend on the selectivities of predicates. We test COUNT, SUM, and AVG queries with four types of predicates, 1+0, 1+1, 2+0, and 2+2 (*a+b means a ordinal dimensions and b categorical dimensions*), and varying selectivities. We plot the results in Figure 10 (COUNT queries have very similar trends to SUM). Their relative error decreases with increasing selectivity. AVG is estimated

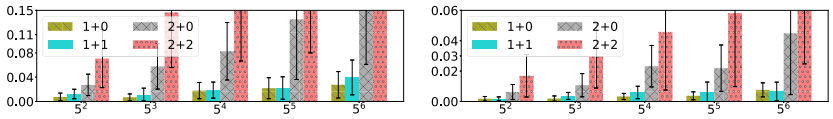
ϵ =	0.5	1	2	5	true
$Q1$	26.29	24.36	26.81	26.07	26.32
$Q2$	36.97	32.36	33.77	32.79	33.11
$Q3$	27.07	34.69	24.22	26.68	27.01

Figure 9: One-run estimations (using HIO) of sample AVG queries and true answers



(a) SUM queries (b) AVG queries

Figure 10: Relative error of HIO: vary selectivity



(a) ϵ = 2: vary domain size (b) ϵ = 5: vary domain size

Figure 11: Relative error of HIO on 2 (ordinal) + 2 (categorical) dimensions: vary domain sizes and query types (SUM queries)

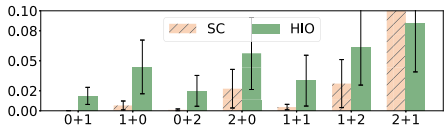


Figure 12: Relative error of HIO and SC on 4 (ordinal) + 4 (categorical) dimensions: vary query types (ϵ = 5)

ϵ =	0.5	1	2	5	true	selectivity
$Q4$	0.185	0.154	0.178	0.167	0.168	0.049
rel. err.	0.102	0.081	0.061	0.005	-	-
$Q5$	0.157	0.148	0.160	0.170	0.171	0.011
rel. err.	0.086	0.138	0.066	0.008	-	-

Table 2: One-run estimated answers in the case study

as SUM/COUNT, and thus, its relative error has a similar trend. We get reasonable relative errors for both $\epsilon = 2$ and 5.

Varying query type and domain size. For varying domain sizes and different types of predicates, we evaluate HIO and SC on queries with selectivity around 0.1. SC is worse in almost all types (refer to Figure 14 in Appendix G). Errors of HIO are reported in Figure 11. The errors are larger when the domains are larger due to the $\log m$ term in the error bounds. Also, queries of types 1+0 and 1+1 can be answered more accurately than 2+0 and 2+2 queries, which is consistent with Theorems 9 and 11 – the error increases as d_q increases.

6.2.2 Four Ordinal and Four Categorical Dimensions. We evaluate HIO and SC in the 8-dim setting, for different query types, and report the results in Figure 12. In this high-dim setting, according to Section 5.4, SC should give better estimations for queries with fewer dimensions in the predicates. Empirically, SC performs better than HIO for the almost all the query types except 2+1. This is consistent with our analytical results and is the purpose of introducing SC.

6.2.3 Case Study: E-Commerce Analytics. We test HIO for a real-world e-commerce application, where we want to collect delivery information from users in a privacy-preserving way. The table T collected via HIO contains more than 150 million users with four attributes. Attributes about each user's location (Region) and the product s/he bought (Category, Price) are sensitive dimensions, and the postage fee (Postage) is a public measure attribute. Suppose we want to analyze the postage distribution for certain group of users and products ($Q4$-$Q5$ listed in Appendix G), Table 2 gives the answers estimated by HIO for different ϵ. With the a large number of users, the accuracy is much improved.

7 EXTENSIONS AND DISCUSSION

Other space partitioning techniques. Frequency oracles can be combined with QuadTree to handle MDA queries: intuitively, a user can encode the tree nodes containing her/his tuple locally via frequency oracles. However, QuadTree incurs larger errors, because, to answer a 2D range query, in the worst case, the entire QuadTree needs to be traversed and thus too many noisy counts (the number is linear in the domain size) are added up which amplifies the error.

Coefficients in wavelet transforms (used in Privelet [40]) can be encoded using frequency oracles. Each user randomly selects a level in the decomposition tree of the wavelet transform, and reports his location on that level. However, as each level has a different weight in the estimation, it is unclear how to partition users across levels to optimize the utility.

In general, the idea proposed in this paper can also be used in other space partitioning techniques. Basically, each user has a local view of the data structure (with one data point), and report this structure under LDP. This can be optimized by having each user randomly select a sub-structure that has a fixed sensitivity (*e.g.*, a level). The server collects and adds up the LDP data structure from users, and then uses the result to answer queries. It is interesting future work to customize frequency oracles for different space partitioning techniques and compare them systematically.

Non-sensitive + private dimensions in predicates. If an MDA query has both private and public dimensions in the predicate, the server can evaluate the public part first, and, process the remaining rows/users with estimation processor

$\bar{\mathbf{P}}$ in our mechanisms. For example, in Table 1, the query

SELECT SUM(Purchase) FROM T WHERE
Age $\in [30, 40]$ AND OS = Win; $\Leftrightarrow$ can be evaluated as:
T_{pub} = SELECT $*$ FROM T WHERE OS = Win;
SELECT SUM(Purchase) FROM T_{pub} WHERE Age $\in [30, 40]$;

where T_{pub} can be evaluated using a normal query processor, and the last line is processed with $\bar{\mathbf{P}}$ in our mechanisms.

Handling other aggregation functions. As long as the aggregation function can be rewritten as SUM() functions, our mechanisms can be extended to handle it. For example, STDEV(M) can be computed from SUM(M^2), SUM(M), and COUNT. We can also support aggregations on multiple measures, *e.g.*, SUM($a \cdot M_1 + b \cdot M_2$), as long as M_1 and M_2 are public (conceptually, define $M' = a \cdot M_1 + b \cdot M_2$).

AND-OR expressions. An MDA query with OR in the predicate can be rewritten as sub-queries with only AND using the inclusion-exclusion principle. For example,

SELECT SUM(Purchase) FROM T
WHERE Age $\in [30, 40]$ OR Salary $\in$ [50K, 150K]

can be rewritten as three sub-queries: "... Age $\in [30, 40]$" + "... Salary $\in$ [50K, 150K]" − "... Age $\in [30, 40]$ AND Salary $\in$ [50K, 150K]", each to be estimated using $\bar{\mathbf{P}}$ in our mechanisms. More generally, an AND-OR expression can be converted into a disjunctive normal form, and we apply the inclusion-exclusion principle over clauses as in the above example.

8 RELATED WORK

We review related topics in both the centralized setting of differential privacy (DP) [15] and its local model (LDP).

LDP mechanisms. There have been several LDP frequency oracles [1, 4, 5, 9, 13, 16, 35] proposed. They rely on techniques like hashing (*e.g.*, [35]) and Hadamard transform (*e.g.*, [1, 4]) for good utility. LDP frequency oracle is also used in other tasks, *e.g.*, finding heavy hitters [4, 6, 36], frequent itemset mining [29, 37], and histogram estimation [24, 34, 38].

LDP mean/median estimation is another relevant line of works [9, 12, 13], as we also support AVG aggregation function in MDA queries. However, the settings are different and orthogonal: we assume that the attribute to be aggregated is public while the dimensions in the predicate are private.

Releasing marginals privately. Releasing marginal tables can be viewed as a special case of MDA queries (with only COUNT aggregations and equality constraints). This task has been studied exhaustively in the central setting of DP.

Laplace noise [14] can be injected to marginal tables to ensure DP. There are two ways to boost the accuracy of released marginal following this line. One is to inject noise to transformations of the data, *e.g.*, Fourier transform [3] or wavelet [40], and then reverse the transformations. Consistency across marginals is automatically enforced in this way. Another one is to carefully select a subset of marginals to inject Laplace noise and compute the rest marginals from them, with the goal of minimizing the max error in a workload [10]. There are also approaches based on, *e.g.*, multiplicative weights [19–21], Chebyshev polynomials [33], maximum entropy estimation [27], and sampling [7].

There have been several works on releasing marginals under LDP [8, 30, 41]. [30] generalizes the Expectation Maximization algorithm for estimating joint distribution of two attributes [18] to handle multiple attributes in a marginal. [8] refines and analyzes how to release marginals via transformations under LDP. [41] adapts the ideas of consistency enforcement and maximum entropy estimation from marginal release in the centralized setting [27] to LDP.

Answering range queries. Range counting queries are supported in the centralized setting of DP via, *e.g.*, hierarchical intervals [22] (one-dim range queries) or via wavelet [40] (multi-dimensional range queries). [28] optimizes the hierarchical intervals in [22] by choosing a proper branching factor. McKenna *et al.* [25] propose a method to collectively optimize errors in high-dimensional queries of a given workload under the centralized setting of DP. It will be interesting to design such workload-dependent techniques under LDP.

In both marginal release and range queries, it has been noticed that constrained inference could boost the accuracy while enforcing the consistency across different marginal tables and intervals (*e.g.*, [3, 10, 22, 28]). Since it is a post-processing step, all the consistency enforcement techniques in the centralized setting of DP can be potentially used in the LDP setting. We leave exploration on how to adapt such techniques for MDA under LDP to future work.

9 CONCLUSIONS

We study the problem of answering multi-dimensional analytical (MDA) queries under local differential privacy. Our approaches do not require a trusted data collector. Sensitive dimensions are encoded locally to preserve LDP. With LDP encoders and estimation algorithms, our approaches can answer a class of MDA queries with tight error bounds and scale well with a large number of dimensions.

Acknowledgments. We thank the anonymous reviewers for their helpful comments that improved the quality of the paper. Somesh Jha was partially supported by NSF grants CCF-FMitF-1836978 and SaTC-Frontiers-1804648 and CCF-1652140 and ARO grant number W911NF-17-1-0405. Ninghui Li was partially supported by NSF grant SaTC-1640374. Any opinions, findings, conclusions, and recommendations expressed herein are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. 2018. Hadamard Response: Estimating Distributions Privately, Efficiently, and with Little Communication. *CoRR* abs/1802.04705 (2018).

[2] Shipra Agrawal and Jayant R. Haritsa. 2005. A Framework for High-Accuracy Privacy-Preserving Mining. In *ICDE*.

[3] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. 2007. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS*.

[4] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. 2017. Practical Locally Private Heavy Hitters. In *NIPS*.

[5] Raef Bassily and Adam D. Smith. 2015. Local, Private, Efficient Protocols for Succinct Histograms. In *STOC*.

[6] Mark Bun, Jelani Nelson, and Uri Stemmer. 2018. Heavy hitters and the structure of local privacy. In *PODS*.

[7] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. 2015. Differentially private high-dimensional data publication via sampling-based inference. In *KDD*.

[8] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2018. Marginal Release Under Local Differential Privacy. In *SIGMOD*.

[9] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *NIPS*.

[10] Bolin Ding, Marianne Winslett, Jiawei Han, and Zhenhui Li. 2011. Differentially private data cubes: optimizing noise sources and consistency. In *SIGMOD*.

[11] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[12] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2013. Local Privacy and Statistical Minimax Rates. In *FOCS*.

[13] John C. Duchi, Martin J. Wainwright, and Michael I. Jordan. 2013. Local Privacy and Minimax Bounds: Sharp Rates for Probability Estimation. In *NIPS*.

[14] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *TCC*.

[15] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014). https://doi.org/10.1561/0400000042

[16] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *CCS*.

[17] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. 2003. Limiting privacy breaches in privacy preserving data mining. In *PODS*.

[18] Giulia C. Fanti, Vasyl Pihur, and Úlfar Erlingsson. 2016. Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries. *PoPETs* 2016, 3 (2016).

[19] Anupam Gupta, Moritz Hardt, Aaron Roth, and Jonathan Ullman. 2013. Privately Releasing Conjunctions and the Statistical Query Barrier. *SIAM J. Comput.* 42, 4 (2013).

[20] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A Simple and Practical Algorithm for Differentially Private Data Release. In *NIPS*.

[21] Moritz Hardt and Guy N. Rothblum. 2010. A Multiplicative Weights Mechanism for Privacy-Preserving Data Analysis. In *FOCS*.

[22] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the Accuracy of Differentially Private Histograms Through Consistency. *PVLDB* 3, 1 (2010).

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

[24] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. 2016. Discrete Distribution Estimation under Local Privacy. In *ICML*.

[25] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2018. Optimizing error of high-dimensional statistical queries under differential privacy. *PVLDB* 11, 10 (2018).

[26] Frank McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*.

[27] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2014. Priview: practical differentially private release of marginal contingency tables. In *SIGMOD*.

[28] Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding Hierarchical Methods for Differentially Private Histograms. *PVLDB* 6, 14 (2013).

[29] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2016. Heavy hitter estimation over set-valued data with local differential privacy. In *CCS*.

[30] 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 Trans. Information Forensics and Security* 13, 9 (2018).

[31] Steven Ruggles, Sarah Flood, Ronald Goeken, Josiah Grover, Erin Meyer, Jose Pacas, and Matthew Sobek. 2019. IPUMS USA: Version 9.0 [dataset].

[32] Apple Differential Privacy Team. 2017. Learning with Privacy at Scale.

[33] Justin Thaler, Jonathan Ullman, and Salil P. Vadhan. 2012. Faster Algorithms for Privately Releasing Marginals. In *ICALP*.

[34] Shaowei Wang, Liusheng Huang, Pengzhan Wang, Yiwen Nie, Hongli Xu, Wei Yang, Xiang-Yang Li, and Chunming Qiao. 2016. Mutual Information Optimally Local Private Discrete Distribution Estimation. *CoRR* abs/1607.08025 (2016).

[35] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally Differentially Private Protocols for Frequency Estimation. In *USENIX Security*.

[36] Tianhao Wang, Ninghui Li, and Somesh Jha. 2017. Locally Differentially Private Heavy Hitter Identification. *CoRR* abs/1708.06674 (2017).

[37] Tianhao Wang, Ninghui Li, and Somesh Jha. 2018. Locally Differentially Private Frequent Itemset Mining. In *SP*.

[38] Yue Wang, Xintao Wu, and Donghui Hu. 2016. Using Randomized Response for Differential Privacy Preserving Data Collection. In *EDBT/ICDT Workshops*.

[39] Stanley L. Warner. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *J. Amer. Statist. Assoc.* 60, 309 (1965).

[40] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2010. Differential privacy via wavelet transforms. In *ICDE*.

[41] Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, and Jiming Chen. 2018. CALM: Consistent Adaptive Local Marginal for Marginal Release under Local Differential Privacy. In *CCS*.

A UNWEIGHTED FREQUENCY ORACLE

Details of $(\mathcal{A}_{\mathsf{FO}}, \bar{\mathbf{f}})$, *i.e.*, OLH from [35], are in Algorithm 3.

Transition probabilities. Consider the input and output states defined in (27)-(28). From (36), we have

$$P_{1\to 1} \triangleq \mathbf{Pr}\left[A_i(t) = 1 \mid B_i(t) = 1\right] = \frac{e^{\epsilon}}{e^{\epsilon} + g - 1}.$$

From universal hashing and (36), we have

$$\begin{aligned} P_{0\to 1} &\triangleq \mathbf{Pr}\left[A_i(t) = 1 \mid B_i(t) = 0\right] \\ &= \mathbf{Pr}\left[H(v_i) = H(t[D_i]) \wedge \text{stay} \mid t[D_i] \neq v_i\right] \\ &+ \mathbf{Pr}\left[H(v_i) \neq H(t[D_i]) \wedge \text{flip} \mid t[D_i] \neq v_i\right], \end{aligned}$$

Client side: Encode private dimension $t[D]$.
Parameters: Privacy budget ϵ and hashing parameter g.

1: A user chooses a hash function $H \in \mathbb{H}$ uniformly at readom, where $\mathbb{H}$ is a universal hash function family such that every H outputs a value in $[g] = \{1, 2, \ldots, g\}$.
2: (Hashing) Let $x \leftarrow H(t[D])$.
3: (Perturbing) Draw $y \in [g]$ from the distribution:

$$\mathbf{Pr}[y = i] = \begin{cases} \frac{e^\epsilon}{e^\epsilon + g - 1}, & \text{for } i = x \text{ (stay)} \\ \frac{1}{e^\epsilon + g - 1}, & \text{for } i \neq x \text{ (flip)} \end{cases}. \tag{36}$$

4: Send $\mathcal{A}_{\mathsf{FO}}(t[D]) \leftarrow \langle H, y \rangle$.

Server side: Collect $\{\mathcal{A}_{\mathsf{FO}}(t[D])\}_{t \in S}$ from S. Answer frequency query $\mathbf{f}_S(v)$.

1: $\theta \leftarrow 0$.
2: For each $t \in S$: suppose $\mathcal{A}_{\mathsf{FO}}(t[D]) = \langle H, y \rangle$
3: If $H(v) = y$ then $\theta \leftarrow \theta + 1$;
4: Output an estimation to $\mathbf{f}_S(v)$ as

$$\bar{\mathbf{f}}_S(v) = \left(\theta - \frac{|S|}{g}\right) \cdot \frac{(e^\epsilon + g - 1)g}{e^\epsilon g - e^\epsilon - g + 1} \tag{37}$$

$$= \frac{2(e^\epsilon + 1)\theta}{(e^\epsilon - 1)} - \frac{2|S|}{e^\epsilon - 1} \text{ when } g = e^\epsilon + 1. \tag{38}$$

Algorithm 3: Unweighted frequency oracle OLH [35]

where "stay" and "flip" are the two events in (36), and thus,

$$P_{0\to1} = \frac{1}{g} \cdot \frac{e^\epsilon}{e^\epsilon + g - 1} + \frac{g-1}{g} \cdot \frac{1}{e^\epsilon + g - 1} = \frac{1}{g}.$$

Indeed, we have $P_{1\to0} = 1 - P_{1\to1}$ and $P_{0\to0} = 1 - P_{0\to1}$.

B MORE DETAILS OF HI MECHANISM

Our hierarchical-interval (HI) mechanism is in Algorithm 4.

C HI WITH CATEGORICAL DIMENSIONS

For an MDA query with "$D = v_i$" in the predicate, we would refer to multi-dim intervals with $[v_i]$ from $\mathcal{L}^1_D$; if D is not in the predicate, we refer to those with $*$ from $\mathcal{L}^0_D$.

Example C.1. *Figure 13 shows a 2-dim hierarchy with an ordinal dimension D_1 and a categorical one D_3 (*State*). There are 4×2 2-dim levels. Each 2-dim interval is a pair of an interval on D_1 and a value (or $*$) on D_3. Two 2-dim levels, $\mathcal{L}^2_{D_1} \times \mathcal{L}^0_{D_2}$ and $\mathcal{L}^2_{D_1} \times \mathcal{L}^1_{D_2}$, are depicted on the right, with 4×1 and 4×4 2-dim intervals, respectively. Tuple t with $t[D_1] = 3$ and $t[D_3] =$ WA has $t[\mathcal{L}^2_{D_1} \times \mathcal{L}^0_{D_2}] = ([3, 4]*)$ and $t[\mathcal{L}^2_{D_1} \times \mathcal{L}^1_{D_2}] = ([3, 4]\text{WA})$ on the two augmented dimensions.*

Consider the following MDA query to be processed:

$$q_3 : \text{SELECT SUM}(M_1) \text{ FROM } T$$
$$\text{WHERE } D_1 \in [2, 7] \text{ AND } D_3 = \text{WA}$$

Client side: Encode dimensions $t[D_1], \ldots, t[D_d]$.

1: For each $(j_1, \ldots, j_d) \in \{0, 1, \ldots, h\}^d$ do:
2: Suppose $t[D_i]$ is in interval $I^{j_i}_i \in \mathcal{L}^{j_i}_{D_i}$ $(i = 1, \ldots, d)$: let $t[\mathcal{L}^{j_1}_{D_1} \times \ldots \times \mathcal{L}^{j_d}_{D_d}] \leftarrow I^{j_1}_1 I^{j_2}_2 \ldots I^{j_d}_d$;
3: Create LDP report with $(h+1)^d$ instances of $\mathcal{A}^{\epsilon'}_{\mathsf{FO}}$ where $\epsilon' = \epsilon/(h+1)^d$, one for each d-dim level:

$$\mathcal{A}_{\mathsf{HI}}(t) \leftarrow \left\langle \mathcal{A}^{\epsilon'}_{\mathsf{FO}}(t[\mathcal{L}^{j_1}_{D_1} \times \ldots \times \mathcal{L}^{j_d}_{D_d}]) \right\rangle_{\substack{(j_1, j_2, \ldots, j_d) \in \\ \{0,1,\ldots,h\}^d}} \tag{39}$$

Server side: MDA query $q = Q_T(\text{SUM}(M), D_1 \in [l_1, r_1] \wedge \ldots \wedge D_d \in [l_d, r_d])$.

1: For $i = 1$ to d do:
2: Decompose $[l_i, r_i]$ into p_i disjoint intervals $[l_i, r_i] \to I^1_i \cup I^2_i \cup \ldots \cup I^{p_i}_i$ in the hierarchy $\mathbf{I}_{D_i}$;
3: For each $(i_1, \ldots, i_d) \in \{1, \ldots, p_1\} \times \ldots \times \{1, \ldots, p_d\}$:
4: Estimate $\mathbf{f}^M_T(I^{i_1}_1 I^{i_2}_2 \ldots I^{i_d}_d)$ as $\bar{\mathbf{f}}^M_T(I^{i_1}_1 I^{i_2}_2 \ldots I^{i_d}_d)$;
5: Output an estimation to q as:

$$\bar{\mathbf{P}}_{\mathsf{HI}}(q) = \sum_{1 \le i_1 \le p_1, \ldots, 1 \le i_d \le p_d} \bar{\mathbf{f}}^M_T(I^{i_1}_1 I^{i_2}_2 \ldots I^{i_d}_d). \tag{40}$$

Algorithm 4: d-dim HI Mechanism $(\mathcal{A}_{\mathsf{HI}}, \bar{\mathbf{P}}_{\mathsf{HI}})$

which can be decomposed into 4 sub-queries. On the level $\mathcal{L}^2_{D_1} \times \mathcal{L}^1_{D_2}$ there are two, with "$D_1 \in [3, 4]$ AND $D_3 =$ WA" and "$D_1 \in [5, 6]$ AND $D_3 =$ WA", respectively. Again, we can use LDP weighted frequency oracles to encode the augmented dimensions and approximate these sub-queries.

When analyzing error bounds, we only need to note that categorical hierarchies have only two levels ($h = 1$).

Corollary 12. *Suppose there are d_1 categorical dimensions and d_2 (d_q) ordinal ones (in the query q), when ϵ is small,*

$$\text{Err}(\bar{\mathbf{P}}_{\mathsf{HI}}(q)) = \text{O}\left(\frac{n\Delta^2 2^{2d_1} \log^{d_q+2d_2} m}{\epsilon^2}\right) \text{ and}$$

$$\text{Err}(\bar{\mathbf{P}}_{\mathsf{HIO}}(q)) = \text{O}\left(\frac{n\Delta^2 2^{d_1} \log^{d_q+d_2} m}{\epsilon^2}\right).$$

D MORE DETAILS OF SC MECHANISM

Our split-and-conjunction (SC) mechanism is in Algorithm 5. On the client side, *each one-dim level of all the dimensions* is encoded using an independent weighted frequency oracle (lines 3 and 4), with equal privacy budget $\epsilon/(dh)$. On the server side, under the same query decomposition scheme, its estimation process is almost identical to that of HI, except that conjunctive estimators $\hat{\mathbf{f}}^M$ on *query dimensions* (instead of estimated weighted frequencies $\bar{\mathbf{f}}^M$ as in Algorithm 4) are assembled as the estimated answer to q (lines 4 and 5).

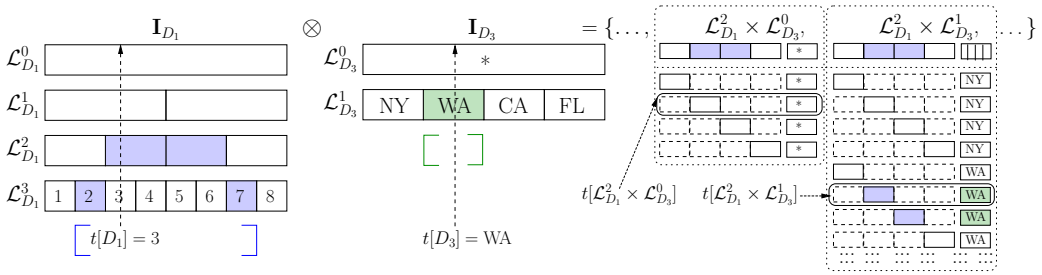


Figure 13: 2D hierarchy on ordinal+categorical dimensions, query decomposition, and HI mechanism

Client side: Encode dimensions $t[D_1], \ldots, t[D_d]$.

1: For each $i = 1$ to d do:
2: For each $j = 1$ to h do:
3: Suppose $t[D_i]$ is in interval $I_i^j \in \mathcal{L}^j_{D_i}$:
let $t[\mathcal{L}^j_{D_i}] \leftarrow I_i^j$;
4: Create LDP report with dh instances of $\mathcal{A}_{\text{FO}}^{\epsilon/(dh)}$:

$$\mathcal{A}_{\text{SC}}(t) \leftarrow \left\langle \mathcal{A}_{\text{FO}}^{\epsilon/(dh)}(t[\mathcal{L}^j_{D_i}]) \right\rangle_{\substack{i=1,\ldots,d \\ j=1,\ldots,h}} \tag{41}$$

Server side: MDA query $q = Q_T(\text{SUM}(M), D_1 \in [l_1, r_1] \wedge \ldots \wedge D_d \in [l_d, r_d])$.

1: For $i = 1$ to d do:
2: Decompose $[l_i, r_i]$ into p_i disjoint intervals $[l_i, r_i] \rightarrow I_i^1 \cup I_i^2 \cup \ldots \cup I_i^{p_i}$ in the hierarchy $\mathbf{I}_{D_i}$;
3: For each $(i_1, \ldots, i_d) \in \{1, \ldots, p_1\} \times \ldots \times \{1, \ldots, p_d\}$:
4: Estimate $\mathbf{f}_T^M(I_1^{i_1} I_2^{i_2} \ldots I_d^{i_d})$ as $\hat{\mathbf{f}}_T^M(I_1^{i_1} I_2^{i_2} \ldots I_d^{i_d})$;
5: Output an estimation to q as:

$$\bar{\mathbf{P}}_{\text{SC}}(q) = \sum_{1 \le i_1 \le p_1, \ldots, 1 \le i_d \le p_d} \hat{\mathbf{f}}_T^M(I_1^{i_1} I_2^{i_2} \ldots I_d^{i_d}). \tag{42}$$

Algorithm 5: d-dim SC Mechanism $(\mathcal{A}_{\text{SC}}, \bar{\mathbf{P}}_{\text{SC}})$

E PROOFS

Proof of Proposition 4:
From (7) and the linearity of expectation,

$$\mathbf{E}\left[\bar{\mathbf{f}}_S^M(v)\right] = \sum_x x \cdot \mathbf{E}\left[\bar{\mathbf{f}}_{S_x}(v)\right] = \sum_x x \cdot \mathbf{f}_{S_x}(v) = \mathbf{f}_S(v),$$

as $\bar{\mathbf{f}}_{S_x}$ is unbiased from Lemma 3. The squared error is:

$$\mathbf{Var}\left[\bar{\mathbf{f}}_S^M(v)\right] = \sum_{\text{distinct } x} x^2 \cdot \mathbf{Var}\left[\bar{\mathbf{f}}_{S_x}(v)\right]$$

$$= \sum_x x^2 \cdot \left(\frac{4|S_x|e^\epsilon}{(e^\epsilon - 1)^2} + \mathbf{f}_{S_x}(v) \right) = \frac{4M_S^2 e^\epsilon}{(e^\epsilon - 1)^2} + M_S^2(v), \tag{43}$$

where (43) is from the fact that $\mathcal{A}_{\text{FO}}$ is used by each user independently, and the error bound of $\bar{\mathbf{f}}_S$ in Lemma 3.

$\mathbf{Var}[\bar{\mathbf{f}}_S^M(u) + \bar{\mathbf{f}}_S^M(v)] = \mathbf{Var}[\bar{\mathbf{f}}_S^M(u)] + \mathbf{Var}[\bar{\mathbf{f}}_S^M(v)]$ is from $\mathbf{Var}[\bar{\mathbf{f}}_S(u) + \bar{\mathbf{f}}_S(v)] = \mathbf{Var}[\bar{\mathbf{f}}_S(u)] + \mathbf{Var}[\bar{\mathbf{f}}_S(v)]$ ($\bar{\mathbf{f}}_S(u)$ and $\bar{\mathbf{f}}_S(v)$ are not independent but their covariance is 0, which can be verified though careful calculations) and (43). □

Proof of Proposition 5:
The unbiasedness has been proved. We can decompose the error $\text{Err}(\tilde{\mathbf{f}}_{S,1/k}^M(v)) = \mathbf{Var}\left[\tilde{\mathbf{f}}_{S,1/k}^M(v)\right]$ into three parts:

$$\begin{aligned}
&\mathbf{Var}\left[\tilde{\mathbf{f}}_{S,1/k}^M(v)\right] = \mathbf{E}\left[\left(k\bar{\mathbf{f}}_{S_1}^M(v) - \mathbf{f}_S^M(v)\right)^2\right] \\
=&\mathbf{E}\left[\left((k\bar{\mathbf{f}}_{S_1}^M(v) - k\mathbf{f}_{S_1}^M(v)) + (k\mathbf{f}_{S_1}^M(v) - \mathbf{f}_S^M(v))\right)^2\right] \\
=&k^2\mathbf{E}\left[\left(\bar{\mathbf{f}}_{S_1}^M(v) - \mathbf{f}_{S_1}^M(v)\right)^2\right] + \mathbf{E}\left[\left(k\mathbf{f}_{S_1}^M(v) - \mathbf{f}_S^M(v)\right)^2\right] + \\
&2k\mathbf{E}\left[(\bar{\mathbf{f}}_{S_1}^M(v) - \mathbf{f}_{S_1}^M(v)) \cdot (k\mathbf{f}_{S_1}^M(v) - \mathbf{f}_S^M(v))\right].
\end{aligned} \tag{44}$$

Using conditional expectation and Proposition 4, we have

$$\begin{aligned}
&\mathbf{E}\left[\left(\bar{\mathbf{f}}_{S_1}^M(v) - \mathbf{f}_{S_1}^M(v)\right)^2\right] = \mathbf{E}\left[\mathbf{E}\left[\left(\bar{\mathbf{f}}_{S_1}^M(v) - \mathbf{f}_{S_1}^M(v)\right)^2 \,\middle|\, S_1\right]\right] \\
=&\mathbf{E}\left[M_{S_1}^2 \cdot \frac{4e^\epsilon}{(e^\epsilon - 1)^2} + M_{S_1}^2(v)\right] = \frac{4M_S^2 e^\epsilon}{k(e^\epsilon - 1)^2} + \frac{M_S^2(v)}{k}.
\end{aligned} \tag{45}$$

And from the standard analysis on sampling process,

$$\mathbf{E}\left[\left(k\mathbf{f}_{S_1}^M(v) - \mathbf{f}_S^M(v)\right)^2\right] = (k-1)M_S^2(v). \tag{46}$$

For the rest term, we can show

$$\begin{aligned}
&\mathbf{E}\left[(\bar{\mathbf{f}}_{S_1}^M(v) - \mathbf{f}_{S_1}^M(v)) \cdot (k\mathbf{f}_{S_1}^M(v) - \mathbf{f}_S^M(v))\right] \\
=&\mathbf{E}\left[(\bar{\mathbf{f}}_{S_1}^M(v) - \mathbf{f}_{S_1}^M(v)) \cdot k\mathbf{f}_{S_1}^M(v)\right] \\
&\text{(as } \mathbf{E}[\bar{\mathbf{f}}_{S_1}^M(v)] = \mathbf{E}[\mathbf{f}_{S_1}^M(v)] \text{ and } \mathbf{f}_S^M(v) \text{ is a constant)} \\
=&k\mathbf{E}\left[\mathbf{E}\left[(\bar{\mathbf{f}}_{S_1}^M(v) - \mathbf{f}_{S_1}^M(v)) \cdot \mathbf{f}_{S_1}^M(v) \,\middle|\, S_1\right]\right] = 0. \\
&\text{(as for a fixed } S_1, \mathbf{E}[\bar{\mathbf{f}}_{S_1}^M(v)] = \mathbf{E}[\mathbf{f}_{S_1}^M(v)])
\end{aligned} \tag{47}$$

Putting (45)-(47) back to (44), we can derive the variance. □

Proof of Theorem 6:
The privacy guarantee, ϵ-LDP of $\mathcal{A}_{\text{HI}}$, follows directly from the sequential composability of DP (ϵ is partitioned on h levels and the change in $t[D]$ can affect all the h levels).

Any interval $[l, r]$ on D can be decomposed into $p \leq 2(b-1)\log_b m$ disjoint intervals $I^1, \ldots, I^p$. The unbiasedness is from the fact that each sub-query in (12) is equivalent to $\mathbf{f}_T^M(I^i)$ and $\tilde{\mathbf{f}}_T^M(I^i)$ is an unbiased estimator of it. The error follows from Proposition 4 (about each $\bar{\mathbf{f}}_T^M(I^i)$'s error). □

Proof of Theorem 7:
$\mathcal{A}_{\text{HIO}}$ is ϵ-LDP because each user applies $\mathcal{A}_{\text{FO}}$ on one level with privacy budget ϵ. Unbiasedness is because each estimator $\tilde{\mathbf{f}}_{T,1/h}^M(I^i)$ in line 6 is an unbiased estimator of $\mathbf{f}_T^M(I^i)$ from Proposition 5. Again, from Proposition 5, we know

$$\text{Err}(\tilde{\mathbf{f}}_{T,1/h}^M(I^i)) \leq \frac{2hM_T^2(e^{2\epsilon}+1)}{(e^\epsilon-1)^2}.$$

We have $h = \log_b m$ in HIO and there are $p \leq 2(b-1)\log_b m$ sub-queries. Therefore, the error upper bound follows. □

Proof of Theorem 8:
The analysis for 1D HI (Theorem 6) naturally extends here. The privacy budget is partitioned for $(h+1)^d = (\log_b m+1)^d$ instances of $\mathcal{A}_{\text{FO}}$. Unbiasedness of $\bar{\mathbf{P}}_{\text{HI}}$ is from the unbiasedness of $\tilde{\mathbf{f}}^M$. And the query q can be answered from $(2(b-1)\log_b m)^{d_q}$ or less sub-queries. □

Proof of Theorem 9:
Similar to the proof of Theorem 7. The term $(2(b-1)(\log_b m+1))^{d_q}$ is the upper bound of the number of sub-queries q is decomposed into. The term $(\log_b m+1)^d$ is from running weighted frequency oracles on random samples. □

Proof of Proposition 10:
The unbiasedness is from the fact that $\mathbf{P}\cdot\mathbf{b} = \mathbf{E}[\mathbf{a}]$, which we have discussed in Section 5.3.1 for the 2-dim case. The k-dim case is similar. Error bounds follow from the property of an inversed transition matrix $\mathbf{P}^{-1}$ and the (co)variance of $\mathbf{a}$. □

Proof of Theorem 11:
ϵ-LDP is straightforward from the above discussion. Unbiasedness of $\bar{\mathbf{P}}_{\text{SC}}$ is from unbiasedness of $\hat{\mathbf{f}}^M$ (Proposition 10). For the worst-case error, since there are $O(\log^{d_q} m)$ sub-queries, each of which is approximated by a conjunctive estimator with an error (from Proposition 10)

$$O\left(\frac{n\Delta^2}{(\epsilon/d\log m)^{2d_q}}\right).$$

Putting them together, it concludes the proof. □

F COMPLEXITY ANALYSIS

Table 3 is a summary about complexity of mechanisms $(\mathcal{A}, \bar{\mathbf{P}})$ we have introduced. Here, n is the total number of users, d is the total number of dimensions, and d_q is the number of dimension in an MDA query. Assume that all dimensions are ordinal with the max range equal to m.

Mechanism	Encoder time/space per user on client	Worst-case query pro-cessing time on server
$(\mathcal{A}_{\text{MG}}, \bar{\mathbf{P}}_{\text{MG}})$	$O(1)$	$O\left(n + m^d\right)$
$(\mathcal{A}_{\text{HI}}, \bar{\mathbf{P}}_{\text{HI}})$	$O\left(\log^d m\right)$	$O\left(n\log^{d_q} m\right)$
$(\mathcal{A}_{\text{HIO}}, \bar{\mathbf{P}}_{\text{HIO}})$	$O(1)$	$O\left(n + \log^{d_q} m\right)$
$(\mathcal{A}_{\text{SC}}, \bar{\mathbf{P}}_{\text{SC}})$	$O(d\log m)$	$O\left((nd_q + 4^{d_q})\log^{d_q} m\right)$

Table 3: Complexity of different LDP mechanisms

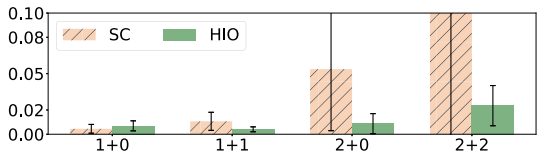


Figure 14: HIO and SC on 2 (ordinal) + 2 (categorical) dimensions: vary query types ($\epsilon = 5$ and $m = 5^2$)

"Encode space per user" is the size of an LDP report encoded by $\mathcal{A}$ and sent to the server from each user (in words). The time taken by $\mathcal{A}$ per user is linear in the size, and thus they are listed in the same column in Table 3. "Query processing time" is the time taken by $\bar{\mathbf{P}}$ to estimate the answer of a query. The proofs are straightforward and thus are omitted here: we just need to count how many augmented dimensions each user reports in different mechanisms, and how many sub-queries an MDA query is decomposed into.

G MORE DETAILS OF EXPERIMENTS

$Q1$: SELECT AVG(weekly_work_hour) FROM IPUMS
WHERE marital_status = Married;

$Q2$: SELECT AVG(weekly_work_hour) FROM IPUMS
WHERE marital_status = Married AND age ∈ [40, 60];

$Q3$: SELECT AVG(Postage) FROM Transactions
WHERE Price ≤ 50 AND Region = State_X;

$Q4$: SELECT AVG(Postage) FROM Transactions
WHERE Price ≤ 50 AND Category = 52001;

Comparing SC with HIO on four dimensions. Under the setting of Section 6.2.1, we compare SC with HIO for different types of queries and report the results in Figure 14. For the low-dimensional predicates "1+0" and "1+1", they have comparable accuracy; otherwise, HIO is much better.