

# A workload-adaptive mechanism for linear queries under local differential privacy

Ryan McKenna

Raj Kumar Maity

Arya Mazumdar

Gerome Miklau

College of Information and Computer Sciences  
University of Massachusetts, Amherst

{rmckenna, rajkmaity, arya, miklau}@cs.umass.edu

## ABSTRACT

We propose a new mechanism to accurately answer a user-provided set of linear counting queries under local differential privacy (LDP). Given a set of linear counting queries (the workload) our mechanism automatically adapts to provide accuracy on the workload queries. We define a parametric class of mechanisms that produce unbiased estimates of the workload, and formulate a constrained optimization problem to select a mechanism from this class that minimizes expected total squared error. We solve this optimization problem numerically using projected gradient descent and provide an efficient implementation that scales to large workloads. We demonstrate the effectiveness of our optimization-based approach in a wide variety of settings, showing that it outperforms many competitors, even outperforming existing mechanisms on the workloads for which they were intended.

### PVLDB Reference Format:

Ryan McKenna Raj Kumar Maity Arya Mazumdar Gerome Miklau . A workload-adaptive mechanism for linear queries under local differential privacy . *PVLDB*, 13(11): 1905-1918, 2020.  
DOI: <https://doi.org/10.14778/3407790.3407798>

## 1. INTRODUCTION

In recent years, Differential Privacy [15] has emerged as the dominant approach to privacy and its adoption in practical settings is growing. Differential privacy is achieved with carefully designed randomized algorithms, called mechanisms. The aim of these mechanisms is to extract utility from the data while adhering to the constraints imposed by differential privacy. Utility is measured on a task-by-task basis, and different tasks require different mechanisms. Utility-optimal mechanisms, or mechanisms that maximize utility for a given task under a fixed privacy budget, are still not known in many cases.

There are two main models of differential privacy: the central model and the local model. In the central model, users provide their data to a trusted data curator, who runs a privacy mechanism on the dataset in its entirety. In the local

model, users execute a privacy mechanism before sending it to the data curator. Local differential privacy (LDP) offers a stronger privacy guarantee than central differential privacy, as it does not rely on the assumption of a trusted data curator. For that reason, it has been embraced by several organizations like Google [18], Apple [36], and Microsoft [14] for the collection of personal data from customers. While the stronger privacy guarantee is a benefit of the local model, it necessarily leads to greater error than the central model [16], which makes error-optimal mechanisms an important goal.

Our focus is answering a workload of linear counting queries under local differential privacy. Answering a query workload is a general task that subsumes other common tasks, like estimating histograms, range queries, and marginals. Furthermore, the expressivity of linear query workloads goes far beyond these special cases, as it can include an arbitrary set of predicate counting queries. By defining the workload, the analyst expresses the exact queries they care about most, and their relative importance. There are several LDP mechanisms for answering particular fixed workloads, like histograms [1, 42, 5, 38], range queries [13, 39], and marginals [12, 39]. These mechanisms were carefully crafted to provide accuracy on the workloads for which they were designed, but their accuracy properties typically do not transfer to other workloads. Some LDP mechanisms are designed to answer an arbitrary collection of linear queries [4, 17], but they do not outperform simple baselines in practice.

In this paper, we propose a new mechanism that automatically adapts in order to prioritize accuracy on a target workload. Adaptation to the workload is accomplished by solving a numerical optimization problem, in which we search over an expressive class of unbiased LDP mechanisms for one that minimizes variance on the workload queries.

Workload-adaptation [21, 27, 33] is a much more developed topic in the central model of differential privacy and has led to mechanisms that offer best-in-class error rates in some settings [22]. Our work is conceptually similar to the Matrix Mechanism [27, 33], which also minimizes variance over a class of unbiased mechanisms. However, because the class of mechanisms we consider is different, the optimization problem is fundamentally different and requires a novel analysis and algorithmic solution. In addition, the optimal mechanism in our problem formulation depends on the setting of the privacy parameter,  $\epsilon$ , while in the Matrix Mechanism this is not a factor. We thoroughly discuss the similarities and differences between these two mechanisms in Section 7.

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>. For any use beyond those covered by this license, obtain permission by emailing [info@vldb.org](mailto:info@vldb.org). Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

*Proceedings of the VLDB Endowment*, Vol. 13, No. 11

ISSN 2150-8097.

DOI: <https://doi.org/10.14778/3407790.3407798>

**Contributions.** The paper consists of four main technical contributions.

- We propose a new class of mechanisms, called *workload factorization mechanisms*, that generalizes many existing LDP mechanisms, and we formulate an optimization problem to select a mechanism from this class that is optimally tailored to a workload (Section 3).
- We give an efficient algorithm to approximately solve this optimization problem, by reformulating it into an algorithmically tractable form (Section 4).
- We provide a theoretical analysis which illuminates error properties of the mechanism and justifies the design choices we made (Section 5).
- In a comprehensive set of experiments we test our mechanism on a range workloads, showing that it consistently delivers lower error than competitors, by as much as a factor of 14.6 (Section 6).

## 2. BACKGROUND AND PROBLEM SETUP

In this section we introduce notation for the input data and query workload as well as provide basic definitions of local differential privacy. A full review of notation is provided in the Appendix.

### 2.1 Input Data and Workload

Given a domain  $\mathcal{U}$  of  $n$  distinct user types, the input data is a collection of  $N$  users  $\langle u_1, \dots, u_N \rangle$ , where each  $u_i \in \mathcal{U}$ . We commonly use a vector representation of the input data, containing a count for each possible user type:

**Definition 2.1 (Data Vector)** *The data vector, denoted by  $\mathbf{x}$ , is an  $n$ -length vector of counts indexed by user types  $u \in \mathcal{U}$  such that:*

$$x_u = \sum_{j=1}^N \mathbb{1}\{u_j = u\} \quad \forall u \in \mathcal{U}.$$

In the local model, we do not have direct access to  $\mathbf{x}$ , but it is still useful to define it for the purpose of analysis. Below is a simple data vector one might obtain from a data set of student grades.

**Example 2.2 (Student Data)** *Consider a data set of student grades, where  $\mathcal{U} = \{A, B, C, D, F\}$ . Suppose 10 students got an A, 20 students got a B, 5 students got a C and no students got a D or F. Then the data vector would be:*

$$\mathbf{x} = [10 \quad 20 \quad 5 \quad 0 \quad 0]^T$$

Linear counting queries have a similar vector representation, as shown in Definition 2.3.

**Definition 2.3 (Linear Query)** *A linear counting query is an  $n$ -length vector  $\mathbf{w}$  indexed by user types  $u \in \mathcal{U}$ , such that the answer to the query is  $\mathbf{w}^T \mathbf{x} = \sum_{u \in \mathcal{U}} w_u x_u$ .*

A workload is a collection  $p$  linear queries  $\mathbf{w}_1, \dots, \mathbf{w}_p \in \mathbb{R}^n$  organized into a  $p \times n$  matrix  $\mathbf{W}$ . Our goal is to accurately estimate answers to each workload query under local differential privacy, i.e., we want to privately estimate  $\mathbf{W}\mathbf{x}$ .

The most commonly studied workload is the so-called Histogram workload, which is represented by a  $n \times n$  identity matrix. A more interesting workload is given below:

**Example 2.4 (Prefix workload)** *The prefix workload contains queries that compute the (unnormalized) empirical cumulative distribution function of the data, or the number of students that have grades  $\geq A, \geq B, \geq C, \geq D, \geq F$ .*

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The workload  $\mathbf{W}$  is an input to our algorithm, reflecting the queries of interest to the analyst and therefore determining the measure of utility that will be used to assess algorithm performance. In this setup, we make no assumptions about the structure or contents of  $\mathbf{W}$ , and allow it to be completely arbitrary, even including the same query multiple times or multiple linearly dependent queries.

### 2.2 Local Differential Privacy

Local differential privacy (LDP) [16] is a property of a randomized mechanism  $\mathcal{M}$  that acts on user data. Instead of reporting their true user type, users instead report a randomized response obtained by executing  $\mathcal{M}$  on their true input. These randomized responses allow an analyst to learn something about the population as a whole, while still providing the individual users a form of plausible deniability about their true input. The formal requirement on  $\mathcal{M}$  is stated in Definition 2.5.

**Definition 2.5 (Local Differential Privacy)** *A randomized mechanism  $\mathcal{M} : \mathcal{U} \rightarrow \mathcal{O}$  is said to be  $\epsilon$ -LDP if an only if, for all  $u, u' \in \mathcal{U}$  and all  $S \subseteq \mathcal{O}$ :*

$$\Pr[\mathcal{M}(u) \in S] \leq \exp(\epsilon) \Pr[\mathcal{M}(u') \in S]$$

The output range  $\mathcal{O}$  can vary between mechanisms. In some simple cases, it is the same as the input domain  $\mathcal{U}$ , but it does not have to be. Choosing the output range is typically the first step in designing a mechanism. When the range of the mechanism is finite, i.e.,  $|\mathcal{O}| = m$ , we can completely specify a mechanism by a so-called *strategy matrix*  $\mathbf{Q} \in \mathbb{R}^{m \times n}$ , indexed by  $(o, u) \in \mathcal{O} \times \mathcal{U}$ . The mechanism  $\mathcal{M}_{\mathbf{Q}}(u)$  is then defined by:

$$\Pr[\mathcal{M}_{\mathbf{Q}}(u) = o] = Q_{o,u}$$

This encoding of a mechanism essentially stores a probability for every possible input, output pair in the strategy matrix  $\mathbf{Q}$ . We translate Definition 2.5 to strategy matrices in Proposition 2.6.

**Proposition 2.6 (Strategy Matrix)** *A mechanism  $\mathcal{M}_{\mathbf{Q}}$  is  $\epsilon$ -LDP if and only if the following conditions are satisfied:*

1.  $Q_{o,u} \leq \exp(\epsilon) Q_{o,u'}, \forall o, u, u'$ .
2.  $Q_{o,u} \geq 0, \forall o, u, \sum_o Q_{o,u} = 1, \forall u$ .

Above, the first condition is the privacy constraint, ensuring that the output distributions for any two users are close, and the second is the probability simplex constraint, ensuring that each column of  $\mathbf{Q}$  corresponds to a valid probability distribution. Representing mechanisms as matrices is useful because it allows us to reason about them mathematically with linear algebra [26, 24]. Example 2.7 shows how a simple

**Table 1:** Existing LDP mechanisms encoded as a strategy matrix.  $e_u$  is the one-hot encoding of  $u$ ,  $\mathbf{H}$  is the  $K \times K$  Hadamard matrix, and  $d$  is a hyper-parameter.

Mechanism	Input	Output	Strategy Matrix
<b>Randomized Response</b> [41]	$u \in [n]$	$o \in [n]$	$Q_{o,u} \propto \begin{cases} \exp(\epsilon) & o = u \\ 1 & o \neq u \end{cases}$
<b>RAPPOR</b> [18]	$u \in [n]$	$o \in \{0, 1\}^n$	$Q_{o,u} \propto \exp\left(\frac{\epsilon}{2}\right)^{n - \ o - e_u\ _1}$
<b>Hadamard</b> [1]	$u \in [n]$	$o \in [K]$ $K = \lceil 2^{\lceil \log_2(n+1) \rceil} \rceil$	$Q_{o,u} \propto \begin{cases} \exp(\epsilon) & H_{o+1,u} = 1 \\ 1 & H_{o+1,u} = -1 \end{cases}$
<b>Subset Selection</b> [42]	$u \in [n]$	$o \in \{0, 1\}^n$ $\ o\ _1 = d$	$Q_{o,u} \propto \begin{cases} \exp(\epsilon) & o_u = 1 \\ 1 & o_u = 0 \end{cases}$

mechanism, called randomized response<sup>1</sup>, can be encoded as a strategy matrix.

**Example 2.7 (Randomized Response)** *The randomized response mechanism [41] can be encoded as a strategy matrix in the following way:*

$$\mathbf{Q} = \frac{1}{e^\epsilon + n - 1} \begin{bmatrix} e^\epsilon & 1 & \dots & 1 \\ 1 & e^\epsilon & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & e^\epsilon \end{bmatrix}$$

For this mechanism, the output range is the same as the input domain, and hence the strategy matrix is square. The diagonal entries of the strategy matrix are proportional to  $e^\epsilon$ , and the off-diagonal entries are proportional to 1. This means that each user reports their true input with probability proportional to  $e^\epsilon$  and all other possible outputs with probability proportional to 1. It is easy to see that the conditions of Proposition 2.6 are satisfied. While this is one of the simplest mechanisms, many other mechanisms can also be represented in this way.

For example, Table 1 shows how RAPPOR [18], Subset Selection [42] and Hadamard [2] can be expressed as a strategy matrix. Other mechanisms with more sophisticated structure, such as Hierarchical [13, 39] and Fourier [12], can also be expressed as a strategy matrix, but they require too much notation to explain in Table 1. A strategy matrix is simply a direct encoding of a conditional probability distribution, where every probability is explicitly enumerated as an entry of the matrix. Hence, the representation can encode *any* LDP mechanism, as long as the  $\mathcal{U}$  and  $\mathcal{O}$  are both finite and the conditional probabilities can be calculated.

When executing the mechanism, each user reports a (randomized) response  $o_i = \mathcal{M}_{\mathbf{Q}}(u_i)$ . When all users randomize their data with the same mechanism, these responses are typically aggregated into a response vector  $\mathbf{y} \in \mathbb{R}^m$  (indexed by elements  $o \in \mathcal{O}$ ), where  $y_o = \sum_{j=1}^N \mathbb{1}\{o_j = o\}$ . Much like the data vector, the response vector is essentially a histogram of responses, as it counts the number of users who reported each response. In the sequel, it is useful to think of the mechanism  $\mathcal{M}_{\mathbf{Q}}$  as being a function from  $\mathbf{x}$  to  $\mathbf{y}$  instead of  $u_i$  to  $o_i$ . Thus, for notational convenience, we overload the definition of  $\mathcal{M}_{\mathbf{Q}}$ , allowing it to consume a data vector and return a response vector, so that  $\mathcal{M}_{\mathbf{Q}}(\mathbf{x}) = \mathbf{y}$ .

<sup>1</sup>the name of this mechanism should not be confused with the outputs of an arbitrary mechanism  $\mathcal{M}$ , which we also call randomized responses.

The response vector  $\mathbf{y}$  is often not that useful by itself, but it can be used to estimate more useful quantities, such as the data vector  $\mathbf{x}$  or the workload answers  $\mathbf{W}\mathbf{x}$ . This is typically achieved with a post-processing step, and does not impact the privacy guarantee of the mechanism.

### 3. THE FACTORIZATION MECHANISM

In this section, we describe our mechanism and the main optimization problem that underlies it. We begin with a high-level problem statement, and reason about it analytically until it is in a form we can deal with algorithmically. We present our key ideas and the main steps of our derivation, but defer the finer details of the proofs to Appendix A.

Our goal is to find a mechanism that has low expected error on the workload. This objective is formalized in Problem 3.1.

**Problem 3.1 (Workload Error Minimization)** *Given a workload  $\mathbf{W}$ , design an  $\epsilon$ -LDP mechanism  $\mathcal{M}^*$  that minimizes worst-case expected  $L_2$  squared error. Formally,*

$$\mathcal{M}^* = \arg \min_{\mathcal{M}} \left\{ \max_{\mathbf{x}} \mathbb{E}[\|\mathbf{W}\mathbf{x} - \mathcal{M}(\mathbf{x})\|_2^2] \right\}$$

In the problem statement above, our goal is to search through the space of *all*  $\epsilon$ -LDP mechanisms for the one that is best for the given workload. Because it is difficult to characterize an arbitrary mechanism  $\mathcal{M}$  in a way that makes optimization possible, we do not solve the above problem in its full generality. Instead, we perform the search over a restricted class of mechanisms which is easier to characterize. While somewhat restricted, this class of mechanisms is quite expressive, and it captures many of the state-of-the-art LDP mechanisms available today [18, 42, 2, 12, 13, 39].

**Definition 3.2 (Workload Factorization Mechanism)** *Given an  $\epsilon$ -LDP strategy matrix  $\mathbf{Q} \in \mathbb{R}^{m \times n}$  and a reconstruction matrix  $\mathbf{V} \in \mathbb{R}^{p \times m}$  such that  $\mathbf{W} = \mathbf{V}\mathbf{Q}$ , the Workload Factorization Mechanism (factorization mechanism for short) is defined as:*

$$\mathcal{M}_{\mathbf{V}, \mathbf{Q}}(\mathbf{x}) = \mathbf{V}\mathcal{M}_{\mathbf{Q}}(\mathbf{x})$$

Note that  $\mathcal{M}_{\mathbf{V}, \mathbf{Q}}$  is defined in terms of  $\mathcal{M}_{\mathbf{Q}}$ , and it is parameterized by an additional *reconstruction matrix*  $\mathbf{V}$  as well. This reconstruction matrix is used to estimate the workload query answers from the response vector output by  $\mathcal{M}_{\mathbf{Q}}$ . In fact, the workload query estimates produced by this class of mechanisms is unbiased, as:

$$\mathbb{E}[\mathcal{M}_{\mathbf{V}, \mathbf{Q}}(\mathbf{x})] = \mathbf{V}\mathbb{E}[\mathcal{M}_{\mathbf{Q}}(\mathbf{x})] = \mathbf{V}\mathbf{Q}\mathbf{x} = \mathbf{W}\mathbf{x}.$$

Furthermore,  $\mathcal{M}_{\mathbf{V}, \mathbf{Q}}$  inherits the privacy guarantee of  $\mathcal{M}_{\mathbf{Q}}$  by the post-processing principle of differential privacy [16].

Many existing LDP mechanisms can be represented as a factorization mechanism. For example, we show how the Randomized Response mechanism can be expressed as a factorization mechanism in Example 3.3.

**Example 3.3 (Randomized Response)** *The randomized response mechanism uses  $\mathbf{Q}$  as defined in Example 2.7 and  $\mathbf{V} = \mathbf{Q}^{-1}$  to estimate the Histogram workload ( $\mathbf{W} = \mathbf{I}$ ).*

$$\mathbf{V} = \frac{1}{e^\epsilon - 1} \begin{bmatrix} e^\epsilon + n - 2 & -1 & \dots & -1 \\ -1 & e^\epsilon + n - 2 & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \dots & e^\epsilon + n - 2 \end{bmatrix}$$

While the randomized response mechanism is intended to be used to answer the Histogram workload, there is no reason why it cannot be used for other workloads as well. In fact, it is quite straightforward to see how it can be extended to answer an arbitrary workload, simply by using  $\mathbf{V} = \mathbf{W}\mathbf{Q}^{-1}$ .

### 3.1 Variance Derivation

While the factorization mechanism is unbiased for any workload factorization, different factorizations lead to different amounts of variance on the workload answers. This creates the opportunity to choose the workload factorization that leads to the lowest possible total variance. In order to do that, we need an analytic expression for the total variance in terms of  $\mathbf{V}$  and  $\mathbf{Q}$ , which we derive in Theorem 3.4.

**Theorem 3.4 (Variance)** *The expected total squared error (total variance) of a workload factorization mechanism is:*

$$\mathbb{E}[\|\mathbf{W}\mathbf{x} - \mathcal{M}_{\mathbf{V}, \mathbf{Q}}(\mathbf{x})\|_2^2] = \sum_{u \in \mathcal{U}} x_u \sum_{i=1}^p \mathbf{v}_i^T \text{Diag}(\mathbf{q}_u) \mathbf{v}_i - (\mathbf{v}_i^T \mathbf{q}_u)^2$$

where  $\mathbf{q}_u$  denotes column  $u$  of  $\mathbf{Q}$  and  $\mathbf{v}_i^T$  denotes row  $i$  of  $\mathbf{V}$ .

Notice above that the exact expression for variance depends on the data vector  $\mathbf{x}$ , which we do not have access to, as it is a private quantity. We want our mechanism to work well for all possible  $\mathbf{x}$ , so we consider *worst-case* variance and a relaxation *average-case* variance instead.<sup>2</sup>

**Corollary 3.5 (Worst-case variance)** *The worst-case variance of  $\mathcal{M}_{\mathbf{V}, \mathbf{Q}}$  occurs when all users have the same worst case type (i.e.,  $x_u = N$  for some  $u$ ), and is:*

$$L_{\text{worst}}(\mathbf{V}, \mathbf{Q}) = N \max_{u \in \mathcal{U}} \sum_{i=1}^p \mathbf{v}_i^T \text{Diag}(\mathbf{q}_u) \mathbf{v}_i - (\mathbf{v}_i^T \mathbf{q}_u)^2.$$

**Corollary 3.6 (Average-case variance)** *The average-case variance of  $\mathcal{M}_{\mathbf{V}, \mathbf{Q}}$  occurs when all user types have the same count (i.e.,  $x_u = \frac{N}{n}$  for all  $u$ ), and is:*

$$L_{\text{avg}}(\mathbf{V}, \mathbf{Q}) = \frac{N}{n} \sum_{u \in \mathcal{U}} \sum_{i=1}^p \mathbf{v}_i^T \text{Diag}(\mathbf{q}_u) \mathbf{v}_i - (\mathbf{v}_i^T \mathbf{q}_u)^2.$$

With these analytic expressions for variance, we can analyze and compare existing mechanisms that can be expressed as a workload factorization mechanism. The variance for randomized response is shown in Example 3.7.

<sup>2</sup>Alternatively, if we had a prior distribution over  $\mathbf{x}$ , we could use that to estimate variance.

### Example 3.7 (Variance of Randomized Response)

*The worst-case and average-case variance of the factorization in Example 3.3 on the Histogram workload is:*

$$L_{\text{worst}}(\mathbf{V}, \mathbf{Q}) = L_{\text{avg}}(\mathbf{V}, \mathbf{Q}) = N(n-1) \left[ \frac{n}{(e^\epsilon - 1)^2} + \frac{2}{e^\epsilon - 1} \right]$$

The expression above is obtained by simply plugging in  $\mathbf{V}$  and  $\mathbf{Q}$  to the equations above and simplifying. Interestingly, the worst-case and average-case variance are the same for this workload factorization due to the symmetry in the workload and strategy matrices.

### 3.2 Strategy Optimization

With an analytic expression for variance, we can state the optimization problem underlying the factorization mechanism. Our goal is to find a workload factorization that minimizes the total variance on the workload. To do that, we set up an optimization problem, using total variance as the objective function while taking into consideration the constraints that have to hold on  $\mathbf{V}$  and  $\mathbf{Q}$ . This is formalized in Problem 3.8.

**Problem 3.8 (Optimal Factorization)** *Given a privacy budget  $\epsilon$  and a workload  $\mathbf{W}$ ,*

$$\begin{aligned} & \underset{\mathbf{V}, \mathbf{Q}}{\text{minimize}} && L(\mathbf{V}, \mathbf{Q}) \\ & \text{subject to} && \mathbf{W} = \mathbf{V}\mathbf{Q} \\ & && \sum_o Q_{o,u} = 1 \quad \forall u \\ & && 0 \leq Q_{o,u} \leq \exp(\epsilon) Q_{o,u'} \quad \forall o, u, u'. \end{aligned}$$

Above,  $L$  is a loss function that captures how good a given factorization is, such as the worst-case variance  $L_{\text{worst}}$  or the average-case variance  $L_{\text{avg}}$ . While our original objective in Problem 3.1 was to find the mechanism that optimizes worst-case variance, for practical reasons we use the average-case variance instead. The average-case variance is a smooth approximation of the worst-case variance, which leads to a more analytically and computationally tractable problem. Additionally, the smoothness of the average-case variance makes the corresponding optimization problem more amenable to numerical methods. We study the ramifications of this relaxation theoretically in Section 5.1. When using  $L_{\text{avg}}$  as the objective function, we observe it can be expressed in a much simpler form using matrix operations, as shown in Theorem 3.9.

**Theorem 3.9 (Objective Function)** *The objective function  $L(\mathbf{V}, \mathbf{Q}) = \text{tr}[\mathbf{V}\mathbf{D}_\mathbf{Q}\mathbf{V}^T]$  is related to  $L_{\text{avg}}(\mathbf{V}, \mathbf{Q})$  by:*

$$L_{\text{avg}}(\mathbf{V}, \mathbf{Q}) = \frac{N}{n} (L(\mathbf{V}, \mathbf{Q}) - \|\mathbf{W}\|_F^2),$$

where  $\mathbf{D}_\mathbf{Q} = \text{Diag}(\mathbf{Q}\mathbf{1})$  and  $\text{tr}[\cdot]$  is the trace of a matrix.

From now on, when we refer to  $L(\mathbf{V}, \mathbf{Q})$ , we are referring to its definition in Theorem 3.9. The new objective is equivalent to  $L_{\text{avg}}$  up to constant factors, and hence can be used in place of it for the purposes of optimization.

With this simplified objective function, we observe that for a fixed strategy matrix  $\mathbf{Q}$ , we can compute the optimal  $\mathbf{V}$  in closed form. If the entries of the response vector were statistically independent and had equal variance, then this would simply be  $\mathbf{V} = \mathbf{W}\mathbf{Q}^\dagger$ , where  $\mathbf{Q}^\dagger$  is the Moore-Penrose pseudo-inverse of  $\mathbf{Q}$  [11, 23]. However, since the entries

of the response vector have unequal variance and are not statistically independent in general, this simple expression is not correct. We can still express the optimal  $\mathbf{V}$  in closed form, however, as shown in Theorem 3.10.

**Theorem 3.10 (Optimal  $\mathbf{V}$  for fixed  $\mathbf{Q}$ )** *For a fixed  $\mathbf{Q}$ , the minimizer of  $L(\mathbf{V}, \mathbf{Q})$  subject to  $\mathbf{W} = \mathbf{V}\mathbf{Q}$  is given by:*

$$\mathbf{V} = \mathbf{W}(\mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q})^\dagger \mathbf{Q}^T \mathbf{D}_Q^{-1},$$

Note that we can assume  $\mathbf{D}_Q$  is invertible without loss of generality. If it were not, then one entry of the diagonal would have to be 0, implying that a row of  $\mathbf{Q}$  is all zero. Such a row corresponds to an output that never occurs under the privacy mechanism, and can be removed without changing the mechanism. Further note that for the above formula to apply, there must exist a  $\mathbf{V}$  such that  $\mathbf{W} = \mathbf{V}\mathbf{Q}$ , which is guaranteed if and only if  $\mathbf{W}$  is in the row space of  $\mathbf{Q}$  [35]. Expressed as a constraint, this is  $\mathbf{W} = \mathbf{W}\mathbf{Q}^\dagger \mathbf{Q}$ .

Now that we know the optimal  $\mathbf{V}$  for any  $\mathbf{Q}$ , we can plug it into  $L(\mathbf{V}, \mathbf{Q})$  to express the objective as a function of  $\mathbf{Q}$  alone. Doing this, and simplifying further, leads to our final optimization objective, stated in Theorem 3.11.

**Theorem 3.11 (Objective Function for  $\mathbf{Q}$ )** *The objective function can be expressed as:*

$$L(\mathbf{Q}) = \text{tr}[(\mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q})^\dagger (\mathbf{W}^T \mathbf{W})].$$

$L(\mathbf{Q})$  is our final optimization objective. We are almost ready to restate the optimization problem in terms of  $\mathbf{Q}$ . However, before we do that, it is useful to simplify the constraints of the problem. The constraints stated in Problem 3.8 are challenging to deal with algorithmically because there are a large number of them. Ignoring the factorization constraint, there are  $n^2 m + n$  constraints on  $\mathbf{Q}$ , and each entry of  $\mathbf{Q}$  is constrained by entries from the same column and entries from the same row.

By introducing an auxiliary optimization variable  $\mathbf{z} \in \mathbb{R}^m$ , we reduce this to  $nm + n$  constraints, so that each entry of  $\mathbf{Q}$  is only constrained by entries from the same column and  $\mathbf{z}$ . Specifically,  $\mathbf{z}$  corresponds to the minimum allowable value on each row of  $\mathbf{Q}$ , and every column of  $\mathbf{Q}$  must be between  $\mathbf{z}$  and  $\exp(\epsilon)\mathbf{z}$  (coordinate-wise inequality). It is clear that this is exactly equivalent to Condition 2 in Proposition 2.6. Also note that Condition 1 can be expressed in matrix form as  $\mathbf{Q}^T \mathbf{1} = \mathbf{1}$ . The final optimization problem underlying the workload factorization mechanism is stated in Problem 3.12.

**Problem 3.12 (Strategy Optimization)** *Given a workload  $\mathbf{W}$  and a privacy budget  $\epsilon$ :*

$$\begin{aligned} \underset{\mathbf{Q}, \mathbf{z}}{\text{minimize}} \quad & \text{tr}[(\mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q})^\dagger (\mathbf{W}^T \mathbf{W})] \\ \text{subject to} \quad & \mathbf{W} = \mathbf{W}\mathbf{Q}^\dagger \mathbf{Q} \\ & \mathbf{Q}^T \mathbf{1} = \mathbf{1} \\ & \mathbf{0} \leq \mathbf{z} \leq \mathbf{q}_u \leq \exp(\epsilon)\mathbf{z} \quad \forall u. \end{aligned}$$

## 4. OPTIMIZATION ALGORITHM

We now discuss our approach to solving Problem 3.12. It is a nonlinear optimization problem with linear and nonlinear constraints. While the objective is smooth (and hence differentiable) within the boundary of the constraint  $\mathbf{W} = \mathbf{W}\mathbf{Q}^\dagger \mathbf{Q}$ , it is not convex. It is typically infeasible to find a closed

---

### Algorithm 1 Projection onto bounded probability simplex

---

**Input:**  $\mathbf{r}, \mathbf{z} \in \mathbb{R}^m, \epsilon$   
 $\triangleright$  sorted vector and corresponding permutation  
 $\mathbf{u}, \boldsymbol{\pi} = \text{sort}([\mathbf{z} - \mathbf{r}, \exp(\epsilon)\mathbf{z} - \mathbf{r}])$   
 $\mathbf{a} = [(-1)^{\mathbb{1}_{\{\pi_i \leq m\}}}]_{i=1 \dots 2m}$   
 $\mathbf{b} = [\sum_{j=1}^{i-1} a_j]_{i=1 \dots 2m}$   
 $\rho = \min\{1 \leq i \leq 2m : \mathbf{1}^T \mathbf{z} + b_i u_i + \sum_{j=1}^{i-1} a_j u_j > 1\}$   
 $\lambda = (1 - \mathbf{1}^T \mathbf{z} - b_\rho u_\rho - \sum_{j=1}^{\rho-1} a_j u_j) / b_\rho + u_\rho$   
**return**  $\text{clip}(\mathbf{r} + \lambda, \mathbf{z}, \exp(\epsilon)\mathbf{z})$

---

form solution to such a problem, and conventional numerical optimization methods are not guaranteed to converge to a global minimum for non-convex objectives. However, such numerical gradient-based methods have seen remarkable empirical success in a variety of domains, often finding high quality local minima. That is the approach we take, however, rather than use an out-of-the-box commercial solver, which would not be able to scale to larger problem sizes, we provide our own optimization algorithm which achieves greater scalability by exploiting structure in the constraint set.

The algorithm we propose is an instance of *projected gradient descent* [34], a variant of gradient descent that handles constraints. To implement this algorithm, the key challenge is to *project* onto the constraint set. In other words, given a matrix  $\mathbf{R}$  that does not satisfy the constraints, find the “closest” matrix  $\mathbf{Q}$  that does satisfy the constraints. Ignoring the constraint  $\mathbf{W} = \mathbf{W}\mathbf{Q}^\dagger \mathbf{Q}$  for now, this sub-problem is stated formally in Problem 4.1.

#### Problem 4.1 (Projection onto LDP Constraints)

*Given an arbitrary matrix  $\mathbf{R}$ , a vector  $\mathbf{z}$ , and a privacy budget  $\epsilon$ , the projection onto the privacy constraint, denoted  $\mathbf{Q} = \Pi_{\mathbf{z}, \epsilon}(\mathbf{R})$  is obtained by solving the following problem:*

$$\begin{aligned} \underset{\mathbf{Q}}{\text{minimize}} \quad & \|\mathbf{Q} - \mathbf{R}\|_F^2 \\ \text{subject to} \quad & \mathbf{Q}^T \mathbf{1} = \mathbf{1} \\ & \mathbf{z} \leq \mathbf{q}_u \leq \exp(\epsilon)\mathbf{z} \quad \forall u. \end{aligned}$$

Problem 4.1 is easier to solve than Problem 3.12 because the objective is now a quadratic function of  $\mathbf{Q}$ . In addition, a key insight to solve this problem efficiently is to notice that it is closely related to the problem of projecting onto the probability simplex [40] (now with bound constraints), and admits a similar solution. Specifically, the form of the solution is stated in Proposition 4.2.

**Proposition 4.2 (Projection Algorithm)** *The solution to Problem 4.1 may be obtained one column at a time using*

$$\mathbf{q}_u = \text{clip}(\mathbf{r}_u + \lambda_u, \mathbf{z}, \exp(\epsilon)\mathbf{z}),$$

where  $\text{clip}$  clips the entries of  $\mathbf{r}_u + \lambda_u$  to the range  $[\mathbf{z}, \exp(\epsilon)\mathbf{z}]$  entry-wise and  $\lambda_u$  is a scalar value that makes  $\mathbf{1}^T \mathbf{q}_u = 1$ .

The solution is remarkably simple. Intuitively, we add the same scalar value to every entry of  $\mathbf{r}_u$  then clip those values that lie outside the allowed range. The scalar value  $\lambda_u$  is the Lagrange multiplier on the constraint  $\mathbf{1}^T \mathbf{q}_u = 1$ , and is chosen so that  $\mathbf{q}_u$  satisfies the constraint. It may be calculated through binary search or any other method to find the root of the function  $\mathbf{1}^T \mathbf{q}_u - 1 = 0$ . We give an efficient  $O(m \log(m))$  algorithm to find  $\lambda_u$  and  $\mathbf{q}_u$  in Algorithm 1.

---

**Algorithm 2** Strategy optimization

---

**Input:** Workload  $\mathbf{W} \in \mathbb{R}^{p \times n}$ , privacy budget  $\epsilon$   
**Initialize**  $\mathbf{Q} \in \mathbb{R}^{m \times n}$ ,  $\mathbf{z} \in \mathbb{R}^m$ ,  $\beta \in \mathbb{R}_+$   
 $\alpha = \beta/n \exp(\epsilon)$   
**for**  $t = 1, \dots, T$  **do**  
     $\mathbf{z} \leftarrow \text{clip}(\mathbf{z} - \alpha \nabla_{\mathbf{z}} L(\mathbf{Q}), \mathbf{0}, \mathbf{1})$   
     $\mathbf{Q} \leftarrow \Pi_{\mathbf{z}, \epsilon}(\mathbf{Q} - \beta \nabla_{\mathbf{Q}} L(\mathbf{Q}))$   
**end for**  
**return**  $\mathbf{Q}$

---

Now that we have discussed the projection problem and its solution, we are ready to state the full projected gradient descent algorithm for finding an optimized strategy. Algorithm 2 is an iterative algorithm, where in each iteration we perform a gradient descent plus projection step on the optimization parameters  $\mathbf{z}$  and  $\mathbf{Q}$ . The gradient  $\nabla_{\mathbf{Q}} L$  is easily obtained as  $L$  is a function of  $\mathbf{Q}$ , but the gradient term  $\nabla_{\mathbf{z}} L$  is less obvious. However, by observing that  $\mathbf{Q}$  is actually a function of  $\mathbf{z}$  (from the projection step  $\Pi_{\mathbf{z}, \epsilon}$ ), we can use the multi-variate chain rule to back-propagate the gradient from  $\mathbf{Q}$  to  $\mathbf{z}$  to obtain  $\nabla_{\mathbf{z}} L$ . We do not discuss the details of computing the gradients here, as it can be easily accomplished with automatic differentiation tools [20, 31].

We note that Algorithm 2 handles the constraint  $\mathbf{W} = \mathbf{W}\mathbf{Q}^\dagger\mathbf{Q}$  “for free” in the sense that we do not need to deal with it explicitly, as long as the step sizes and initialization are chosen appropriately. Specifically, as long as the initial  $\mathbf{Q}$  satisfies the constraint, and the step sizes are sufficiently small, every subsequent  $\mathbf{Q}$  in the algorithm will also satisfy the constraint. Intuitively, this is because as we move closer to the boundary of the constraint, the objective function blows up and eventually reaches a point of discontinuity when the constraint is not satisfied. Because we update using the negative gradient, which is a descent direction, we will never approach the boundary of the constraint. We discuss the choice of step size and initialization below. This trick is a very nice way to deal with a constraint that is otherwise challenging to deal with. We note that similar ideas have been used to deal with related constraints in prior work [43].

The step size for the gradient descent step must be supplied as input, and two different step sizes are used to update  $\mathbf{Q}$  and  $\mathbf{z}$ . Notice that we take a smaller step size to update  $\mathbf{z}$  than  $\mathbf{Q}$ . This is a heuristic we use to make sure  $\mathbf{z}$  doesn’t change too fast; it improves the robustness of the algorithm. We perform a hyper-parameter search to find a step size that works well, only running the algorithm for a few iterations in this phase, then running it longer once a step size is chosen. Decaying the step size at each iteration is also possible, as smaller step sizes typically work better in later iterations.

The final missing piece in Algorithm 2 is the initialization of  $\mathbf{Q}$ , for which there are multiple options. One option is to initialize with the strategy matrix from an existing mechanism, such as the best one from Table 1. Then intuitively the optimized strategy will never be worse than the other mechanisms, because the negative gradient is a descent direction. This is an informal argument, as it is technically possible that the optimized strategy has better average-case variance but worse worst-case variance than the initial strategy. We do not take this approach, however, as we find initializing  $\mathbf{Q}$  randomly tends to work better. Specifically, we let  $\mathbf{R}$  be a random  $4n \times n$  matrix, where each entry is sampled from

$U[0, 1]$ . Then we obtain  $\mathbf{Q}$  by projecting onto the constraint set; i.e.,  $\mathbf{Q} = \Pi_{\mathbf{z}, \epsilon}(\mathbf{R})$ , where  $\mathbf{z} = \frac{1+\epsilon-\epsilon}{8n}\mathbf{1}$ , where  $\mathbf{1}$  is a vector of ones. The choice of  $m$  is also an important consideration when initializing  $\mathbf{Q}$ . While larger  $m$  leads to a more expressive strategy space, it also leads to more expensive optimization. Our choice of  $m = 4n$  represents a sweet spot that we found works well empirically across a variety of workloads. In general, a hyper-parameter search can be executed to find the best  $m$ . This hyper-parameter search does not degrade the privacy guarantee in any way because we can evaluate the quality of a strategy without consuming the privacy budget, by using the analytic formulas for variance.

It requires  $O(n^2m + n^3)$  time to evaluate the objective function and its gradient (assuming  $\mathbf{W}^T\mathbf{W}$  has been pre-computed) and  $O(nm \log m)$  time to perform the projection. Thus, the per-iteration time complexity of Algorithm 2 is  $O(n^2m + n^3 + nm \log m)$ , or  $O(n^3)$  when using  $m = 4n$ .

## 5. THEORETICAL RESULTS

In this section, we answer several theoretical questions about our mechanism. First, we justify the relaxation in the objective function, used to make the optimization analytically tractable. Second, we theoretically analyze the error achieved by our mechanism, measured in terms of sample complexity. Third, we derive lower bounds on the achievable error of workload factorization mechanisms. All the proofs are deferred to Appendix A.

### 5.1 Relaxed Objective

In Section 3 we replaced our true optimization objective  $L_{\text{worst}}$  with a relaxation  $L_{\text{avg}}$ . In this section, we justify that choice theoretically, showing that  $L_{\text{worst}}$  is tightly bounded above and below by  $L_{\text{avg}}$ .

**Theorem 5.1 (Bounds on  $L_{\text{worst}}$ )** *Let  $\mathbf{W} = \mathbf{V}\mathbf{Q}$  be an arbitrary factorization of  $\mathbf{W}$  where  $\mathbf{Q}$  is an  $\epsilon$ -LDP strategy matrix. Then the worst case variance  $L_{\text{worst}}(\mathbf{V}, \mathbf{Q})$  and average-case variance  $L_{\text{avg}}(\mathbf{V}, \mathbf{Q})$  are related as follows:*

$$L_{\text{avg}}(\mathbf{V}, \mathbf{Q}) \leq L_{\text{worst}}(\mathbf{V}, \mathbf{Q}) \leq e^\epsilon (L_{\text{avg}}(\mathbf{V}, \mathbf{Q}) + \frac{N}{n} \|\mathbf{W}\|_F^2)$$

Theorem 5.1 suggests that relaxing  $L_{\text{worst}}$  to  $L_{\text{avg}}$  does not significantly impact the optimization problem. Intuitively, this theorem holds because of the privacy constraint on  $\mathbf{Q}$ , which guarantees that the column of  $\mathbf{Q}$  for the worst-case user cannot be too different from any other column. Hence, all users must have a similar impact on the total variance of the mechanism. Empirically, we find that  $L_{\text{worst}}$  is often even closer to  $L_{\text{avg}}$  than the upper bound suggests. Furthermore, in some cases  $L_{\text{worst}}$  is exactly equal to  $L_{\text{avg}}$ , as we showed in Example 3.7.

### 5.2 Sample complexity

We gave an analytic expression for the expected total squared error (total variance) of our mechanism in Corollary 3.5. However, this quantity might be difficult to interpret, and it is more natural to look at the number of samples needed to achieve a fixed error instead. Furthermore, when running an LDP mechanism it is important to know how much data is required to obtain a target error rate, as that information is critical for determining an appropriate privacy budget.

Because the total variance increases with the number of individuals  $N$  and the number of workload queries  $m$ , we instead look at a normalized measure of variance.

**Definition 5.2 (Normalized Variance)** *The normalized worst-case variance of  $\mathcal{M}_{\mathbf{V}, \mathbf{Q}}$  is:*

$$L_{norm}(\mathbf{V}, \mathbf{Q}) = \max_{\mathbf{x}} \mathbb{E} \left[ \frac{1}{m} \left\| \frac{1}{N} \{ \mathbf{W}\mathbf{x} - \mathcal{M}_{\mathbf{V}, \mathbf{Q}}(\mathbf{x}) \} \right\|_2^2 \right]$$

$L_{norm}$  is the same as  $L_{worst}$  up to constant factors, although it is more interpretable because it is a measure of variance on a single “average” workload query, where variance is measured on the normalized data vector  $\mathbf{x}/N$ .

**Corollary 5.3 (Normalized Variance)** *The normalized variance is:*

$$\begin{aligned} L_{norm}(\mathbf{V}, \mathbf{Q}) &= \frac{1}{mN^2} L_{worst}(\mathbf{V}, \mathbf{Q}) \\ &= \frac{1}{mN} \max_{u \in \mathcal{U}} \sum_{i=1}^p \mathbf{v}_i^T \text{Diag}(\mathbf{q}_u) \mathbf{v}_i - (\mathbf{v}_i^T \mathbf{q}_u)^2 \end{aligned}$$

Interestingly, the dependence on  $N$  does not change with  $\mathbf{V}$  and  $\mathbf{Q}$  — it is always  $\Theta(\frac{1}{N})$ , but the constant factor depends on the quality of the workload factorization.

**Corollary 5.4 (Sample Complexity)** *The number of samples needed to achieve normalized variance  $\alpha$  is:*

$$N \geq \frac{1}{m\alpha} \max_{u \in \mathcal{U}} \sum_{i=1}^p \mathbf{v}_i^T \text{Diag}(\mathbf{q}_u) \mathbf{v}_i - (\mathbf{v}_i^T \mathbf{q}_u)^2$$

We can readily compute the sample complexity numerically for any factorization  $\mathbf{V}\mathbf{Q}$ . In fact, the sample complexity and worst-case variance of a mechanism are proportional, as evident from the above equation. Additionally, by replacing  $L_{worst}(\mathbf{V}, \mathbf{Q})$  with a lower bound, we can get an analytical expression for the sample complexity in terms of the privacy budget  $\epsilon$  and the properties of the workload  $\mathbf{W}$ .

**Example 5.5 (Sample complexity, Randomized Response)** *The Randomized Response mechanism described in Example 3.3 has sample complexity:*

$$N \geq \frac{(n-1)}{\alpha n} \left[ \frac{n}{(e^\epsilon - 1)^2} + \frac{2}{e^\epsilon - 1} \right]$$

on the Histogram workload.

Example 5.5 suggests the sample complexity of the randomized response mechanism grows roughly at a linear rate with the domain size  $n$ .

### 5.3 Lower Bound

For a given workload, a theoretical lower bound on the achievable error is useful for checking how close to optimal our strategies are. It also can be used to characterize the inherent difficulty of the workload. In this section, we derive an easily-computable lower bound on the achievable error under our mechanism in terms of the *singular values* of the workload matrix.

**Theorem 5.6 (Lower Bound, Factorization Mechanism)** *Let  $\mathbf{W}$  be a workload matrix and let  $\mathbf{Q}$  be any  $\epsilon$ -LDP strategy matrix. Then we have:*

$$\frac{1}{\exp(\epsilon)} (\lambda_1 + \dots + \lambda_n)^2 \leq L(\mathbf{Q})$$

where  $\lambda_1, \dots, \lambda_n$  are the singular values of  $\mathbf{W}$  and  $L(\mathbf{Q})$  is the loss function defined in Theorem 3.11.

This result is similar to lower bounds known to hold in the central model of differential privacy, based on the analysis of the Matrix Mechanism [29]. In both cases the hardness of a workload is characterized by its singular values.

Other lower bounds for this problem have characterized the hardness of a workload in terms of quantities like the largest  $L_2$  column norm of  $\mathbf{W}$  [4], the so-called factorization norm of  $\mathbf{W}$  [17], and the so-called packing number associated with  $\mathbf{W}$  [9]. While interesting theoretically, the factorization norm and packing number are hard to calculate in practice. In contrast, our bound can be easily calculated.

Theorem 5.6 gives a lower bound on our optimization objective. Translating that back to worst-case variance gives us Corollary 5.7.

**Corollary 5.7 (Worst-case variance)** *The worst-case variance of any factorization mechanism must be at least:*

$$\frac{N}{n \exp(\epsilon)} (\lambda_1 + \dots + \lambda_n)^2 - \frac{N}{n} \|\mathbf{W}\|_F^2 \leq L_{worst}(\mathbf{V}, \mathbf{Q})$$

Combining Corollary 5.7 with Corollary 5.4 and applying it to the Histogram workload gives us a lower bound on the sample complexity.

**Example 5.8 (Lower Bound, Histogram Workload)** *Every workload factorization mechanism requires at least  $\frac{1}{\alpha} \left( \frac{1}{\exp(\epsilon)} - \frac{1}{n} \right)$  samples to achieve normalized variance  $\alpha$  on the Histogram workload.*

Note the very weak dependence on  $n$  in Example 5.8, which suggests that the sample complexity should not change much with  $n$ . Further, recall from Example 5.5 that the sample complexity of randomized response is linear in  $n$ . This suggests randomized response is not the best mechanism for the Histogram workload. This result is not new, as there are several mechanisms that are known to perform better than randomized response [1, 42, 38, 18]. We show empirically in Section 6 that some of these mechanisms achieve the optimal sample complexity for the Histogram workload up to constant factors (i.e., no dependence on  $n$ ). Our mechanism also achieves the optimal sample complexity for this workload, but has better constant factors.

For other workloads, the sample complexity may depend on  $n$ . Calculating the exact dependence on  $n$  for other workloads requires deriving the singular values of the workload as a function of  $n$  in closed form, which may be challenging for workloads with complicated structure.

## 6. EXPERIMENTS

In this section we experimentally evaluate our mechanism. We extensively study the utility of our mechanism on a variety of workloads, domains, and privacy levels, and compare it against multiple competing mechanisms from the literature. We demonstrate consistent improvements in utility compared to other mechanisms in all settings (Section 6.2 and Section 6.3 and Section 6.4). We also study the robustness and scalability of our optimization algorithm (Section 6.5 and Section 6.6). The source code for our mechanism, and other mechanisms represented as a strategy matrix, is available at <https://github.com/ryan112358/workload-factorization-mechanism>.

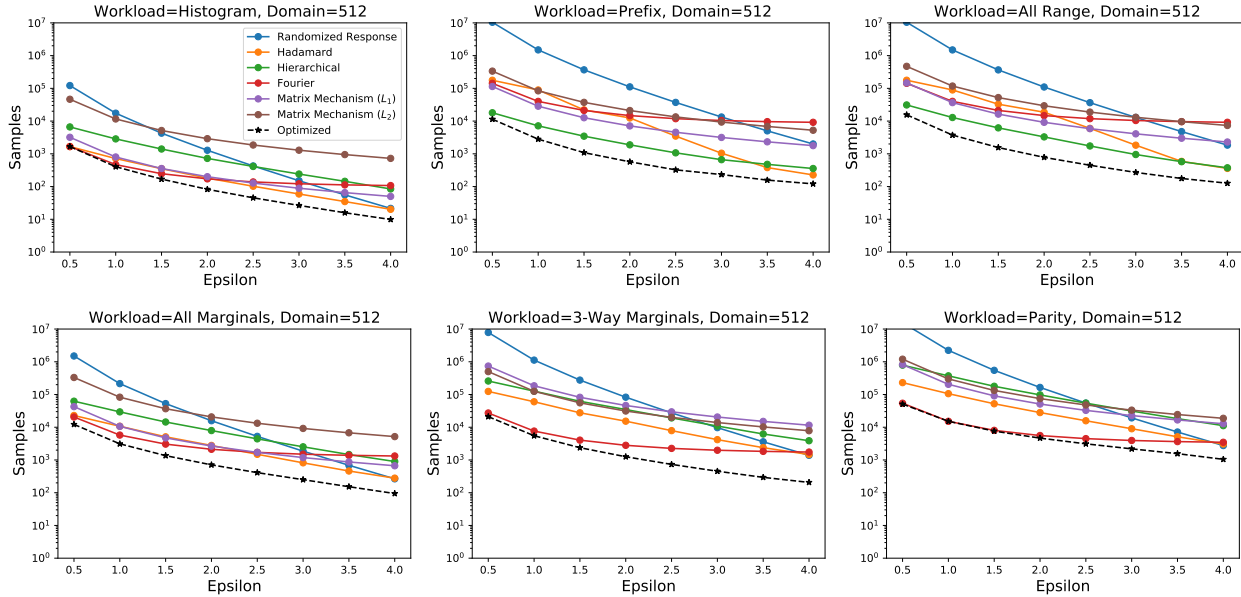


Figure 1: Sample complexity of 7 algorithms on 6 workloads for  $\epsilon \in [0.5, 4.0]$

## 6.1 Experimental setup

**Workloads.** We consider six different workloads in our empirical analysis, each of which can be defined for a specified domain size. These workloads are intended to capture common queries an analyst might want to perform on data and have been studied previously in the privacy literature. These workloads are Histogram, Prefix, All Range, All Marginals, 3-Way Marginals, and Parity. Histogram is the simplest workload, studied as a running example throughout the paper, and encoded as an identity matrix. Prefix was introduced in Example 2.4, and includes a set of range queries required to compute the empirical CDF over a 1-dimensional domain. All Range is a workload containing all range queries over a 1-dimensional domain, studied in [12]. All Marginals and 3-Way Marginals contain queries to compute the marginals over a multidimensional binary domain, and were studied in [13]. Parity also contains queries defined over a multidimensional binary domain, and was studied in [19].

**Mechanisms.** We compare our mechanism against six other state-of-the-art mechanisms, including Randomized Response [41], Hadamard [2], Hierarchical [13, 39], Fourier [12], and the Matrix Mechanism [27, 17] (both  $L_1$  and  $L_2$  versions). While the Matrix Mechanism is typically thought of as a technique for central differential privacy, it has been studied theoretically as a mechanism for local differential privacy as well [17]. This version of the “distributed” Matrix Mechanism is what we compare against in experiments.

The first four mechanisms are all particular instances of the class of factorization mechanisms, just with different factorizations. They were all designed to answer a fixed workload (e.g., Randomized Response was designed for the Histogram workload), but they can still be run on other workloads with minor modifications. In particular, for each mechanism we use the same  $\mathbf{Q}$  across different workloads, but change  $\mathbf{V}$  based on the workload, using Theorem 3.10.

We omit from comparison the Gaussian mechanism [4], as it is strictly dominated by the  $L_2$  Matrix Mechanism. We also omit from comparison RAPPOR [18] and Subset Selection [42], as they require exponential space to represent the strategy matrix, making it prohibitive to calculate worst-case variance and sample complexity. However, we note that these mechanisms have been previously compared with Hadamard, and shown to offer comparable performance on the Histogram workload [2].

**Evaluation.** Our primary evaluation metric for comparing algorithms is *sample complexity*, which we calculate exactly using Corollary 5.4 with  $\alpha = 0.01$ . Recall that the sample complexity is proportional to the worst-case variance, but is appropriately normalized and easier to interpret. Furthermore, we remark that for most experiments in this section, no input data is required, as the sample complexities we report apply for the worst-case dataset. In practice, we have found that the variance on real world datasets is quite close to the worst-case variance, as we demonstrate in Section 6.4. We also vary the privacy budget  $\epsilon$  and domain size  $n$ , studying their impact on the sample complexity for each mechanism and workload.

## 6.2 Impact of Epsilon

Figure 1 shows the relationship between workload and  $\epsilon$  on the sample complexity for each mechanism. We consider  $\epsilon$  ranging from 0.5 to 4.0, fixing  $n$  to be 512. These privacy budgets are common in practical deployments of differential privacy, and local differential privacy in particular [38, 14, 18]. We state our main findings below:

- Our mechanism (labeled Optimized) is consistently the best in all settings: it requires fewer samples than every other mechanism on every workload and  $\epsilon$  we tested.
- The magnitude of improvement over the best competitor varies between 1.0 (Histogram,  $\epsilon = 0.5$ ) and 14.6 (All Range,  $\epsilon = 4.0$ ), but the improvement is typically around



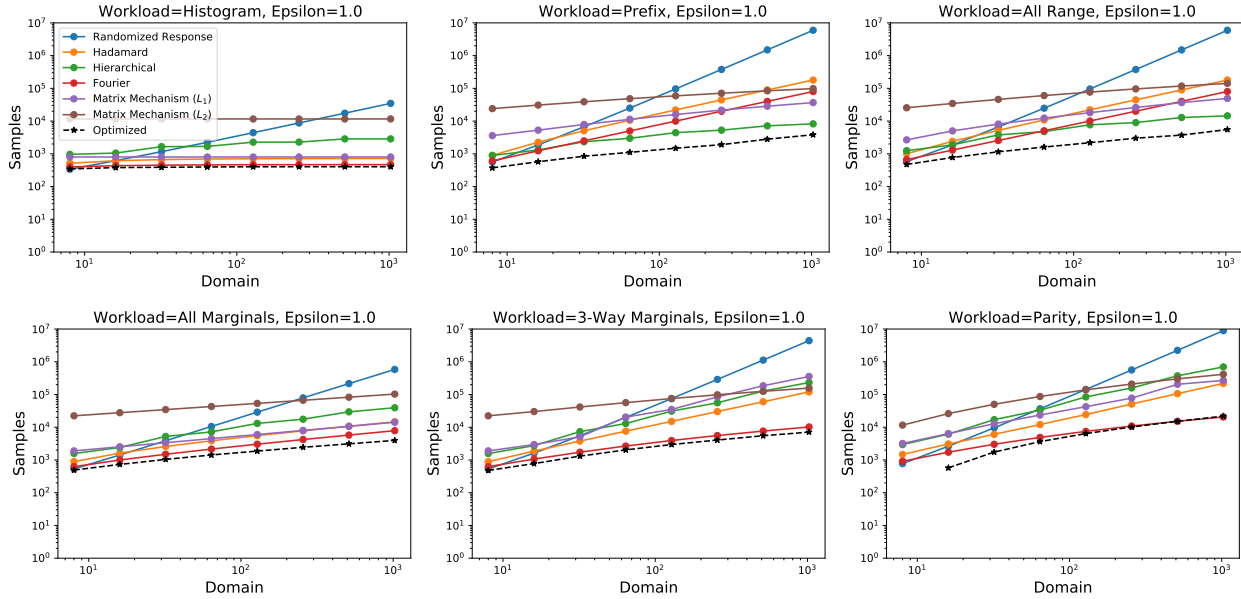


Figure 2: Sample complexity of 7 algorithms on 6 workloads for  $n \in [8, 1024]$

2.5 in the medium privacy regime. In the very high-privacy regime with  $\epsilon \ll 0.5$ , our mechanism is typically quite close to the best competitor, and in the very low-privacy regime with  $\epsilon \gg 4.0$ , our mechanism matches randomized response, which is optimal in that regime. The reduction of required samples translates to a context that really matters: data collectors can now run their analyses on smaller samples to achieve their desired accuracy.

- The best competitor changes with the workload and  $\epsilon$ . For example, the best competitor on the Prefix workload was Hierarchical, while the best competitor on the 3-Way Marginals workload was Fourier. In both cases, these mechanisms were specifically designed to offer low error on their respective workloads, but they don't work as well on other workloads. Additionally, Randomized Response is often the best mechanism at high  $\epsilon$ , so even for a fixed workload, the best competitor is not always the same. On the other hand, our mechanism adapts effectively to the workload and  $\epsilon$ , and works well in all settings. As a result, only one algorithm needs to be implemented, rather than an entire library of algorithms, and, accordingly it is not necessary to select among alternative algorithms.
- Some workloads are inherently harder to answer than others: the number of samples required by our mechanism differs by up to two orders of magnitude between workloads. The easiest workload appears to be Histogram, while the hardest is Parity. This is consistent with the lower bound we gave in Theorem 5.6, which characterizes the hardness of the workload in terms of its singular values – the bound is much lower for Histogram than for Parity.

### 6.3 Impact of Domain Size

Figure 2 shows the relationship between workload and  $n$  on the sample complexity for each mechanism. We consider  $n$  ranging from 8 to 1024, fixing  $\epsilon$  to be 1.0. We state our main findings below:

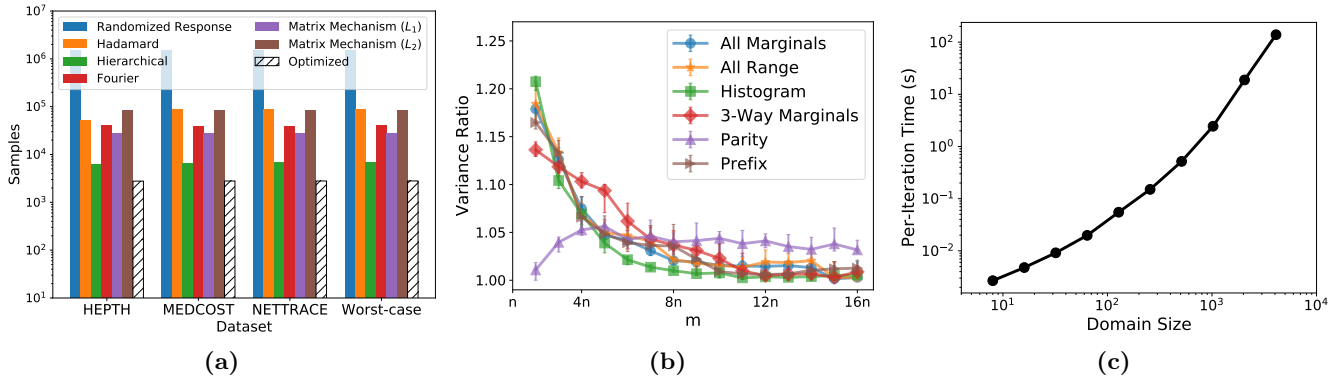
- For the Histogram workload, there is almost no dependence on the domain size for every mechanism except randomized response. This is consistent with our finding in Example 5.8 regarding the lower bound on sample complexity. This observation is unique to the Histogram workload, however.
- The mechanisms that were designed for a given workload, and those that adapt to the workload, have a better dependence on the domain size (smaller slope) than the mechanisms that do not. This includes the  $L_2$  Matrix Mechanism, which is worse than every other mechanism in most settings, but slowly overtakes the other mechanisms for large domain sizes.
- The sample complexity of our mechanism and other mechanisms tailored to the workload is generally  $O(\sqrt{n})$ , as the slope of the lines are  $\approx 0.5$  in log space.<sup>3</sup> On the other hand, the sample complexity of the mechanisms not tailored to the workload is more like  $O(n)$  (as the slope of the line is  $\approx 1.0$ ). These findings are quite interesting: they suggest the improvements offered by workload adaptivity are more than just a constant factor, they grow with the domain size.

### 6.4 Impact of Dataset

Whereas results in previous sections focused on *worst-case* sample complexity, we now turn our attention to sample complexity on *real-world* benchmark datasets obtained from the DPBench study [22]. To calculate the sample complexity on real data, we simply replace  $L_{worst}$  in Corollary 5.4 with the exact (data-dependent) expression for total variance stated in Theorem 3.4.

In Figure 3a, we plot the sample complexity of each mechanism on three datasets for each mechanism on the Prefix workload, fixing  $n = 512$  and  $\epsilon = 1.0$ . We also plot the worst-case sample complexity for reference. As expected, our mechanism still outperforms all others on each dataset. In fact, all

<sup>3</sup>the slope of a line in log space corresponds to the power of a polynomial in linear space; i.e.,  $\log y = m \log x \rightarrow y = x^m$ .



**Figure 3:** (a) Sample complexity on benchmark datasets for Prefix workload. (b) Worst-case variance (ratio to best found) of optimized strategy for various  $m$ . (c) Per-iteration time complexity of optimization for increasing domain sizes.

mechanisms performed pretty consistently, offering similar sample complexities for each dataset. The largest deviation between datasets occurs for the Hadamard mechanism, where it needs  $1.69\times$  more samples for the NETTRACE dataset than for the HEPH dataset. The Optimized mechanism is even more consistent, as the largest deviation is only  $1.006\times$ . Additionally, the real-world sample complexity is very well-approximated by the worst-case sample complexity for the Optimized mechanism, as the maximum deviation is only  $1.009\times$ . This suggests the conclusions drawn based on worst-case sample complexity in Figure 1 and Figure 2 hold for real-world data as well. Although not shown, we repeated this experiment for other workloads and settings and made similar observations.

## 6.5 Initialization

Recall that our optimization algorithm is initialized with a random strategy matrix, and that different initial strategies can lead to different optimized strategies. In this section, we aim to understand how sensitive our optimization algorithm is to the different initializations, and whether it depends on  $m$ , the number of rows in the strategy matrix.

We fix  $n = 64$  and  $\epsilon = 1.0$ , and vary  $m$  from  $2n$  to  $16n$ . For each  $m$ , we compute 10 optimized strategies with different random initializations and record the worst-case variance for each strategy. In order to plot all workloads on the same figure, we normalize the worst-case variance to the best found across all trials. In Figure 3b, we plot the median variance ratio for each  $m$  as well as an error bar to indicate the min and max ratio obtained across the 10 trials. We observe that the optimization is quite robust to the initialization, and produces pretty consistent results between runs, as evident by the small error bars. Furthermore, the optimization is not very sensitive to the choice of  $m$ , as all optimized strategies are within a factor of 1.21 to the best found. Strategies tend to get closer to optimal for larger  $m$ , and eventually level off, with the exception of the Parity workload. We suspect this difference is due to the fact that Parity is a low-rank workload, and doesn't require a large strategy. Using  $m = 4n$  as we did in other experiments tends to produce strategies within a factor of 1.05 to 1.1 of the best found. With enough computational time and resources, we recommend using a hyper-parameter search to find the best  $m$ , as this extra 10% improvement is meaningful in practice.

## 6.6 Scalability of Optimization

We measure the scalability of optimization by looking at the per-iteration time complexity. In each iteration, we must evaluate the objective function (and its gradient) stated in Theorem 3.11, then project onto the constraint set using Algorithm 1. We assume  $\mathbf{W}^T \mathbf{W}$  has been precomputed, and note that the per-iteration time complexity only depends on  $\mathbf{W}^T \mathbf{W}$  through its size, and not its contents. We therefore use the  $n \times n$  identity matrix for  $\mathbf{W}$ . Additionally, we let  $\mathbf{Q}$  be a random  $4n \times n$  strategy matrix. In Figure 3c, we report the per-iteration time required for increasing domain sizes, averaged over 15 iterations. As we can see, optimization scales up to domains as large as  $n = 4096$ , where it takes about 139 seconds per iteration. While expensive, it is not unreasonable to run for a few hundred iterations in this case, and is an impressive scalability result given that there are over 67 million optimization variables when  $n$  is that large. Additionally, we note that strategy optimization is a one-time cost, and it can be done offline before deploying the mechanism to the users. Furthermore, as we showed in Section 6.3, the number of samples required typically increases with the domain size, so there is good reason to run mechanisms on small domains whenever possible, compressing it if necessary. In general, the plot shows that the time grows roughly at a  $O(n^3)$  rate, as it took about 19 seconds for  $n = 2048$  and 2.5 seconds for  $n = 1024$ , confirming the theoretical time complexity analysis.

## 7. RELATED WORK

The mechanism we propose in this work is related to a large body of research in both the central and local model of differential privacy.

Answering linear queries under central differential privacy is a widely studied topic [3, 30, 33, 8, 28]. Many state-of-the-art mechanisms for this task, like the Matrix Mechanism [27], achieve privacy by adding Laplace or Gaussian noise to a carefully selected set of “strategy queries”. This query strategy is tailored to the workload, and can even be optimized for it, as the Matrix Mechanism does. The optimization problem posed by the Matrix Mechanism has been studied theoretically [27, 29], and several algorithms have been proposed to solve it or approximately solve it [44, 28, 43, 33]. While similar in spirit to our mechanism, the optimization problem underlying the Matrix Mechanism is substantially

different from ours, as it requires search over a different space of mechanisms tailored to central differential privacy. For both optimization problems, the optimization variable is a so-called “strategy matrix”, but these represent fundamentally different things in each mechanism, and hence the constraints on the strategy matrix differ. For the Matrix Mechanism, the strategy matrix encodes a set of linear queries which will be answered with a noise-addition mechanism. In contrast, the strategy matrix for our mechanism encodes a conditional probability distribution.

Answering linear queries under local differential privacy has received less attention, but one notable idea is to directly apply mechanisms from central differential privacy to the local model. This translation can be achieved by simply executing the mechanism independently for each single-user database, then aggregating the results. This approach has been studied theoretically with the Gaussian mechanism [4] and the Matrix Mechanism [17]. While these mechanisms trivially provide privacy, they tend to have poor utility in practice, as they are not tailored to the local model. Another notable approach for this task casts it as a mean estimation problem, and uses LDP mechanisms designed for that [9]. These works provide a thorough theoretical treatment of this problem, showing bounds on achieved error, but no practical implementation or evaluation.

More work has been done to answer specific, fixed workloads of general interest, such as histograms [1, 42, 38, 41, 18, 25, 6, 37], range queries [13, 39], and marginals [12, 39]. A very nice summary of the computational complexity, sample complexity and communication complexity for the mechanisms designed for the Histogram workload is given in [1]. Interestingly, even for the very simple Histogram workload, there are multiple mechanisms because the optimal mechanism is not clear. This is in stark contrast to the central model of differential privacy, where, for the Histogram workload it is clear that the optimal strategy is the workload itself. Almost all of these mechanisms are instances of the general class of mechanisms we consider in Definition 3.2, just with different workload factorizations. The strategy matrices for these workloads were all carefully designed to offer low error on the workloads they were designed for, by exploiting knowledge about those specific workloads. However, none of these mechanisms perform optimization to choose the strategy matrix, instead it is fixed in advance.

Kairouz et al. propose an optimization-based approach to mechanism design as well [26]. The mechanism they propose is not designed to estimate histograms or workload queries, but for other statistical tasks, namely hypothesis testing and something they call information preservation. They also consider the class of mechanisms characterized by a strategy matrix (Proposition 2.6), and propose an optimization problem over this space to maximize utility for a given task. Moreover, for convex and sublinear utility functions, they show that the optimal mechanism is a so-called *extremal mechanism*, and state a linear program to find this optimal mechanism. Unfortunately, there are  $2^n$  optimization variables in this linear program, making it infeasible to solve in practice. Furthermore, the restriction on the utility function (sublinear, convex) prevents the technique from applying to our setting.

## 8. CONCLUSION

We proposed a new LDP mechanism that adapts to a workload of linear queries provided by an analyst. We formulated this as a constrained optimization problem over an expressive class of unbiased LDP mechanisms and proposed a projected gradient descent algorithm to solve this problem. We showed experimentally that our mechanism outperforms all other existing LDP mechanisms in a variety of settings, even outperforming mechanisms on the workloads for which they were intended. In the full version of this paper [32], we propose a simple extension to our mechanism that can improve its practical performance even further, through a post-processing technique that ensures non-negativity.

**Acknowledgements.** We would like to thank Daniel Sheldon for his insights regarding Theorem 3.4. This work was supported by the National Science Foundation under grants CNS-1409143, CCF-1642658, CCF-1618512, TRIPODS-1934846; and by DARPA and SPAWAR under contract N66001-15-C-4067.

## A. NOTATION AND MISSING PROOFS

Symbol	Meaning
$n$	domain size
$N$	number of users
$\epsilon$	privacy budget
$\mathcal{M}$	privacy mechanism
$\mathcal{U}$	set of possible users, $ \mathcal{U}  = n$
$\mathcal{O}$	set of possible outcomes, $ \mathcal{O}  = m$
$\mathbf{W}$	$p \times n$ workload matrix
$\mathbf{Q}$	$m \times n$ strategy matrix
$\mathbf{V}$	$p \times m$ reconstruction matrix
$\mathbf{x}$	data vector
$\mathbf{y}$	response vector, $\mathbf{y} = \mathcal{M}_{\mathbf{Q}}(\mathbf{x})$
$\mathbf{1}$	vector of ones
$\mathbf{D}_{\mathbf{Q}}$	$\text{Diag}(\mathbf{Q}\mathbf{1})$
$\mathbf{Q}^\dagger$	pseudo-inverse
$\mathbf{q}_u$	$u^{\text{th}}$ column of $\mathbf{Q}$
$\mathbf{v}_i$	$i^{\text{th}}$ row of $\mathbf{V}$
$\text{tr}[\cdot]$	trace of a matrix
$L_{\text{worst}}(\mathbf{V}, \mathbf{Q})$	worst-case variance
$L_{\text{avg}}(\mathbf{V}, \mathbf{Q})$	average-case variance
$L(\mathbf{V}, \mathbf{Q})$	optimization objective
$L(\mathbf{Q})$	optimization objective for $\mathbf{Q}$
$\text{Diag}(\cdot)$	diagonal matrix from vector

**PROOF OF THEOREM 3.4.** We begin by deriving the variance for a single query  $\mathbf{v}^T \mathbf{y}$  (where  $\mathbf{y} = \mathcal{M}_{\mathbf{Q}}(\mathbf{x})$ ). Note that  $\mathbf{y}$  is a sum of multinomial random variables  $\mathbf{s}_u$  instantiated with parameters  $n = x_u$  and  $\mathbf{p} = \mathbf{q}_u$ , where  $\mathbf{s}_u$  is the response vector for all users of type  $u$ . Using the well-known formula for the covariance of a multinomial random variable, the covariance of a sum, and the variance of a linear combination of correlated random variables, we obtain:

$$\begin{aligned}
\text{Cov}[\mathbf{s}_u] &= x_u(\text{Diag}(\mathbf{q}_u) - \mathbf{q}_u \mathbf{q}_u^T) \\
\text{Cov}[\mathbf{y}] &= \sum_u x_u(\text{Diag}(\mathbf{q}_u) - \mathbf{q}_u \mathbf{q}_u^T) \\
\text{Var}[\mathbf{v}^T \mathbf{y}] &= \mathbf{v}^T \text{Cov}[\mathbf{y}] \mathbf{v} \\
&= \sum_u x_u (\mathbf{v}^T \text{Diag}(\mathbf{q}_u) \mathbf{v} - \mathbf{v}^T \mathbf{q}_u \mathbf{q}_u^T \mathbf{v}) \\
&= \sum_u x_u (\mathbf{v}^T \text{Diag}(\mathbf{q}_u) \mathbf{v} - (\mathbf{v}^T \mathbf{q}_u)^2)
\end{aligned}$$

The total variance is obtained by summing over all the rows of  $\mathbf{V}$ . This completes the proof.  $\square$

PROOF OF THEOREM 3.9. The proof follows from simple algebraic manipulations of  $L_{avg}$ :

$$\begin{aligned} L_{avg}(\mathbf{V}, \mathbf{Q}) &= \frac{N}{n} \sum_{u \in \mathcal{U}} \sum_{i=1}^p \mathbf{v}_i^T \text{Diag}(\mathbf{q}_u) \mathbf{v}_i - (\mathbf{v}_i^T \mathbf{q}_u)^2 \\ &= \frac{N}{n} \left[ \sum_{u \in \mathcal{U}} \sum_{i=1}^p \mathbf{v}_i^T \text{Diag}(\mathbf{q}_u) \mathbf{v}_i - \|\mathbf{V}\mathbf{Q}\|_F^2 \right] \\ &= \frac{N}{n} \left[ \sum_{i=1}^p \mathbf{v}_i^T \text{Diag} \left( \sum_u \mathbf{q}_u \right) \mathbf{v}_i - \|\mathbf{W}\|_F^2 \right] \\ &= \frac{N}{n} [\text{tr}[\mathbf{V}\mathbf{D}_Q \mathbf{V}^T] - \|\mathbf{W}\|_F^2] \end{aligned}$$

$\square$

PROOF OF THEOREM 3.10. Observe that we can construct  $\mathbf{V}$  one row at a time because there are no interaction terms between  $\mathbf{v}_i$  and  $\mathbf{v}_j$  for  $i \neq j$  in the objective function. Furthermore, we can optimize  $\mathbf{v}_i$  through the following quadratic program.

$$\begin{aligned} &\underset{\mathbf{v}_i}{\text{minimize}} && \mathbf{v}_i^T \mathbf{D}_Q \mathbf{v}_i \\ &\text{subject to} && \mathbf{Q}^T \mathbf{v}_i = \mathbf{w}_i \end{aligned}$$

The above problem is closely related to a standard norm minimization problem, and can be transformed to one by making the substitution  $\mathbf{u}_i = \mathbf{D}_Q^{-\frac{1}{2}} \mathbf{v}_i$ . The problem becomes:

$$\begin{aligned} &\underset{\mathbf{u}_i}{\text{minimize}} && \|\mathbf{u}_i\|_2^2 \\ &\text{subject to} && (\mathbf{Q}^T \mathbf{D}_Q^{-\frac{1}{2}}) \mathbf{u}_i = \mathbf{w}_i \end{aligned}$$

This unique solution to this problem is given by  $\mathbf{u}_i = (\mathbf{Q}^T \mathbf{D}_Q^{-\frac{1}{2}})^\dagger \mathbf{w}_i$  [10]. Using the Hermitian reduction identity [7]  $\mathbf{X}^\dagger = \mathbf{X}^T (\mathbf{X}\mathbf{X}^T)^\dagger$ , we have:

$$\begin{aligned} \mathbf{v}_i &= \mathbf{D}_Q^{-\frac{1}{2}} \mathbf{u}_i \\ &= \mathbf{D}_Q^{-\frac{1}{2}} (\mathbf{Q}^T \mathbf{D}_Q^{-\frac{1}{2}})^\dagger \mathbf{w}_i \\ &= \mathbf{D}_Q^{-\frac{1}{2}} \mathbf{D}_Q^{-\frac{1}{2}} \mathbf{Q} (\mathbf{Q}^T \mathbf{D}_Q^{-\frac{1}{2}} \mathbf{D}_Q^{-\frac{1}{2}} \mathbf{Q})^\dagger \mathbf{w}_i \\ &= \mathbf{D}_Q^{-1} \mathbf{Q} (\mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q})^\dagger \mathbf{w}_i \end{aligned}$$

Applying this for all  $i$ , we arrive at the desired solution.  $\mathbf{V} = \mathbf{W} (\mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q})^\dagger \mathbf{Q}^T \mathbf{D}_Q^{-1}$   $\square$

PROOF OF THEOREM 3.11. We plug in the optimal solution for  $\mathbf{V}$  as given in Theorem 3.10 and simplify using linear algebra identities and the cyclic permutation property of trace. We have  $L(\mathbf{Q}) \triangleq \min_{\mathbf{V}} L(\mathbf{V}, \mathbf{Q})$ , which simplifies to:

$$\begin{aligned} L(\mathbf{Q}) &= \text{tr}[\mathbf{W} (\mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q})^\dagger \mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{D}_Q \mathbf{D}_Q^{-1} \mathbf{Q} (\mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q})^\dagger \mathbf{W}^T] \\ &= \text{tr}[\mathbf{W} (\mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q})^\dagger (\mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q}) (\mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q})^\dagger \mathbf{W}^T] \\ &= \text{tr}[\mathbf{W} (\mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q})^\dagger \mathbf{W}^T] \\ &= \text{tr}[(\mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q})^\dagger (\mathbf{W}^T \mathbf{W})] \end{aligned}$$

$\square$

PROOF OF THEOREM 5.1. It is obvious that the worst-case variance is greater than (or equal to) the average-case

variance. We will now bound the worst-case variance from above. Using  $u^*$  to denote the worst-case user, we have the following upper bound on worst-case variance:

$$\begin{aligned} L_{worst}(\mathbf{V}, \mathbf{Q}) &= N \max_{u \in \mathcal{U}} \sum_{i=1}^p \mathbf{v}_i^T \text{Diag}(\mathbf{q}_u) \mathbf{v}_i - (\mathbf{v}_i^T \mathbf{q}_u)^2 \\ &\stackrel{(a)}{\leq} N \sum_{i=1}^p \mathbf{v}_i^T \text{Diag}(\mathbf{q}_{u^*}) \mathbf{v}_i \\ &= \frac{N}{n} \sum_{u \in \mathcal{U}} \sum_{i=1}^p \mathbf{v}_i^T \text{Diag}(\mathbf{q}_{u^*}) \mathbf{v}_i \\ &\stackrel{(b)}{\leq} \frac{e^\epsilon N}{n} \sum_{u \in \mathcal{U}} \sum_{i=1}^p \mathbf{v}_i^T \text{Diag}(\mathbf{q}_u) \mathbf{v}_i \\ &\stackrel{(c)}{=} e^\epsilon \left( \frac{N}{n} \sum_{u \in \mathcal{U}} \sum_{i=1}^p \mathbf{v}_i^T \text{Diag}(\mathbf{q}_u) \mathbf{v}_i - (\mathbf{v}_i^T \mathbf{q}_u)^2 \right) + \frac{e^\epsilon N}{n} \|\mathbf{W}\|_F^2 \\ &= e^\epsilon (L_{avg}(\mathbf{V}, \mathbf{Q}) + \frac{N}{n} \|\mathbf{W}\|_F^2) \end{aligned}$$

In step (a), we use the fact that  $(\mathbf{v}_i^T \mathbf{q}_u)^2$  is non-negative. In step (b), we apply the fact that  $\mathbf{q}_{u^*} \leq \exp(\epsilon) \mathbf{q}_u$  for all  $u$ . In step (c), we express the bound in terms of  $L_{avg}$ , adding 0 in the form of  $\|\mathbf{W}\|_F^2 - \sum_u \sum_i (\mathbf{v}_i^T \mathbf{q}_u)^2$ . This completes the proof.  $\square$

PROOF OF THEOREM 5.6. Consider the following optimization problem which is closely related to Problem 3.12.

$$\begin{aligned} &\underset{\mathbf{X} > 0}{\text{minimize}} && \text{tr}[\mathbf{X}^{-1} (\mathbf{W}^T \mathbf{W})] \\ &\text{subject to} && \mathbf{X}_{uu} \leq 1 \end{aligned}$$

Li et al. derived the SVD bound, which shows the minimum above is at least  $\frac{1}{n} (\lambda_1 + \dots + \lambda_n)^2$  [29]. See also [43]. Furthermore, if  $\mathbf{X}^*$  is the optimal solution and the constraint is replaced with  $\mathbf{X}_{uu} \leq c$  then  $c\mathbf{X}^*$  remains optimal [30], in which case the bound becomes  $\frac{1}{nc} (\lambda_1 + \dots + \lambda_n)^2$ .

We will now argue that any feasible solution to our problem can be directly transformed into a feasible solution of the above related problem. Suppose  $\mathbf{Q}$  is a feasible solution to Problem 3.12 and let  $\mathbf{X} = \mathbf{Q}^T \mathbf{D}_Q^{-1} \mathbf{Q}$ . Note that the objective functions are identical now. We will argue that  $\mathbf{X}_{uu} \leq \frac{\exp(\epsilon)}{n}$ .

$$\begin{aligned} \mathbf{X}_{uu} &= \sum_o \mathbf{Q}_{ou}^2 \frac{1}{\sum_{u'} \mathbf{Q}_{ou'}} \\ &\leq \sum_o \mathbf{Q}_{ou}^2 \frac{\exp(\epsilon)}{n \mathbf{Q}_{ou}} \\ &= \frac{\exp(\epsilon)}{n} \sum_o \mathbf{Q}_{ou} \\ &= \frac{\exp(\epsilon)}{n} \end{aligned}$$

Thus, we have shown that any solution to Problem 3.12 gives rise to a corresponding solution to the above problem. Thus, the SVD Bound applies and we arrive at the desired result:

$$\frac{1}{\exp(\epsilon)} (\lambda_1 + \dots + \lambda_n)^2 \leq L(\mathbf{Q})$$

$\square$

## B. REFERENCES

- [1] J. Acharya, Z. Sun, and H. Zhang. Communication efficient, sample optimal, linear time locally private discrete distribution estimation. *arXiv preprint arXiv:1802.04705*, 2018.
- [2] J. Acharya, Z. Sun, and H. Zhang. Hadamard response: Estimating distributions privately, efficiently, and with little communication. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1120–1129, 2019.
- [3] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 273–282. ACM, 2007.
- [4] R. Bassily. Linear queries estimation with local differential privacy. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 721–729. PMLR, 16–18 Apr 2019.
- [5] R. Bassily, K. Nissim, U. Stemmer, and A. G. Thakurta. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*, pages 2288–2296, 2017.
- [6] R. Bassily and A. Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 127–135, 2015.
- [7] A. Ben-Israel and T. N. Greville. *Generalized inverses: theory and applications*, volume 15. Springer Science & Business Media, 2003.
- [8] A. Bhaskara, D. Dadush, R. Krishnaswamy, and K. Talwar. Unconditional differentially private mechanisms for linear queries. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1269–1284. ACM, 2012.
- [9] J. Blasiok, M. Bun, A. Nikolov, and T. Steinke. Towards instance-optimal private query release. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2480–2497. Society for Industrial and Applied Mathematics, 2019.
- [10] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [11] G. Casella and R. L. Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.
- [12] G. Cormode, T. Kulkarni, and D. Srivastava. Marginal release under local differential privacy. In *Proceedings of the 2018 International Conference on Management of Data*, pages 131–146. ACM, 2018.
- [13] G. Cormode, T. Kulkarni, and D. Srivastava. Answering range queries under local differential privacy. *PVLDB*, 12(10):1126–1138, 2019.
- [14] B. Ding, J. Kulkarni, and S. Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3571–3580, 2017.
- [15] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.
- [16] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.
- [17] A. Edmonds, A. Nikolov, and J. Ullman. The power of factorization mechanisms in local and central differential privacy. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 425–438, 2020.
- [18] Ú. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067. ACM, 2014.
- [19] M. Gaboardi, E. J. G. Arias, J. Hsu, A. Roth, and Z. S. Wu. Dual query: Practical private query release for high dimensional data. In *International Conference on Machine Learning*, pages 1170–1178, 2014.
- [20] A. Griewank et al. On automatic differentiation. *Mathematical Programming: recent developments and applications*, 6(6):83–107, 1989.
- [21] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems*, pages 2339–2347, 2012.
- [22] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, and D. Zhang. Principled evaluation of differentially private algorithms using dpbench. In *Proceedings of the 2016 International Conference on Management of Data*, pages 139–154. ACM, 2016.
- [23] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1-2):1021–1032, 2010.
- [24] N. Holohan, D. J. Leith, and O. Mason. Extreme points of the local differential privacy polytope. *Linear Algebra and its Applications*, 534:78–96, 2017.
- [25] P. Kairouz, K. Bonawitz, and D. Ramage. Discrete distribution estimation under local privacy. In *International Conference on Machine Learning*, pages 2436–2444, 2016.
- [26] P. Kairouz, S. Oh, and P. Viswanath. Extremal mechanisms for local differential privacy. In *Advances in neural information processing systems*, pages 2879–2887, 2014.
- [27] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 123–134. ACM, 2010.
- [28] C. Li and G. Miklau. An adaptive mechanism for accurate query answering under differential privacy. *PVLDB*, 5(6):514–525, 2012.
- [29] C. Li and G. Miklau. Lower bounds on the error of query sets under the differentially-private matrix mechanism. *Theory of Computing Systems*, 57(4):1159–1201, 2015.
- [30] C. Li, G. Miklau, M. Hay, A. McGregor, and V. Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal*, 24(6):757–781, 2015.
- [31] D. Maclaurin, D. Duvenaud, and R. P. Adams. Autograd: Effortless gradients in numpy. In *ICML 2015*

*AutoML Workshop*, volume 238, 2015.

- [32] R. McKenna, R. K. Maity, A. Mazumdar, and G. Miklau. A workload-adaptive mechanism for linear queries under local differential privacy. *arXiv preprint arXiv:2002.01582*, 2020.
- [33] R. McKenna, G. Miklau, M. Hay, and A. Machanavajjhala. Optimizing error of high-dimensional statistical queries under differential privacy. *PVLDB*, 11(10):1206–1219, 2018.
- [34] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
- [35] G. Strang, G. Strang, G. Strang, and G. Strang. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.
- [36] A. G. Thakurta, A. H. Vyrros, U. S. Vaishampayan, G. Kapoor, J. Freudiger, V. R. Sridhar, and D. Davidson. Learning new words, Mar. 14 2017. US Patent 9,594,741.
- [37] S. Wang, L. Huang, P. Wang, Y. Nie, H. Xu, W. Yang, X.-Y. Li, and C. Qiao. Mutual information optimally local private discrete distribution estimation. *arXiv preprint arXiv:1607.08025*, 2016.
- [38] T. Wang, J. Blocki, N. Li, and S. Jha. Locally differentially private protocols for frequency estimation. In *Proc. of the 26th USENIX Security Symposium*, pages 729–745, 2017.
- [39] T. Wang, B. Ding, J. Zhou, C. Hong, Z. Huang, N. Li, and S. Jha. Answering multi-dimensional analytical queries under local differential privacy. In *Proceedings of the 2019 International Conference on Management of Data*, pages 159–176, 2019.
- [40] W. Wang and M. A. Carreira-Perpinán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*, 2013.
- [41] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.
- [42] M. Ye and A. Barg. Optimal schemes for discrete distribution estimation under locally differential privacy. *IEEE Transactions on Information Theory*, 64(8):5662–5676, 2018.
- [43] G. Yuan, Y. Yang, Z. Zhang, and Z. Hao. Convex optimization for linear query processing under approximate differential privacy. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2005–2014. ACM, 2016.
- [44] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao. Low-rank mechanism: optimizing batch queries under differential privacy. *PVLDB*, 5(11):1352–1363, 2012.