

# Efficient Batch Query Answering Under Differential Privacy

Chao Li, Gerome Miklau

University of Massachusetts Amherst, Massachusetts, USA  
 {chaoli, miklau}@cs.umass.edu

May 13, 2017

## Abstract

Differential privacy is a rigorous privacy condition achieved by randomizing query answers. This paper develops efficient algorithms for answering multiple queries under differential privacy with low error. We pursue this goal by advancing a recent approach called the *matrix mechanism*, which generalizes standard differentially private mechanisms. This new mechanism works by first answering a different set of queries (a strategy) and then inferring the answers to the desired workload of queries. Although a few strategies are known to work well on specific workloads, finding the strategy which minimizes error on an arbitrary workload is intractable. We prove a new lower bound on the optimal error of this mechanism, and we propose an efficient algorithm that approaches this bound for a wide range of workloads.

## 1 Introduction

Differential privacy [4] is a rigorous privacy condition, guaranteeing participants that the information released about the data will be virtually indistinguishable whether or not their personal data is included. Differential privacy is achieved by randomizing query answers. While there are a number of general-purpose mechanisms for satisfying differential privacy [2], there are comparatively few results showing that these mechanisms are optimally accurate—that is, that the least possible distortion has been introduced to satisfy the privacy criterion. For a single numerical query, the addition of appropriately-scaled Laplace noise satisfies  $\epsilon$ -differential privacy and has been proven optimally accurate [5]. For workloads of multiple queries, optimally accurate mechanisms are not known.

Our focus is on batch query answering, in which multiple queries are answered at one time, in a single interaction with the private server. A batch of queries, or a workload, here consists of a set of linear counting queries. These include predicate counting queries, histograms, sets of marginals, data cubes, or any combination of these.

The goal of research in this area is to devise an efficient algorithm that can achieve the least possible error under differential privacy. In this work we pursue this goal by advancing a recently-proposed technique called the matrix mechanism [10]. The matrix mechanism generalizes standard differentially private output perturbation techniques, and we explain it by comparing it with the standard Laplace mechanism.

The Laplace mechanism answers a workload of queries by adding to each query a sample chosen independently at random from a Laplace distribution. The distribution is scaled to the sensitivity of the workload (the maximum possible change to the query answers induced by the addition or removal of one tuple). Consider using the Laplace mechanism to simultaneously release answers to the set of all range-count queries over a database containing ages for a community. This workload consists of all queries  $\text{AgeCount}(a, b)$  which return the number of individuals whose age is between  $a$  and  $b$ , for any constants  $a, b \in \{1, \dots, 120\}$ .

Using the Laplace mechanism directly results in extremely noisy query answers for this workload because the noise added to *each* query in the workload is proportional to the sensitivity of the workload. The sensitivity of this workload is  $O(n^2)$  where  $n$  is the size of the domain (120 in this example). In addition, because independent noise is added to each query, the answers are inconsistent: e.g. the sum of  $\text{AgeCount}(20, 40)$  and  $\text{AgeCount}(41, 60)$  will not, in general, equal  $\text{AgeCount}(20, 60)$  as one might hope.

An alternative to the direct application of the Laplace mechanism arises from recognizing that the workload of all range-count queries can be computed from a smaller set of query answers—namely counts for each individual age  $1, 2, \dots, 120$ . Because each count is independent, this approach has very low sensitivity and does not introduce inconsistency. However, the released query results must be summed to get the answers to the desired range queries: e.g.  $AgeCount(1, 10)$  is the sum  $AgeCount(1, 1) + \dots + AgeCount(10, 10)$ . Although each individual count has low error, the expected error accumulates when computing the sum, leading to significant error for large ranges.

The matrix mechanism (so named because workloads are represented as matrices and analyzed algebraically) can be seen as a generalization of the multi-query Laplace mechanism. The approaches above can be seen as two extremes: one in which the workload is submitted to the Laplace mechanism, and one in which the workload is divided into independent queries and those are submitted. The matrix mechanism encompasses both of these extremes, along with many other approaches, some offering significantly lower error.

Given a workload of queries, the matrix mechanism uses the Laplace mechanism to answer a different set of queries, called a *strategy*. The answers to the strategy queries are then used to derive the query answers ultimately desired—the workload queries. If there are related queries in the strategy, linear regression is used to combine the evidence from all available query answers (in an optimal way) to produce the final answers. The result is a consistent final answer to the workload queries, often with improved error. The matrix mechanism can improve error because the strategy queries can be used to avoid or reduce redundancy that may be present in the workload, thus lowering sensitivity. Also, redundancy that does exist in the strategy queries is exploited by the linear regression process to improve accuracy.

Using the matrix mechanism requires instantiating it with a set of strategy queries which is a good match for the workload. For the workload of all range queries, two approaches were independently proposed recently, one based on a wavelet transform [13], and one based on a hierarchical query set [9]. While these approaches look quite different at first glance, they are in fact different sets of strategy queries which can be analyzed as instances of the matrix mechanism [10]. Both strategies result in much lower error:  $O(\log^3 n)$  instead of  $O(n^4)$  for the workload itself or  $O(n)$  for the approach that asks for individual counts.

Thus research to date has shown that for a particular workload (the set of all range-count queries) there are strategies that offer significantly lower error. But these strategies do not work well for all workloads. To exploit the full power of the matrix mechanism, we must customize strategies to the given workload.

Unfortunately, we cannot hope for exact solutions to this problem. The *strategy design problem*, i.e., calculating the strategy that results in the minimal error for a given workload, is intractable [10]. Nevertheless, in this paper we provide efficient algorithms for computing a set of strategy queries for a workload and show that they approach optimally accurate strategies.

A key aspect of our approach is the move from standard  $\epsilon$ -differential privacy, to  $(\epsilon, \delta)$ -differential privacy, a modest relaxation of the privacy condition often called *approximate* differential privacy. The matrix mechanism is easily adapted to approximate differential privacy. Computing the strategy with minimal error remains computationally infeasible. But we show that under this definition the matrix mechanism has a number of nice features which make it amenable to analysis and which lead to better approximate solutions.

Our contributions are organized as follows. The central theoretical result, shown in Sec. 3, is a tight lower bound on the minimum total error achievable for a workload. It can be efficiently computed from the spectral properties of a workload when it is represented in matrix form. In Sec. 4, we propose an efficient localized search algorithm for solving the strategy design problem. In Sec 5, we show experimentally that, for a variety of workloads, our algorithm produces strategies that approach the optimal.

We use our bound on error to understand *workload complexity*—that is, the relative difficulty of simultaneously answering a workload of queries accurately. For instance, sets of multi-dimensional range queries are “easier” to answer than one dimensional range queries. We also use the error bound to evaluate the quality of existing approaches. The hierarchical and wavelet strategies mentioned above are fairly close to optimal in some cases, but can be significantly improved in others. For example, for two-dimensional range queries, the wavelet strategy results in error 2.2 times the optimal while our algorithm finds a strategy that is just 10% greater than the optimal. Ultimately, we conclude that adapting the matrix mechanism to the

specific properties of a workload is crucial: there are workloads for which the total error achieved using our algorithm is an order of magnitude less than that of existing techniques.

## 2 Background

In this section we describe our data model and privacy conditions. We also review the fundamentals of the matrix mechanism, including error measurement and our main problem of strategy design.

### 2.1 Data model and linear queries

The database  $I$  is an instance of a single-relation schema  $R(\mathbb{A})$ , with attributes  $\mathbb{A} = \{A_1, A_2, \dots, A_m\}$ . The crossproduct of the attribute domains, written  $\text{dom}(\mathbb{A})$ , is the set of all possible tuples. In order to express our queries, we first construct from  $I$  a vector  $\mathbf{x}$  of *cell counts*. Each element  $x_i$  of  $\mathbf{x}$  is associated with  $\text{cell}(x_i)$ , a disjoint subset of  $\text{dom}(\mathbb{A})$ . Then  $x_i$  is the count of the tuples from  $\text{cell}(x_i)$  that are present in  $I$ :  $x_i = |\{t \in I \mid t \in \text{cell}(x_i)\}|$ . All queries are expressed using the cell counts in  $\mathbf{x}$ . We always use  $n$  for the size of  $\mathbf{x}$ , which we sometimes refer to simply as the domain size.

One way to define the vector  $\mathbf{x}$  is to choose the smallest possible cells: one cell for each tuple in  $\text{dom}(\mathbb{A})$ . This is often inefficient (the size of the  $\mathbf{x}$  vector is the product of the attribute domain sizes) and ineffective (the base counts are typically too small to be estimated accurately under the privacy condition). Instead, we often consider queries over larger cells. A common way to form a vector of base counts is to partition each  $\text{dom}(A_i)$  into  $d_i$  regions, which could correspond to ranges over an ordered domain, or individual elements (or sets of elements) in a categorical domain. Then the individual cells are defined by taking the cross-product of the regions in each domain. Other alternatives are possible as the cells need not be formed from contiguous subsets of the attribute domains as long as all cells are disjoint. Constructing a vector of base counts appropriate to a workload is usually straightforward but we provide a practical example in App. A to aid the reader.

A linear query computes a specified linear combination of the cell counts in  $\mathbf{x}$ .

**Definition 2.1** (Linear query). *A linear query is a length- $n$  row vector  $\mathbf{q} = [q_1 \dots q_n]$  with each  $q_i \in \mathbb{R}$ . The answer to a linear query  $\mathbf{q}$  on  $\mathbf{x}$  is the vector product  $\mathbf{q}\mathbf{x} = q_1x_1 + \dots + q_nx_n$ .*

A set of queries is represented as a matrix, each row of which is a single linear query.

**Definition 2.2** (Query matrix). *A query matrix is a collection of  $m$  linear queries, arranged by rows to form an  $m \times n$  matrix.*

If  $\mathbf{W}$  is an  $m \times n$  query matrix, the query answer for  $\mathbf{W}$  is a length  $m$  column vector of query results, which can be computed as the matrix product  $\mathbf{W}\mathbf{x}$ .

A *workload* is a query matrix representing a batch of linear queries of interest to a user. We introduce notation for two common workloads which contain all queries of a certain type.  $\text{ALLRANGE}(d_1, \dots, d_k)$  refers to the set of all multi-dimensional range-count queries over  $k$  ordered domains, where each is divided into  $d_i$  regions. Here the size of the cell count vector,  $n$ , is the product  $\prod_{i=1}^k d_i$ .  $\text{ALLPREDICATE}(n)$  is the set of all predicate counting queries over  $n$  cells.  $\text{ALLPREDICATE}(n)$  contains all  $2^n$  linear queries of size  $n$  with coefficients of 0 or 1. We also consider workloads consisting of arbitrary subsets of each of these types of queries, low-order marginals, and their combinations.

### 2.2 Privacy definitions and mechanisms

Standard  $\epsilon$ -differential privacy [4] places a bound (controlled by  $\epsilon$ ) on the difference in the probability of query answers for any two *neighboring* databases. For database instance  $I$ , we denote by  $\text{nbrs}(I)$  the set of databases differing from  $I$  in at most one record. Approximate differential privacy [3, 11], is a modest relaxation in which the  $\epsilon$  bound on query answer probabilities may be violated with small probability (controlled by  $\delta$ ).

**Definition 2.3** (Approximate Differential Privacy). *A randomized algorithm  $\mathcal{K}$  is  $(\epsilon, \delta)$ -differentially private if for any instance  $I$ , any  $I' \in \text{nbrs}(I)$ , and any subset of outputs  $S \subseteq \text{Range}(\mathcal{K})$ , the following holds:*

$$\Pr[\mathcal{K}(I) \in S] \leq \exp(\epsilon) \times \Pr[\mathcal{K}(I') \in S] + \delta$$

Both definitions can be satisfied by adding random noise to query answers. The magnitude of the required noise is determined by the *sensitivity* of a set of queries: the maximum change in a vector of query answers over any two neighboring databases. However, the two privacy definitions differ in the measurement of sensitivity and in their noise distributions. Standard differential privacy can be achieved by adding Laplace noise calibrated to the  $L_1$  sensitivity of the queries [4]. Approximate differential privacy can be achieved by adding Gaussian noise calibrated to the  $L_2$  sensitivity of the queries [3, 11]. This small difference in the sensitivity metric—from  $L_1$  to  $L_2$ —has important consequences for our algorithms and our theoretical results. Our results focus on approximate differential privacy, but App. B contains a thorough comparison of these two definitions as they pertain to the matrix mechanism and the results of this paper.

Since our query workloads are represented as matrices, we express the sensitivity of a workload as a matrix norm. Because neighboring databases  $I$  and  $I'$  differ in exactly one tuple, and because cells are disjoint, it follows that the corresponding vectors  $\mathbf{x}$  and  $\mathbf{x}'$  differ in exactly one component, by exactly one, in which case we write  $\mathbf{x}' \in \text{nbrs}(\mathbf{x})$ . The  $L_2$  sensitivity of  $\mathbf{W}$  is equal to the maximum  $L_2$  norm of the columns of  $\mathbf{W}$ . Below,  $\text{cols}(\mathbf{W})$  is the set of column vectors  $W_i$  of  $\mathbf{W}$ .

**Proposition 2.1** ( $L_2$  Query matrix sensitivity). *The  $L_2$  sensitivity of a query matrix  $\mathbf{W}$  is denoted  $\|\mathbf{W}\|_2$ , defined as follows:*

$$\|\mathbf{W}\|_2 \stackrel{\text{def}}{=} \max_{\mathbf{x}' \in \text{nbrs}(\mathbf{x})} \|\mathbf{W}\mathbf{x} - \mathbf{W}\mathbf{x}'\|_2 = \max_{W_i \in \text{cols}(\mathbf{W})} \|W_i\|_2$$

The classic differentially private mechanism adds independent noise calibrated to the sensitivity of a query workload. We use  $\text{Normal}(\sigma)^m$  to denote a column vector consisting of  $m$  independent samples drawn from a Gaussian distribution with mean 0 and scale  $\sigma$ .

**Proposition 2.2.** (GAUSSIAN MECHANISM [3, 11]) *Given an  $m \times n$  query matrix  $\mathbf{W}$ , the randomized algorithm  $\mathcal{G}$  that outputs the following vector is  $(\epsilon, \delta)$ -differentially private:*

$$\mathcal{G}(\mathbf{W}, \mathbf{x}) = \mathbf{W}\mathbf{x} + \text{Normal}(\sigma)^m$$

where  $\sigma = \|\mathbf{W}\|_2 \sqrt{2 \ln(2/\delta)} / \epsilon$

Recall that  $\mathbf{W}\mathbf{x}$  is a vector of the true answers to each query in  $\mathbf{W}$ . The algorithm above adds independent Gaussian noise (scaled by the sensitivity of  $\mathbf{W}$ ,  $\epsilon$ , and  $\delta$ ) to each query answer. Thus  $\mathcal{G}(\mathbf{W}, \mathbf{x})$  is a length- $m$  column vector containing a noisy answer for each linear query in  $\mathbf{W}$ .

The matrix mechanism has a similar form as the algorithm above, but adds a more complex noise vector. It uses a different set of queries (the strategy matrix,  $\mathbf{A}$ ) to construct this vector.

**Proposition 2.3.** ( $(\epsilon, \delta)$ -MATRIX MECHANISM [10]) *Given an  $m \times n$  query matrix  $\mathbf{W}$ , and assuming  $\mathbf{A}$  is a full rank  $p \times n$  strategy matrix, the randomized algorithm  $\mathcal{M}_{\mathbf{A}}$  that outputs the following vector is  $(\epsilon, \delta)$ -differentially private:*

$$\mathcal{M}_{\mathbf{A}}(\mathbf{W}, \mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{W}\mathbf{A}^+ \text{Normal}(\sigma)^m.$$

where  $\sigma = \|\mathbf{A}\|_2 \sqrt{2 \ln(2/\delta)} / \epsilon$

Here  $\mathbf{A}^+$  is the pseudo-inverse of  $\mathbf{A}$ :  $\mathbf{A}^+ = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$ ; if  $\mathbf{A}$  is a square, then  $\mathbf{A}^+$  is just the inverse of  $\mathbf{A}$ . The intuitive justification for this mechanism is that it is equivalent to the following three-step process: (1) the queries in the strategy are submitted to the Gaussian mechanism; (2) an estimate  $\hat{\mathbf{x}}$  for  $\mathbf{x}$  is derived by computing the  $\hat{\mathbf{x}}$  that minimizes the squared sum of errors (this step consists of standard linear regression and requires that  $\mathbf{A}$  be full rank to ensure a unique solution); (3) noisy answers to the workload queries are then computed as  $\mathbf{W}\hat{\mathbf{x}}$ . The answers to  $\mathbf{W}$  derived in step (3) are always consistent because they are computed from a single noisy version of the cell counts,  $\hat{\mathbf{x}}$ .

Like the Gaussian mechanism, the matrix mechanism computes the true answer vector  $\mathbf{W}\mathbf{x}$  and adds noise to each component. But a key difference is that the scale of the Gaussian noise is *calibrated to the sensitivity of the strategy matrix  $\mathbf{A}$ , not that of the workload*. In addition, the noise added to query answers is no longer independent, because the vector of independent Gaussian samples is transformed by the matrix  $\mathbf{W}\mathbf{A}^+$ .

**Example 2.1.** *The full rank strategy matrix with least sensitivity is the identity matrix,  $\mathbf{I}$ , which has sensitivity 1. With  $\mathbf{A} = \mathbf{I}$ , the matrix mechanism formalizes the approach mentioned in Sec. 1 which computes individual counts and sums them to answer range queries. At the other extreme, the workload itself can be used as the strategy:  $\mathbf{A} = \mathbf{W}$ . In this case, there is no benefit in sensitivity over the Gaussian mechanism. For many workloads, neither of these naive strategies offer optimal error. For  $\mathbf{W} = \text{ALLRANGE}(n)$ , two strategies were recently proposed. The hierarchical strategy includes the total sum over the whole domain, the count of each half of the domain, and so on, terminating with counts of individual elements of the domain. The wavelet strategy consists of the matrix describing the Haar wavelet transformation. Informally, these achieve low error because they each have low sensitivity,  $O(\log n)$ , and every range query can be expressed as a linear combination of few strategy queries.*

## 2.3 Error of the Matrix Mechanism

We measure the accuracy of differentially private query answers using mean squared error. For a workload of queries, the error is defined as the total of individual query errors.

**Definition 2.4** (Query and Workload Error). *Let  $\hat{\mathbf{w}}$  be the estimate for query  $\mathbf{w}$  under the matrix mechanism using query strategy  $\mathbf{A}$ . That is,  $\hat{\mathbf{w}} = \mathcal{M}_{\mathbf{A}}(\mathbf{w}, \mathbf{x})$ . The mean squared error of the estimate for  $\mathbf{w}$  using strategy  $\mathbf{A}$  is:*

$$\text{ERROR}_{\mathbf{A}}(\mathbf{w}) \stackrel{\text{def}}{=} \mathbb{E}[(\mathbf{w}\mathbf{x} - \hat{\mathbf{w}})^2].$$

*Given a workload  $\mathbf{W}$ , the total mean squared error of answering  $\mathbf{W}$  using strategy  $\mathbf{A}$  is:  $\text{ERROR}_{\mathbf{A}}(\mathbf{W}) = \sum_{\mathbf{w}_i \in \mathbf{W}} \text{ERROR}_{\mathbf{A}}(\mathbf{w}_i)$ .*

The query answers returned by the matrix mechanism are linear combinations of noisy strategy query answers to which independent Gaussian noise has been added. Thus, as the following proposition shows, we can directly compute the error for any linear query  $\mathbf{w}$  or workload  $\mathbf{W}$ :

**Proposition 2.4.** (TOTAL ERROR) *Given a workload  $\mathbf{W}$ , the total error of answering  $\mathbf{W}$  using the  $(\epsilon, \delta)$  matrix mechanism with query strategy  $\mathbf{A}$  is:*

$$\text{ERROR}_{\mathbf{A}}(\mathbf{W}) = P(\epsilon, \delta) \|\mathbf{A}\|_2^2 \text{trace}(\mathbf{W}^T \mathbf{W} (\mathbf{A}^T \mathbf{A})^{-1}) \quad (1)$$

where  $P(\epsilon, \delta) = \frac{2 \log(2/\delta)}{\epsilon^2}$ .

The *trace* is the sum of the elements on the main diagonal of a square matrix. Our main objective is to minimize this formula, which is determined by the relationship between  $\mathbf{A}$  and  $\mathbf{W}$ . Note that  $\mathbf{x}$  (the vector of cell counts corresponding to the database) does not appear in this expression. The minimum error strategy depends on the workload alone, not on the database instance.

## 2.4 The strategy design problem

The optimal strategy for a workload  $\mathbf{W}$  is defined to be one that minimizes the total error under the  $(\epsilon, \delta)$ -matrix mechanism.

**Problem 2.1.** (MINIMIZING TOTAL ERROR) *Given a workload  $\mathbf{W}$ , find a query strategy  $\mathbf{A}_0$  such that:*

$$\text{ERROR}_{\mathbf{A}_0}(\mathbf{W}) = \arg \min_{\mathbf{A}} \text{ERROR}_{\mathbf{A}}(\mathbf{W}). \quad (2)$$

The exact solution to Problem 2.1 can be computed using a semi-definite program (SDP) [10]. However, finding the solutions of the program with standard SDP solvers takes  $O(n^8)$  time, making it infeasible for common applications. Therefore, our goal in this paper is to efficiently find approximations to the optimal strategy, for any provided workload.

### 3 Theoretical Analysis

In this section we present novel theoretical results which provide a foundation for the algorithms that come later, and which aid in evaluating both the quality of strategy matrices and the complexity of workloads.

#### 3.1 Analysis of Strategies

An analysis of strategies allows us to characterize those which are possible solutions to the strategy design problem.

**Definition 3.1** (Partial Order on Strategies). *The following relation defines a partial order on strategies.*

**Strategy Equivalence:** Two strategy matrices  $\mathbf{A}_1$  and  $\mathbf{A}_2$  are equivalent ( $\mathbf{A}_1 \equiv \mathbf{A}_2$ ) if for all linear queries  $\mathbf{q}$ ,

$$\text{ERROR}_{\mathbf{A}_1}(\mathbf{q}) = \text{ERROR}_{\mathbf{A}_2}(\mathbf{q}).$$

**Strategy Efficiency:** Strategy  $\mathbf{A}_1$  is more efficient than  $\mathbf{A}_2$ , written  $\mathbf{A}_1 \leq \mathbf{A}_2$ , if for all linear queries  $\mathbf{q}$ ,

$$\text{ERROR}_{\mathbf{A}_1}(\mathbf{q}) \leq \text{ERROR}_{\mathbf{A}_2}(\mathbf{q}).$$

$\mathbf{A}_1$  is strictly more efficient than  $\mathbf{A}_2$  if for all queries  $\mathbf{q}$ ,  $\text{ERROR}_{\mathbf{A}_1}(\mathbf{q}) < \text{ERROR}_{\mathbf{A}_2}(\mathbf{q})$ .

A strategy can never be the solution to the strategy design problem (Problem 2.1) if there is another strategy that is more efficient than it. We call a strategy *minimal* if it is a minimal element in the partial order above.

We show next that the set of minimal strategies has a straightforward characterization for the  $(\epsilon, \delta)$  matrix mechanism. Notice that since the sensitivity of a strategy is measured according to its maximum column norm, a strategy with some columns that do not match the maximum is wasteful: we could add queries containing non-zero entries in the deficient columns without increasing sensitivity. These added queries will provide additional evidence that can be used to reduce overall error, so completing deficient columns can never result in a worse strategy.

**Definition 3.2** (Column Uniform Matrix). *A matrix is column-uniform with respect to  $L_p$  if each of its columns has the same  $L_p$  norm.*

This suggests that column uniformity is a necessary condition for a minimal strategy. But we can also show that it is a sufficient condition. (The proof is included in App. D.1.)

**Theorem 3.1.** *A strategy matrix  $\mathbf{A}$  is minimal iff it is column-uniform.*

Another important property of the  $(\epsilon, \delta)$  matrix mechanism is that redundant queries do *not* lead to less efficient strategies. We say a query in strategy matrix  $\mathbf{A}$  is *redundant* if it is a linear combination of another query in  $\mathbf{A}$ . In this case the queries provide the same evidence about the database, but one may be scaled relative to the other, resulting in lesser or greater accuracy. The following theorem shows that a strategy with a redundant queries is equivalent to a strategy with the redundancy removed.

**Theorem 3.2** (Redundant Queries). *Suppose strategy  $\mathbf{A}_1 = \{\mathbf{A}_0 \cup \mathbf{q} \cup c_1 \mathbf{q}\}$  for some strategy  $\mathbf{A}_0$ , some linear query  $\mathbf{q}$ , and some constant  $c_1$ . Then  $\mathbf{A}_1$  is equivalent to the reduced strategy  $\mathbf{A}_2 = \{\mathbf{A}_0 \cup c_2 \mathbf{q}\}$  where  $c_2 = \sqrt{1 + c_1^2}$ .*

The fact that the presence of redundant queries does not lead to less efficient strategies has important consequences for efficient strategy design algorithms. Because of this property, an algorithm can make a local choice to add a query to a strategy, adding the same query again later to augment its weight if necessary.

Thm. 3.1 and Thm. 3.2 do not hold for the  $\epsilon$ -matrix mechanism. Please see App. B for details.

### 3.2 The singular value bound

Next we explain our main theoretical result, a lower bound on the error for a workload. We first define the singular value decomposition of a workload.

**Definition 3.3** (Decomposition of Workload). *Let  $\mathbf{W}$  be any  $m \times n$  query workload. The singular value decomposition (SVD) of  $\mathbf{W}$  is a factorization of the form  $\mathbf{A} = \mathbf{Q}_\mathbf{W} \mathbf{D}_\mathbf{W} \mathbf{P}_\mathbf{W}^T$  such that  $\mathbf{Q}_\mathbf{W}$  is a  $m \times m$  orthogonal matrix,  $\mathbf{D}_\mathbf{W}$  is a  $m \times n$  diagonal matrix and  $\mathbf{P}_\mathbf{W}$  is a  $n \times n$  orthogonal matrix. When  $m > n$ , the diagonal matrix  $\mathbf{D}_\mathbf{W}$  consists of an  $n \times n$  diagonal submatrix combined with  $\mathbf{0}^{(m-n) \times n}$ .*

The bound on the total error for a workload is derived from its singular value decomposition.

**Theorem 3.3.** (SINGULAR VALUE BOUND) *Given an  $m \times n$  workload  $\mathbf{W}$ , let  $\lambda_1, \lambda_2, \dots, \lambda_n$  be the singular values of  $\mathbf{W}$ .*

$$\min_{\mathbf{A}} \text{ERROR}_{\mathbf{A}}(\mathbf{W}) \geq P(\epsilon, \delta) \frac{1}{n} \left( \sum_{i=1}^n \lambda_i \right)^2,$$

where  $P(\epsilon, \delta) = \frac{2 \log(2/\delta)}{\epsilon^2}$ .

A strategy matrix  $\mathbf{A}$  can be considered to have two parts: its eigenvalues and its eigenvectors. When  $\mathbf{A}$  is column-uniform, the total error can be represented as a polynomial of those two parts. The key idea behind the SVD bound is to ignore the constraint that  $\mathbf{A}$  is column uniform and choose eigenvalues and eigenvectors separately to minimize the polynomial, which leads to an under-estimate of the total error. The details of this proof are presented in App. D.1.

We use  $\text{SVDB}(\mathbf{W})$  as shorthand for the singular value bound of workload  $\mathbf{W}$ . Given a workload  $\mathbf{W}$ ,  $\text{SVDB}(\mathbf{W})$  can be computed easily using standard methods for matrix decomposition, or it can be computed directly from  $\mathbf{W}^T \mathbf{W}$ :

**Corollary 3.1.** *Given an  $n \times n$  positive semi-definite matrix  $\mathbf{W}^T \mathbf{W}$ , let  $\lambda_1, \dots, \lambda_n$  be the eigenvalues of  $\mathbf{W}^T \mathbf{W}$ .*

$$\min_{\mathbf{A}} \text{ERROR}_{\mathbf{A}}(\mathbf{W}) \geq P(\epsilon, \delta) \frac{1}{n} \left( \sum_{i=1}^n \sqrt{\lambda_i} \right)^2, \quad (3)$$

where  $P(\epsilon, \delta) = \frac{2 \log(2/\delta)}{\epsilon^2}$ .

For  $m \times n$  workload  $\mathbf{W}$ , computing  $\text{SVDB}(\mathbf{W})$  takes  $O(mn^2)$  time on  $\mathbf{W}$  itself and takes  $O(n^3)$  time on  $\mathbf{W}^T \mathbf{W}$ , which significantly reduces the running time when the number of queries,  $m$ , is much larger than  $n$ . In the case of large regular workloads such as ALLRANGE and ALLPREDICATE,  $\mathbf{W}^T \mathbf{W}$  can be computed directly. Then computing  $\text{SVDB}(\mathbf{W})$  improves from  $O(n^5)$  to  $O(n^3)$  for ALLRANGE and  $O(n^2 2^n)$  to  $O(n^3)$  for ALLPREDICATE.

**Example 3.1.** *For ALLRANGE(1024) and ALLRANGE(32, 32), the SVD bound on the total error is  $5.32 \times 10^6$  and  $4.39 \times 10^6$ , respectively. Below we report, as a ratio of the SVDB bound, the total error of these workloads for a number of known strategy matrices: the workload itself, the identity, hierarchical and wavelet:*

	workload	identity	hierarchical	wavelet
1024	50.58	33.75	2.14	1.84
32 × 32	17.25	8.15	2.92	2.23

*In the next section we present an algorithm that finds better strategies: the ratio of total error on ALLRANGE(1024) and ALLRANGE(32, 32) using the strategies computed by the algorithm is 1.26 and 1.08, respectively.*

### 3.3 Properties of the singular value bound

The SVD bound has a number of properties which make it a reliable measure of workload complexity. Notice that in the expression for  $\text{ERROR}_{\mathbf{A}}(\mathbf{W})$  in Prop. 2.4, the workload  $\mathbf{W}$  appears only in the trace term, as  $\mathbf{W}^T \mathbf{W}$ . An immediate implication is that a strategy that achieves minimal total error for a given workload  $\mathbf{W}_1$  achieves minimal error for any workload  $\mathbf{W}_2$  such that  $\mathbf{W}_1^T \mathbf{W}_1 = \mathbf{W}_2^T \mathbf{W}_2$ . We therefore define the following notion of equivalence:

**Definition 3.4** (Workload Equivalence). *An  $n \times m_1$  workload  $\mathbf{W}_1$  and an  $n \times m_2$  workload  $\mathbf{W}_2$  are equivalent, denoted  $\mathbf{W}_1 \equiv \mathbf{W}_2$ , iff  $\mathbf{W}_1^T \mathbf{W}_1 = \mathbf{W}_2^T \mathbf{W}_2$ .*

Since the SVD bound is determined by the singular values, it follows immediately that equivalent workloads have equal error bounds. That is, if  $\mathbf{W}_1 \equiv \mathbf{W}_2$ , then  $\text{SVDB}(\mathbf{W}_1) = \text{SVDB}(\mathbf{W}_2)$ . In addition, as we would hope, the SVD bound of a workload increases monotonically under the addition of new queries. Thus if  $\mathbf{W}_1$  is a workload and  $\mathbf{W}_2$  is a workload that results from adding one or more linear queries to the rows of  $\mathbf{W}_1$ , then  $\text{SVDB}(\mathbf{W}_1) \leq \text{SVDB}(\mathbf{W}_2)$ . A more general result is shown below, accounting for the fact that the larger workload may be the augmentation of any workload equivalent to the smaller workload. (The proof is omitted.)

**Theorem 3.4.** *Let  $\mathbf{W}_1, \mathbf{W}_2$  be workloads. If there exists a workload  $\mathbf{W}'_2$  equivalent to  $\mathbf{W}_2$  and the rows of  $\mathbf{W}'_2$  contain all the rows of  $\mathbf{W}_1$ , then  $\text{SVDB}(\mathbf{W}_1) \leq \text{SVDB}(\mathbf{W}_2)$ .*

Lastly, and most importantly, the singular value bound is tight. For a certain set of *variable agnostic* workloads, it is possible to directly construct a strategy achieving the bound. Intuitively, variable agnostic workloads treat every variable in the domain in an equivalent manner. The workload  $\text{ALLPREDICATE}(n)$  is variable agnostic, but  $\text{ALLRANGE}(n)$  is not because, for example, variables in the middle of the domain occur more often in the set of all range queries. In App. C we show how to construct the optimal strategy for any variable agnostic workload, including for  $\text{ALLPREDICATE}(n)$ .

We do not know if the singular value bound is achievable for every workload (namely those that are not variable-agnostic). However experimentally we find that we can compute strategies that approach this bound.

## 4 Strategy Selection Algorithms

In this section we present an algorithm, along with a set of performance optimizations, for computing a close-to-optimal strategy for a given workload.

### 4.1 The Level Selection Algorithm (LSA)

The Level Selection Algorithm (LSA) takes as input a workload and returns a strategy matrix designed to offer low error for the workload. LSA is a localized search algorithm which builds a strategy matrix by choosing, at each step, the *level* of queries whose addition maximally reduces error for the workload. A level is a set of linear queries consisting of coefficients 0 or 1, determined by a partitioning of the variables, so that each variable appears in exactly one query. Recall that, according to Thm. 3.1, minimal strategy matrices are precisely those that are column uniform. By constructing a strategy by levels, the LSA algorithm always chooses among minimal strategies.

To construct a new level, the LSA algorithm starts with the simplest level: the query  $[1, 1, \dots, 1]$  which is the sum of all cells. The algorithm then performs a top-down search, recursively bisecting the query into smaller queries such that the error of the workload is maximally reduced. Once a level is completed, the LSA algorithm computes the total error of the workload with and without that level. If the total error is not improved by the level, the algorithm terminates and outputs the current strategy. Otherwise the level is added to the output strategy, and, subject to a user-defined threshold  $k$  on the number of levels, the next level is then constructed.

The search space of LSA is quite general. Both the hierarchical strategies from [9] and (a strategy equivalent to) the wavelet strategy [13] can be constructed by levels. But the LSA algorithm is not limited to hierarchical strategies because there is no constraint that lower level queries are contained in higher level queries. Further, while levels are constructed initially with coefficients of 0 or 1, the final output strategy may have a complex set of weights on queries, because redundant queries may be added in different levels. These redundant queries can be combined, as shown in Thm. 3.2, by appropriate weighting.

**Example 4.1.** *For  $\mathbf{W} = \text{ALLRANGE}(512)$ , the LSA algorithm terminates at 33 levels. The output strategy consists of 4746 queries, reducible to 1931 non-redundant queries.*



---

**Program 4.1** The Level Selection Algorithm (LSA)

---

**Input:** A workload matrix  $\mathbf{W}$  and the size of each dimension of the domain. An upper bound  $k$  to the maximum number of levels.

**Output:** A strategy matrix  $\mathbf{A}$ .

```
1:  $\mathbf{A} = \mathbf{I}$ ;  
2: repeat  
3:    $\delta_{\mathbf{A}} = [1, 1, \dots, 1]$ ;  
4:   repeat  
5:     for each query  $\mathbf{q}$  in  $\delta_{\mathbf{A}}$  do  
6:       find a pair  $(i, j)$  such that split at position  $i$  on dimension  $j$  such that using query:  
  
                $\mathbf{q}_1 = \{\text{buckets with value 1 in } \mathbf{q} \text{ with}$   
                    $i\text{-th dimension less than } j\}$   
  
                $\mathbf{q}_2 = \{\text{buckets with value 1 in } \mathbf{q} \text{ with}$   
                    $i\text{-th dimension greater than or equal to } j\}$   
  
               to take the place of query  $\mathbf{q}$  in  $\delta_{\mathbf{A}}$  and  
                $\mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ \delta_{\mathbf{A}} \end{bmatrix}$  minimizes  $\text{ERROR}_{\mathbf{A}'}(\mathbf{W})$ .  
7:     end for  
8:     if  $\text{ERROR}_{\mathbf{A}'}(\mathbf{W})$  is reduced then update  $\delta_{\mathbf{A}}$ ,  
9:     until  $\delta_{\mathbf{A}}$  was not updated;  
10:    if  $\text{ERROR}_{\mathbf{A}'}(\mathbf{W}) \leq \text{ERROR}_{\mathbf{A}}(\mathbf{W})$  then  $\mathbf{A} = \mathbf{A}'$ .  
11:  until  $\mathbf{A}$  was not updated or  $\|\mathbf{A}\|_2^2 \geq k$ ;  
12: return  $\mathbf{A}$ .
```

---

## 4.2 Complexity of the LSA algorithm

Note that Program 4.1 can use  $\mathbf{W}^T \mathbf{W}$  instead of  $\mathbf{W}$  as input without impacting any computation or the final output. This implies that equivalent workloads produce equivalent outputs, and also that the complexity of Program 4.1 does not depend on the number of queries in  $\mathbf{W}$ .

In each iteration, the algorithm needs to find the split point which maximizes the reduction of total error. This occurs in Step 6 and it is the most time-consuming part of Prog. 4.1. The total error for each possible split has to be computed, which means computing the total error  $O(n)$  times to determine one split point. There are at most  $n$  split points so the total error is computed at most  $O(n^2)$  over all the iterations. Each total error computation requires recomputing  $(\mathbf{A}'^T \mathbf{A}')^{-1}$  for an updated strategy  $\mathbf{A}'$ . This requires  $O(n^3)$  time by ordinary matrix inversion, so the entire algorithm would take  $O(kn^5)$  time, where the number of levels is bounded by  $k$ .

However, because only three rows of  $\delta_{\mathbf{A}}$  are updated during each iteration, we can improve the running time by exploiting a more efficient incremental computation of the matrix inverse. Details of this technique and the proof of the following theorem are included in App. D.2.

**Theorem 4.1.** *The LSA algorithm can be implemented in  $O(kn^4)$  time, where  $k$  is the maximum number of levels.*

In experiments we have run the LSA algorithm to termination and found that the number of levels is much smaller than  $n$  and that it converges quickly. Fig. 1 shows the convergence of the error as a function of the bound  $k$  on the number of levels for  $\mathbf{W} = \text{ALLRANGE}(512)$ . In addition, although the worst case cost of the LSA algorithm is  $O(kn^4)$ , empirically we find that the running time increases approximately with  $O(kn^3)$ .

## 4.3 The LSA Algorithm on Large Domains

In the remainder of this section we present two techniques for efficiently scaling the LSA algorithm to larger domains.

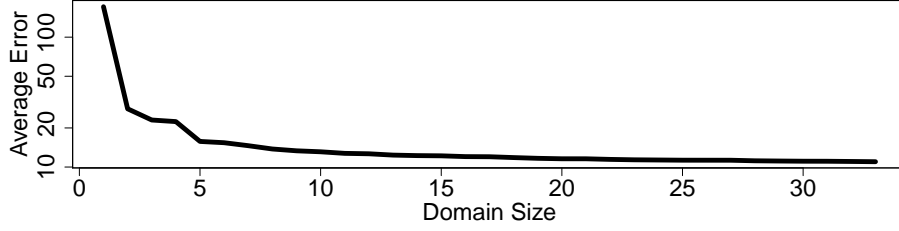


Figure 1: The total error of strategies found by LSA as the bound  $k$  on the number of levels increases.

#### 4.3.1 Workload Separation

The domain for workloads over multi-dimensional data can increase with the product of the attribute domains. To avoid this, many common workloads, such as those resulting from sets of low-order marginals, can be decomposed so that each dimension can be treated separately. This process of *workload separation* can significantly improve efficiency with little impact on overall error.

**Definition 4.1.** Given a workload  $\mathbf{W}$ , query  $\mathbf{q}$  is called a separating query for  $\mathbf{W}$  if there exists a list of subsets  $\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_k$  of  $\mathbf{W}$  for which the estimate of  $\mathbf{W}_i$  does not involve queries in  $\mathbf{W}_1, \dots, \mathbf{W}_{i-1}, \mathbf{W}_{i+1}, \dots, \mathbf{W}_k$  when the answer of query  $\mathbf{q}$  is fixed. The subsets  $\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_k$  are called separated workloads of  $\mathbf{W}$  under query  $\mathbf{q}$ .

Given a workload  $\mathbf{W}$  and one of its separating queries  $\mathbf{q}$ , notice that answering queries in each separated workload does not involve other queries in the workload. Thus workload separation can be applied in two phases. In the first phase, the separating query  $\mathbf{q}$  is answered directly, using the Gaussian mechanism with fixed accuracy. In the second phase, the strategies for each separated workload of  $\mathbf{W}$  are computed with the LSA algorithm. Since each separated workload contains queries with simpler structure, the buckets that always appear at the same time in one separated workload can be grouped into a single bucket so as to reduce the domain size.

One of the key applications of workload separation is for workloads consisting of sets of marginal queries and predicate queries over marginals, explained in the following theorem.

**Theorem 4.2.** Let  $\mathbf{W}_1, \dots, \mathbf{W}_k$  be sets of predicate queries over one-way marginals on  $k$  different dimensions. Let  $\mathbf{q}_0$  be the sum of all the entries on the domain. With the answer of  $\mathbf{q}_0$  given, the estimate of any query in  $\mathbf{W}_i$  does not involve queries in  $\mathbf{W}_1, \dots, \mathbf{W}_{i-1}, \mathbf{W}_{i+1}, \dots, \mathbf{W}_k$ .

More generally, when a workload contains predicate queries over up to  $k$ -way marginals, such as datacube queries, workload separation can then be applied recursively, reducing the problem to low- or even one-dimensional problems.

**Example 4.2.** Consider a query workload consisting of all one-dimensional range queries over each of two dimensions, with domains of size  $n_1$  and  $n_2$ . Such a workload would typically be represented over a set of cell counts of size  $n_1 \times n_2$ , and the LSA algorithm would take  $O(k(n_1 n_2)^4)$  time. The separated workloads are  $\text{ALLRANGE}(n_1)$  and  $\text{ALLRANGE}(n_2)$  and the cost of running LSA is reduced to  $O(k(n_1^4 + n_2^4))$ .

#### 4.3.2 Workload Generalization

As a second optimization, called workload generalization, we initially merge cells in the domain, creating an approximation of the original workload over a much smaller domain. The LSA algorithm is executed on this generalized workload to produce an initial generalized strategy. Then, in a second phase, we run a slightly modified Program 4.1 over each merged cell. The modified algorithm computes error and sensitivity

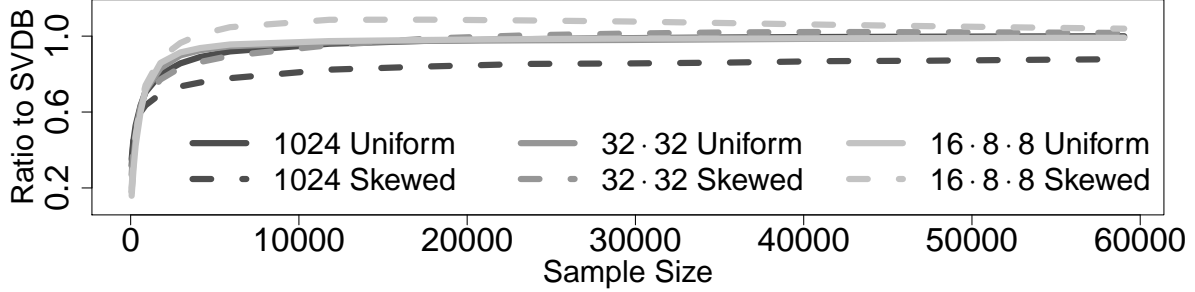


Figure 2: The SVD Bound for Sampled Workloads

with the strategy of the first phase taken into account. The strategies on both phases are then combined to produce the final strategy.

The cost of this method is a function of domain size  $n$  and the generalized domain size  $m$ . The first phase runs Prog 4.1 once over domain size  $m$  and the second phase runs modified Prog 4.1  $m$  times over domain size  $n/m$ . Thus, the total cost of workload generalization is  $O(k(m^4 + n^4/m^3))$ . If  $m$  is set to  $O(n^{1/3})$ , then the total cost can be reduced to  $O(kn^3)$ . This is the asymptotic cost as matrix multiplication, thus it is a natural lower bound for the running time of an algorithm of this form. Workload generalization results in much faster execution time, but sacrifices solution quality in comparison the LSA algorithm, resulting in modestly higher error. We evaluate this experimentally in the next section.

## 5 Experimental Evaluation

There are two parts to the experimental evaluation of our techniques. First, we study the complexity of different workloads using the singular value bound. Second, we evaluate the quality of strategies generated by the LSA algorithm, as well as the effectiveness of the workload separation and generalization techniques. Recall from Sec. 2.3 that the error of the matrix mechanism, and therefore the choice of strategy, is independent of the database instance. As a consequence, the results that follow do not use an input database—they are purely an analysis of workloads and the error rates possible in answering these workloads.

### 5.1 Workload Complexity

We can use the singular value bound to gain insight into the complexity of workloads—the essential hardness of answering a set of queries under the matrix mechanism. We begin by reporting the SVD bound for all range workloads on five different domains, all with a domain size 1024 but different number of dimensions. The comparison is shown as the lightest grey bars in **Figure 3(a)**, where  $2^{10}$  means the 10-dimensional domain with each dimension size equal to 2. As the number of dimensions increases with the overall domain fixed, the total number of range queries decreases, leading to lower complexity workloads, as shown by the decreasing SVD bound in the figure.

Our next experiment, shown in **Figure 2**, samples a matching number of distinct queries from each workload and considers the average error of the sampled query sets. The complexity of a sampled query set is measured by the ratio between average error of the sample and the average error of the complete query set. Moreover, the experiment uses both uniform sampling and a biased sampling method<sup>1</sup> to study whether the SVD bound is related to the sampling method.

<sup>1</sup>Biased sampling chooses queries whose center is far away from the center of the domain, and the probability for

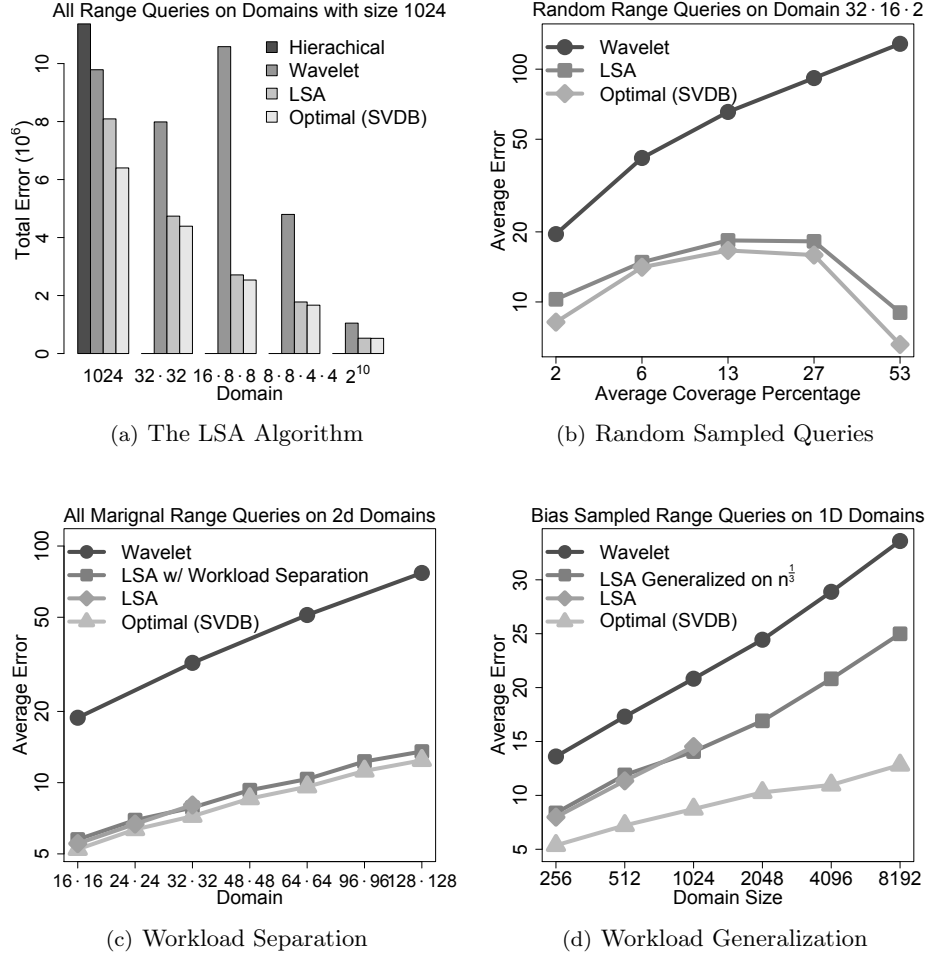


Figure 3: Performance of the LSA algorithm and auxiliary techniques.

**Figure 2** shows that even very small workloads of uniformly sampled queries (10000 queries, or roughly 2-6% of the entire workload of range queries) reach the error levels of the entire workload. When biased sampling is used to construct workloads, the error ratio approaches 1 more slowly. Overall, this may have important consequences for the strategy selection problem. It suggests that, for workloads of just modest size, there is no error penalty in adopting a query strategy tuned to a larger regular workload such as ALLRANGE for the appropriate domain size.

## 5.2 Effectiveness of the LSA Algorithm

To assess the effectiveness of the LSA algorithm we compare the total error under its strategies with other strategies from the literature, and with the optimal error as derived from the SVD bound. The first experiment, shown in **Figure 3(a)**, explores ALLRANGE workloads over the five multi-dimensional domains mentioned above. We compare the LSA-computed strategy with the wavelet strategy [13] and the hierar-

choosing a range decreases exponentially with the distance between the center of the query and the center of the domain. The motivation of having this sampling method is to create a way of sampling that has different properties with uniform sampling, which can also be viewed as a process where users whose most interested region is at the extreme of the domain so that they tend to generate queries that are far away from the center.

chical strategy [9]. Recall that both these strategy matrices were originally designed for  $\epsilon$ -differential privacy, but in fact they perform even better under  $(\epsilon, \delta)$ -differential privacy. The hierarchical tree algorithm only appears in the first group of comparisons because it is oriented towards one-dimensional queries.

Both the wavelet and the hierarchical strategies perform well in the one-dimensional case: their error is 1.53 times the optimal and 1.78 times the optimal, respectively. The LSA algorithm finds a better strategy, 1.26 times the optimal. In higher dimensions the performance improvement over wavelet is much larger, demonstrating the importance of adapting the strategy to the workload. For example, for ALLRANGE(16, 8, 8), LSA error is 1.07 times optimal while wavelet is 4.17 times optimal. Overall, the error of LSA strategies comes very close to the optimal singular value bound in higher dimensions.

To further test the benefits of adapting the strategy to the workload, we considered random workloads of range queries grouped by average coverage percentage, which is the average percentage of cells that are covered by a query. As shown in **Figure 3(b)** (note the logarithmic scale), the LSA strategies have error rates very close to optimal and outperform the wavelet strategy by more than an order of the magnitude in the most extreme case.

Next we evaluate the workload separation and workload generalization techniques for improved efficiency of the LSA algorithm on large domains. **Figure 3(c)** shows the workload separation technique on range queries over pairs of one-way marginals with increasing domain sizes. Using LSA with workload separation has a negligible penalty in error: it is almost identical to the original LSA (and the optimal SVD bound), yet the computational cost is improved by more than three orders of magnitude: On domain  $32 \cdot 32$ , the running time is reduced from 5079 seconds to merely 2.44 seconds.

To evaluate the workload generalization techniques we considered skew-sampled workloads on one dimension, since their properties differ from ALLRANGE workloads. **Figure 3(d)** compares the performance of LSA generalized on domain size  $n^{1/3}$ , the original LSA algorithm, the wavelet strategy, and the singular value bound. Generalized LSA performs almost as well as the original LSA algorithm. However, the computational cost is decreased significantly: for domain 1024, the cost is reduced by a factor of 18.

## 6 Related Work

The matrix mechanism [10] analyzed in a unified framework two prior techniques for accurately answering range queries. The first used a wavelet transformation [13]; the second used a hierarchical set of queries followed by inference [9]. The original work on the matrix mechanism focused primarily on  $\epsilon$ -differential privacy, although  $(\epsilon, \delta)$ -differential privacy was considered briefly. In both cases, it was shown that semi-definite programming could be used to compute a minimum error strategy for a workload, but solving such programs is not infeasible. The present work provides tractable methods for computing low-error strategies for any given workload.

Workload complexity under  $\epsilon$ -differential privacy has been studied before. Hardt and Talwar [8] present a lower bound for randomly generated predicate workloads. More generally, Blum et al. [1] show that workload complexity is related to the VC dimension of the workload. This measure is not directly comparable to our singular value bound since the former concerns  $\epsilon$ -differential privacy. Also, the VC dimension of any one-dimensional range workload is constant, while our measure captures more detailed differences in workloads.

Put in our terms, Xiao et al. [14] propose a method for computing a strategy matrix using KD-trees and private accesses to the database. Their query answers could be improved and made consistent by employing the matrix mechanism. Roth and Roughgarden [12] describe a data-dependent mechanism to answer predicate queries on databases with 0-1 entries. Hardt et. al [7] provide a linear time algorithm for the same query and database setting. The trade-off between accuracy and efficiency among data-dependent and data-independent methods deserves further investigation.

## 7 Conclusion

Standard differentially-private mechanisms for answering a workload of queries require noise determined by the sensitivity of the queries. We have shown that it is possible to satisfy the privacy condition with substantially less noise, and that the noise required is related to the spectral properties workload. Our methods allow the privacy mechanism to be efficiently adapted to the workload, achieving error improvements of as much as one order of magnitude over prior techniques.

## References

- [1] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, 2008.
- [2] C. Dwork. A firm foundation for privacy. In *Communications of the ACM*, 2011.
- [3] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, 2006.
- [4] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.
- [5] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. In *STOC*, 2009.
- [6] W. W. Hager. Updating the inverse of a matrix. *SIAM Rev.*, 31:221–239, June 1989.
- [7] M. Hardt and G. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, pages 61–70. IEEE, 2010.
- [8] M. Hardt and K. Talwar. On the geometry of differential privacy. In *STOC*, 2010.
- [9] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially-private histograms through consistency. In *PVLDB*, 2010.
- [10] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, pages 123–134, 2010.
- [11] F. McSherry and I. Mironov. Differentially Private Recommender Systems : Building Privacy into the Netflix Prize Contenders. In *SIGKDD*, 2009.
- [12] A. Roth and T. Roughgarden. The median mechanism: Interactive and efficient privacy with multiple queries. In *STOC*, 2010.
- [13] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, 2010.
- [14] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through multidimensional partitioning. In *SDM*, 2010.

## A Data model and Domain

In this section we provide a brief concrete example to explain how a workload of counting queries can be expressed as a set of linear queries over a vector of cell counts  $\mathbf{x}$ . Consider the following relational schema describing students:

$$R = (\text{name}, \text{gradyear}, \text{gender}, \text{gpa})$$

and suppose  $\text{dom}(\text{gradyear}) = \{2011, 2012, 2013, 2014\}$  and  $\text{dom}(\text{gender}) = \{M, F\}$ . If the desired workload consists of statistics about the gender of students graduating in specified years, then we can define the cells of  $\mathbf{x}$  as the crossproduct of  $\text{dom}(\text{gradyear})$  and  $\text{dom}(\text{gender})$ , which has size  $n = 8$ . That is,

$$\mathbf{x} = [\text{cnt}(2011, M), \text{cnt}(2011, F), \dots, \text{cnt}(2014, M), \text{cnt}(2014, F)]$$

Then the following matrix represents a workload of five queries:

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

By rows, the queries represented by  $\mathbf{W}$  are:  $Q_1$ : the count of all students;  $Q_2$ : the count of students with  $\text{gradyear} \in [2011, 2012]$ ;  $Q_3$ : the count of female students with  $\text{gradyear} \in [2011, 2012]$ ;  $Q_4$ : the count of male students with  $\text{gradyear} \in [2011, 2012]$ ;  $Q_5$ : the difference between 2013 grads and 2014 grads.

Note that each query of interest should be listed in the workload. We do *not* omit queries whose answers could be calculated from other queries in the workload. For example,  $Q_2$  is included in the workload even though it could be computed by summing  $Q_3$  and  $Q_4$ . Noise accumulates when summing noisy query answers, so we include each desired query and minimize the total error. Also, the relative accuracy of a query in the workload can be controlled by linearly scaling its row by a coefficient.

For this domain, the workload of all range queries is written  $\text{ALLRANGE}(4, 2)$  and consists of all two dimensional range queries over  $\text{gradyear}$  and  $\text{gender}$ , where the possible “ranges” for  $\text{gender}$  are simply  $M$ ,  $F$ , or  $(M \vee F)$ . This workload consists of 30 queries. The workload  $\text{ALLPREDICATE}(8)$  includes all  $2^8$  linear queries expressed over  $\mathbf{x}$  with coefficients in  $\{0, 1\}$ .

Also note that for the workload  $\mathbf{W}$  above, the pair of elements  $x_5, x_6$  and  $x_7, x_8$  always appear together. The included queries do not distinguish between males and females in year 2013, or in year 2014. For this reason, each pair of variables could be replaced by a single variable, reducing the vector  $\mathbf{x}$  to size 6.

## B Comparison of the $\epsilon$ - and $(\epsilon, \delta)$ -Matrix Mechanisms

In this appendix we consider how our main results, which apply to approximate differential privacy, compare to corresponding results under standard differential privacy. We begin with definitions for the  $\epsilon$ -matrix mechanism in Sec. B.1. Then, in Sec. B.2, we describe key differences that make the strategy selection problem harder under  $\epsilon$ -differential privacy. While the privacy guarantees of the two mechanisms are formally distinct, for conservative settings of  $\delta$ , one may be indifferent to the two guarantees and consider which mechanism offers lower error for a fixed  $\epsilon$ . In Sec. B.3 we show that the error of the  $(\epsilon, \delta)$ -matrix mechanism is favorable for a wide range of strategies including those considered in this paper.

### B.1 Definitions, $\epsilon$ -Matrix Mechanism

Standard differential privacy is defined as follows:

**Definition B.1** (Differential Privacy). *A randomized algorithm  $\mathcal{K}$  is  $\epsilon$ -differentially private if for any instance  $I$ , any  $I' \in \text{nbrs}(I)$ , and any subset of outputs  $S \subseteq \text{Range}(\mathcal{K})$ , the following holds:*

$$\Pr[\mathcal{K}(I) \in S] \leq \exp(\epsilon) \times \Pr[\mathcal{K}(I') \in S],$$

$$\begin{array}{cccc}
\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & 
\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} & 
\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & 
\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ \sqrt{2} & 0 & 0 & 0 \\ 0 & \sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 \\ 0 & 0 & 0 & \sqrt{2} \end{bmatrix} \\
\mathbf{H} & \mathbf{Y} & \mathbf{Y}_1 & \mathbf{Y}_2
\end{array}$$

Figure 4: Strategy matrices for domain  $n = 4$ .  $\mathbf{H}$  is the hierarchical strategy.  $\mathbf{Y}$  is the wavelet strategy.  $\mathbf{Y}_1$  has redundant queries which are reduced in  $\mathbf{Y}_2$ .  $\mathbf{Y}, \mathbf{Y}_1, \mathbf{Y}_2$  are all equivalent under the  $(\epsilon, \delta)$ -matrix mechanism. Under the  $\epsilon$ -matrix mechanism,  $\mathbf{Y}_2$  is strictly more efficient than  $\mathbf{Y}_1$  because  $\|\mathbf{Y}_2\|_1 < \|\mathbf{Y}_1\|_1$ .

Under  $\epsilon$ -differential privacy, query sensitivity is measured using the  $L_1$  distance. For a query matrix  $\mathbf{W}$ , the  $L_1$  sensitivity is the maximum  $L_1$  norm of the columns of  $\mathbf{W}$ .

**Proposition B.1** ( $L_1$  Query matrix sensitivity). *The  $L_1$  sensitivity of a query matrix  $\mathbf{W}$  is denoted  $\|\mathbf{W}\|_1$  and is defined as follows:*

$$\|\mathbf{W}\|_1 \stackrel{\text{def}}{=} \max_{\mathbf{x}' \in \text{nbrs}(x)} \|\mathbf{W}\mathbf{x} - \mathbf{W}\mathbf{x}'\|_1 = \max_{w_i \in \text{cols}(\mathbf{W})} \|w_i\|_1$$

The standard mechanism for achieving  $\epsilon$ -differential privacy adds Laplace noise calibrated to the  $L_1$  sensitivity. We use  $\text{Laplace}(b)^m$  to denote a column vector consisting of  $m$  independent samples drawn from a Laplace distribution with mean 0 and scale  $b$ .

**Proposition B.2** (Laplace mechanism). *Given an  $m \times n$  query matrix  $\mathbf{W}$ , the randomized algorithm  $\mathcal{L}$  that outputs the following vector is  $\epsilon$ -differentially private:*

$$\mathcal{L}(\mathbf{W}, \mathbf{x}) = \mathbf{W}\mathbf{x} + \text{Laplace}\left(\frac{\|\mathbf{W}\|_1}{\epsilon}\right)^m$$

The matrix mechanism is defined almost identically to Prop. 2.2, but with Laplace noise in place of Gaussian noise.

**Proposition B.3.** ( $\epsilon$ -MATRIX MECHANISM [10]) *Let  $\mathbf{A}$  be a full rank  $m \times n$  strategy matrix and let  $\mathbf{W}$  be any  $p \times n$  workload matrix. Then the randomized algorithm  $\mathcal{M}_{\mathbf{A}}$  that outputs the following vector is  $\epsilon$ -differentially private:*

$$\mathcal{M}_{\mathbf{A}}(\mathbf{W}, \mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{W}\mathbf{A}^+ \text{Laplace}(b)^m.$$

where  $b = \|\mathbf{A}\|_1 / \epsilon$

The analysis of error for the  $\epsilon$ -matrix mechanism differs only in the sensitivity and  $\epsilon$  terms:

**Proposition B.4.** (TOTAL ERROR) *Given a workload  $\mathbf{W}$ , the total error of answering  $\mathbf{W}$  using the  $\epsilon$  matrix mechanism with query strategy  $\mathbf{A}$  is:*

$$\text{ERROR}_{\mathbf{A}}(\mathbf{W}) = \frac{2}{\epsilon^2} \|\mathbf{A}\|_1^2 \text{trace}(\mathbf{W}^T \mathbf{W} (\mathbf{A}^T \mathbf{A})^{-1}) \quad (4)$$



## B.2 Main Distinctions

The analysis of error differs between the two mechanisms because of the difference in sensitivity metrics. From Prop. B.4 and Prop. 2.4 the two expressions for error can be compared (ignoring privacy parameters):

$$\begin{aligned}\epsilon\text{-ERROR}_{\mathbf{A}}(\mathbf{W}) &\propto \|\mathbf{A}\|_1^2 \text{trace}(\mathbf{W}^T \mathbf{W} (\mathbf{A}^T \mathbf{A})^{-1}) \\ (\epsilon, \delta)\text{-ERROR}_{\mathbf{A}}(\mathbf{W}) &\propto \|\mathbf{A}\|_2^2 \text{trace}(\mathbf{W}^T \mathbf{W} (\mathbf{A}^T \mathbf{A})^{-1})\end{aligned}$$

The trace terms are identical, so these expressions differ only in the sensitivity metric applied to  $\mathbf{A}$ . The implications of this small difference are significant for optimizing the expressions, primarily because  $\|\mathbf{A}\|_2$  is uniquely determined by  $\mathbf{A}^T \mathbf{A}$  (it is in fact the largest diagonal entry of matrix  $\mathbf{A}^T \mathbf{A}$ ). This means that whenever two different strategies  $\mathbf{A}$  and  $\mathbf{B}$  are such that  $\mathbf{A}^T \mathbf{A} = \mathbf{B}^T \mathbf{B}$ , then under  $(\epsilon, \delta)$ -matrix mechanism they have equivalent error, so  $\mathbf{A} \equiv \mathbf{B}$ . But  $\|\mathbf{A}\|_1$  is not determined by  $\mathbf{A}^T \mathbf{A}$ , so it is possible to have  $\mathbf{A}^T \mathbf{A} = \mathbf{B}^T \mathbf{B}$ , while  $\|\mathbf{B}\|_1 < \|\mathbf{A}\|_1$ . Then it follows that under  $\epsilon$ -matrix mechanism  $\mathbf{A}$  and  $\mathbf{B}$  are not equivalent. Instead strategy  $\mathbf{B}$  is strictly more efficient than  $\mathbf{A}$ .

This difference in invalidates many results that hold for the  $(\epsilon, \delta)$ -matrix mechanism. First of all, the optimal strategy under  $(\epsilon, \delta)$ -matrix mechanism can be found via a semi-definite programming (SDP) that computes the matrix  $\mathbf{A}^T \mathbf{A}$  to minimize the total error formula. The solution to this program is insufficient, however, under the  $\epsilon$ -matrix mechanism, because we need to compute an strategy  $\mathbf{A}$  with minimum  $L_1$  sensitivity. To do so, extra non-convex constraints must be added which requires solving a non-convex extension of an SDP (an SDP with rank constraints) to get an optimal strategy.

Further, the singular value bound does not hold under the  $\epsilon$ -matrix mechanism and the properties of redundant queries under  $\epsilon$ -matrix mechanism change. Unlike Thm. 3.2, the presence of redundant queries leads to unneeded error:

**Theorem B.1** (Redundant Queries). *Suppose strategy  $\mathbf{A}_1 = \{\mathbf{A}_0 \cup \mathbf{q} \cup c_1 \mathbf{q}\}$  for some strategy  $\mathbf{A}_0$ , some linear query  $\mathbf{q}$ , and some constant  $c_1$ . Then, under  $\epsilon$ -matrix mechanism, the reduced strategy  $\mathbf{A}_2 = \{\mathbf{A}_0 \cup c_2 \mathbf{q}\}$  where  $c_2 = \sqrt{1 + c_1^2}$  is strictly more efficient than  $\mathbf{A}_1$ .*

Fig. 4 shows an example of a strategy matrix with redundant queries ( $\mathbf{Y}_1$ ) and a reduced strategy with lower  $L_1$  sensitivity.

Finally, column uniformity no longer characterizes the minimal strategies (as in Thm. 3.1) and the quality of the output of the LSA algorithm tends to be lower. Intuitively, in the  $\epsilon$ -matrix mechanism, the algorithm only has one chance to add a query to the strategy and needs to choose the correct weight before knowing how the remainder of the strategy will be built.

## B.3 Error Comparison

Suppose we fix  $\epsilon$  in both mechanisms and choose  $\delta = 2/n^2$  in  $(\epsilon, \delta)$ -matrix mechanism, where  $n$  is the domain size. This is a conservative choice for  $\delta$ , which results in error of the  $(\epsilon, \delta)$ -matrix mechanism equal to:

$$\text{ERROR}_{\mathbf{A}}(\mathbf{W}) = \frac{2}{\epsilon^2} (\log^{\frac{1}{4}} n \|\mathbf{A}\|_2)^2 \text{trace}(\mathbf{W}^T \mathbf{W} (\mathbf{A}^T \mathbf{A})^{-1}).$$

Comparing this equation with Eq. (4), indicates that the  $(\epsilon, \delta)$  mechanism introduces less error whenever  $\|\mathbf{A}\|_1 > \log^{\frac{1}{4}} n \|\mathbf{A}\|_2$ . In particular, for strategy matrices  $\mathbf{A}$  which consists of predicate queries (such as the output of LSA algorithm, hierarchical strategies [9] and (a strategy equivalent to) the wavelet strategy [13]),  $\|\mathbf{A}\|_2 = \sqrt{\|\mathbf{A}\|_1}$ . Using such a strategy matrix  $\mathbf{A}$ , when  $\|\mathbf{A}\|_1 > \sqrt{\log n}$ , the  $(\epsilon, \delta)$ -matrix mechanism provides less error than  $\epsilon$ -matrix mechanism.

## C The Optimal Strategy for Variable Agnostic Workloads

To demonstrate that the singular value bound is a tight lower bound, we discuss a special type of workloads called the variable agnostic workloads and construct a strategy to such workloads that introduce error as low as the singular value bound.

**Definition C.1** (Variable agnostic workload). A

workload  $\mathbf{W}$  is variable agnostic if  $\mathbf{W}^T \mathbf{W}$  is unchanged when we swap two columns of  $\mathbf{W}$ .

For any variable agnostic workloads  $\mathbf{W}$ ,  $\mathbf{W}^T \mathbf{W}$  has the following form, for constants  $a$  and  $b$ :

The next corollary gives a strategies which work on variable agnostic workloads with size  $n$  and have the total error equal to the the singular value bound.

**Theorem C.1.** For positive integer  $k$  and  $n = 2^k$ , let  $\mathbf{W}$  be any  $m \times n$  variable-agnostic workload. Then  $\mathbf{W}^T \mathbf{W}$  has the following form, for constants  $a$  and  $b$ :

$$\begin{bmatrix} a & b & \dots & b \\ b & a & \dots & b \\ \vdots & \vdots & \ddots & \vdots \\ b & b & \dots & a \end{bmatrix}.$$

and there exists a strategy  $\mathbf{A}$  which attains the singular value bound which is  $\text{SVDB}(\mathbf{W}) = \frac{1}{n}(\sqrt{a + (n-1)b} + (n-1)\sqrt{a-b})^2$ .

*Proof.* Let  $\mathbf{Q}_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$  and  $\mathbf{Q}_k = \begin{bmatrix} \mathbf{Q}_{k-1} & \mathbf{Q}_{k-1} \\ \mathbf{Q}_{k-1} & -\mathbf{Q}_{k-1} \end{bmatrix}$ . Here we prove the following result:  $\mathbf{Q}_k$  is an eigenvector matrix for  $\mathbf{W}_k^T \mathbf{W}_k$  where  $\mathbf{W}_k$  is any  $2^k \times 2^k$  variable-agnostic workload. The eigenvalue corresponds to the first column of  $\mathbf{Q}_k$  is  $a + (2^k - 1)b$  and the eigenvalue corresponds to all the other columns is  $a - b$ , where  $a$  is the diagonal entry of  $\mathbf{W}_k^T \mathbf{W}_k$  and  $b$  is the off diagonal entry of  $\mathbf{W}_k^T \mathbf{W}_k$ .

One can verify the result is true when  $k = 1$ . Now suppose the result is true for  $k - 1$ . Then

$$\begin{aligned} \mathbf{W}^T \mathbf{W} \mathbf{Q}_k &= \begin{bmatrix} \mathbf{W}_{k-1} & b\mathbf{J}_{2^{k-1}} \\ b\mathbf{J}_{2^{k-1}} & \mathbf{W}_{k-1} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{k-1} & \mathbf{Q}_{k-1} \\ \mathbf{Q}_{k-1} & -\mathbf{Q}_{k-1} \end{bmatrix} \\ &= \begin{bmatrix} a + (2^k - 1)b & 0 & \dots & 0 \\ 0 & a - b & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a - b \end{bmatrix} \cdot \mathbf{Q}_k \end{aligned} \quad (5)$$

Here  $\mathbf{J}$  is the matrix whose all entries are 1. The computation of Eq. 5 uses the fact that the sum of first column of  $\mathbf{Q}_k$  is  $2^k$  and the sum of any other column of  $\mathbf{Q}_k$  is 0. In addition, since,

$$\mathbf{Q}_k^T \mathbf{Q}_k = \begin{bmatrix} \mathbf{Q}_{k-1} & \mathbf{Q}_{k-1} \\ \mathbf{Q}_{k-1} & -\mathbf{Q}_{k-1} \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{k-1} & -\mathbf{Q}_{k-1} \\ \mathbf{Q}_{k-1} & \mathbf{Q}_{k-1} \end{bmatrix} = \frac{1}{2^k} \mathbf{I},$$

we know  $\mathbf{Q}_k$  is a eigenvector matrix for  $\mathbf{W}^T \mathbf{W}$ .

From above, we know the eigenvalues and a set of corresponding eigenvectors of  $\mathbf{W}^T \mathbf{W}$ . According to the proof of Thm. 3.3, the  $\text{SVDB}(\mathbf{W})$  can be achieved if and only if

$$\begin{bmatrix} \sqrt{a + (2^k - 1)b} & 0 & \dots & 0 \\ 0 & \sqrt{a - b} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sqrt{a - b} \end{bmatrix} \cdot \mathbf{Q}_k$$

is a column-uniform matrix. Notice  $\mathbf{Q}_k^s = \mathbf{J}_{2^k}$ , according to Thm. D.1, the matrix above is column-uniform.  $\square$

The total errors of using the identity matrix or the workload itself as the strategy matrix are both  $na$ . Compared with those total errors, the ratio of total error reduced by using the strategy in Thm. C.1 is approximately  $1 - \frac{b}{a}$ .

Since the workload  $\text{ALLPREDICATE}(n)$  is variable agnostic, a consequence of the above theorem is that we can find its optimal strategy.

**Corollary C.1.** For  $n = 2^k$ , the minimized total error for the workload  $\text{ALLPREDICATE}(n)$  is equal to the singular value bound, which is  $\frac{2^{n-2}}{n}(n - 1 + \sqrt{n + 1})^2$ .

## D Proof of Main Theorems

This section contains proofs of the most important theorems in Section 3 and 4.

### D.1 Analysis of Minimal Strategies

Here we complete the proof characterizing minimal strategies and continue the discussion in order to prove the singular value bound.

**Theorem 3.1.** *A strategy matrix  $\mathbf{A}$  is minimal iff it is column-uniform.*

*Proof.* If a strategy  $\mathbf{A}_1$  is not column-uniform, the strategy  $\mathbf{A}_2$  that is more efficient  $\mathbf{A}_1$  can be found according to the augmenting strategy theorem from [10]. Here we only show the proof that any column-uniform matrix is minimal under the partial order.

Given two column uniform strategies  $\mathbf{A}_1$  and  $\mathbf{A}_2$  such that  $\mathbf{A}_1 \leq \mathbf{A}_2$ . Without loss of generality, we assume both of them have  $L_2$  sensitivity 1, which means all diagonal entries of  $\mathbf{A}_1^T \mathbf{A}_1$  and  $\mathbf{A}_2^T \mathbf{A}_2$  are 1. Since  $\mathbf{A}_1 \leq \mathbf{A}_2$ , for any query  $\mathbf{w}$ ,  $\text{ERROR}_{\mathbf{A}_1}(\mathbf{w}) \leq \text{ERROR}_{\mathbf{A}_2}(\mathbf{w})$ . According to the definition,  $\text{ERROR}_{\mathbf{A}_1}(\mathbf{w}) = \mathbf{w}^T \mathbf{A}_1^T \mathbf{A}_1 \mathbf{w}$ ,  $\text{ERROR}_{\mathbf{A}_2}(\mathbf{w}) = \mathbf{w}^T \mathbf{A}_2^T \mathbf{A}_2 \mathbf{w}$ . Therefore,

$$\begin{aligned} \text{ERROR}_{\mathbf{A}_1}(\mathbf{w}) - \text{ERROR}_{\mathbf{A}_2}(\mathbf{w}) &= \mathbf{w}^T \mathbf{A}_1^T \mathbf{A}_1 \mathbf{w} - \mathbf{w}^T \mathbf{A}_2^T \mathbf{A}_2 \mathbf{w} \\ &= \mathbf{w}^T (\mathbf{A}_1^T \mathbf{A}_1 - \mathbf{A}_2^T \mathbf{A}_2) \mathbf{w} \\ &\leq 0 \end{aligned} \tag{6}$$

Since (6) is true for arbitrary  $\mathbf{w}$ ,  $\mathbf{A}_2^T \mathbf{A}_2 - \mathbf{A}_1^T \mathbf{A}_1$  is a positive semi-definite matrix. In addition, as the assumption, the diagonal entries of  $\mathbf{A}_1^T \mathbf{A}_1$  and  $\mathbf{A}_2^T \mathbf{A}_2$  are 1. Therefore the diagonal entries of  $\mathbf{A}_2^T \mathbf{A}_2 - \mathbf{A}_1^T \mathbf{A}_1$  are all 0. According to properties of positive semi-definite matrix, there is an unique semi-definite matrix whose diagonal entries are all 0, which is the 0 matrix. Thus we have  $\mathbf{A}_1 = \mathbf{A}_2$ .  $\square$

As an application of Thm. 3.1, we have the following theorem.

**Theorem D.1.** *If an  $m \times n$  strategy matrix  $\mathbf{A}$  connects to an optimal solution of Problem 2.1,  $\mathbf{A} \in \mathcal{U}_{m \times n}$ , which is also equivalent to*

$$\mathbf{P}_{\mathbf{A}}^s \lambda_{\mathbf{A}}^s = k \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{7}$$

Here  $\mathbf{A} = \mathbf{Q}_{\mathbf{A}} \Lambda_{\mathbf{A}} \mathbf{P}_{\mathbf{A}}$  is the singular decomposition of  $\mathbf{A}$ , vector  $\lambda_{\mathbf{A}}$  is the vector consists of the singular values of  $\mathbf{A}$ ,  $k$  is a real number and  $\mathbf{A}^s$  represents the element-wise square of a matrix  $\mathbf{A}$ .

For any strategy matrix  $\mathbf{A}$ , one can find an  $n \times n$  matrix  $\mathbf{B}$  such that  $\mathbf{A}^T \mathbf{A} = \mathbf{B}^T \mathbf{B}$  by decomposing matrix  $\mathbf{A}^T \mathbf{A}$ . It follows that  $\text{ERROR}_{\mathbf{A}}(\mathbf{W}) = \text{ERROR}_{\mathbf{B}}(\mathbf{W})$  and therefore it is sufficient to consider only  $n \times n$  strategy matrices in Problem 2.1. Thus (1) can be represented using Frobenius norms, which is defined as following.

**Definition D.1.** (FROBENIUS NORM) *Given an  $m \times n$  matrix  $\mathbf{A} = \{a_{ij}\}$ , the Frobenius norm of matrix  $\mathbf{A}$  is defined as the square root of the sum of squares of its elements*

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2}.$$

It is also equal to the square root of the trace of matrix  $\mathbf{A}^T \mathbf{A}$

$$\|\mathbf{A}\|_F = \sqrt{\text{trace}(\mathbf{A}^T \mathbf{A})}.$$

According to the definition of Frobenius norm, assume strategy matrix  $\mathbf{A}$  is a square matrix, (2) can be rewritten as follows:

$$\begin{aligned} & \min_{\mathbf{A}} \|\mathbf{A}\|_2^2 \text{trace}((\mathbf{A}^T \mathbf{A})^{-1} \mathbf{W}^T \mathbf{W}) \\ &= \min_{\mathbf{A}} \|\mathbf{A}\|_2^2 \text{trace}(\mathbf{W} \mathbf{A}^{-1} (\mathbf{A}^{-1})^T \mathbf{W}^T) \\ &= \min_{\mathbf{A}} \|\mathbf{A}\|_2^2 \|\mathbf{W} \mathbf{A}^{-1}\|_F^2. \end{aligned}$$

**Lemma 1.** *If a workload  $\mathbf{W}$  consists of two sets of independent queries, i.e. there exists an orthogonal matrix  $\mathbf{P}$  such that*

$$\mathbf{P} \mathbf{W} = \begin{bmatrix} \mathbf{W}_1 & 0 \\ 0 & \mathbf{W}_2 \end{bmatrix},$$

*the strategy matrix  $\mathbf{A}$  that minimizes total error has form*

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 \end{bmatrix},$$

*where  $\mathbf{A}_1, \mathbf{A}_2$  are the solutions to problem 2.1 for  $\mathbf{W}_1, \mathbf{W}_2$ , respectively.*

*Proof.* Given a strategy matrix  $\mathbf{A}$ , consider the cholesky decomposition of  $\mathbf{A}^T \mathbf{A}$ , which is in form

$$\mathbf{A}' = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_3 \\ 0 & \mathbf{A}_2 \end{bmatrix}.$$

Then

$$\begin{aligned} \mathbf{W} \mathbf{A}'^{-1} &= \begin{bmatrix} \mathbf{W}_1 & 0 \\ 0 & \mathbf{W}_2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1^{-1} & -\mathbf{A}_1^{-1} \mathbf{A}_3 \mathbf{A}_2^{-1} \\ 0 & \mathbf{A}_2^{-1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{W}_1 \mathbf{A}_1^{-1} & -\mathbf{W}_1 \mathbf{A}_1^{-1} \mathbf{A}_3 \mathbf{A}_2^{-1} \\ 0 & \mathbf{W}_2 \mathbf{A}_2^{-1} \end{bmatrix}. \end{aligned}$$

Let strategy matrix  $\mathbf{A}_d$  be

$$\mathbf{A}_d = \begin{bmatrix} \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 \end{bmatrix}.$$

Notice that

$$\begin{aligned} \|\mathbf{W} \mathbf{A}'^{-1}\|_F^2 &= \|\mathbf{W}_1 \mathbf{A}_1^{-1}\|_F^2 + \|\mathbf{W}_2 \mathbf{A}_2^{-1}\|_F^2 + \|\mathbf{W}_1 \mathbf{A}_1^{-1} \mathbf{A}_3 \mathbf{A}_2^{-1}\|_F^2 \\ &\leq \|\mathbf{W}_1 \mathbf{A}_1^{-1}\|_F^2 + \|\mathbf{W}_2 \mathbf{A}_2^{-1}\|_F^2 = \|\mathbf{W} \mathbf{A}_d^{-1}\|_F^2 \\ \|\mathbf{A}'\|_2^2 &\leq \max\{\|\mathbf{A}_1\|_2^2, \|\mathbf{A}_2\|_2^2\} = \|\mathbf{A}_d\|_2^2, \end{aligned}$$

we know  $\text{ERROR}_{\mathbf{A}_d}(\mathbf{W}) \leq \text{ERROR}_{\mathbf{A}}(\mathbf{W})$ . Thus the strategy matrix that minimizes (1) also has the same form as  $\mathbf{A}_d$ . Furthermore, if  $\mathbf{A}_1$  is not an optimal strategy for workload  $\mathbf{W}_1$ , let  $\mathbf{A}_1^*$  be an optimal strategy for  $\mathbf{W}_1$ . Notice  $\frac{\|\mathbf{A}_1\|_2}{\|\mathbf{A}_1^*\|_2} \mathbf{A}_1^*$  is also an optimal strategy for  $\mathbf{W}_1$  and substitute  $\mathbf{A}_1$  by  $\frac{\|\mathbf{A}_1\|_2}{\|\mathbf{A}_1^*\|_2} \mathbf{A}_1^*$  in  $\mathbf{A}_d$  will bring down the Frobenius norm without boosting  $\|\mathbf{A}_d\|_2$  so that the total error becomes smaller. Thus, if  $\mathbf{A}_d$  is an optimal strategy for  $\mathbf{W}$ ,  $\mathbf{A}_1, \mathbf{A}_2$  must be the solutions to problem 2.1 for  $\mathbf{W}_1, \mathbf{W}_2$ , respectively.  $\square$

**Theorem 3.3.** (SINGULAR VALUE BOUND) *Given an  $m \times n$  workload  $\mathbf{W}$ , let  $\lambda_1, \lambda_2, \dots, \lambda_n$  be the singular values of  $\mathbf{W}$ .*

$$\min_{\mathbf{A}} \text{ERROR}_{\mathbf{A}}(\mathbf{W}) \geq P(\epsilon, \delta) \frac{1}{n} \left( \sum_{i=1}^n \lambda_i \right)^2,$$

where  $P(\epsilon, \delta) = \frac{2 \log(2/\delta)}{\epsilon^2}$ .

*Proof.* For a given workload  $\mathbf{W}$ , according to Thm. D.1, its optimal strategy matrix  $\mathbf{A}$  satisfies  $\mathbf{A} \in \mathcal{U}_{n \times n}$ . Therefore,

$$\|\mathbf{A}\|_2 = \frac{1}{n} \text{trace}(\mathbf{A}).$$

Let  $\mathbf{W} = \mathbf{Q}_\mathbf{W}\mathbf{\Lambda}_\mathbf{W}\mathbf{P}_\mathbf{W}$  and  $\mathbf{A} = \mathbf{Q}_\mathbf{A}\mathbf{\Lambda}_\mathbf{A}\mathbf{P}_\mathbf{W}$  be the singular decomposition of  $\mathbf{W}$  and  $\mathbf{W}$ , respectively, we have:

$$\begin{aligned}
& \text{ERROR}_\mathbf{A}(\mathbf{W}) \\
&= \min_{\mathbf{A} \in \mathcal{U}_{n \times n}} \|\mathbf{A}\|_2 \text{trace}(\mathbf{W}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{W}^T) \\
&= \frac{1}{n} \min_{\mathbf{A} \in \mathcal{U}_{n \times n}} \text{trace}(\mathbf{A}) \text{trace}((\mathbf{A}^T \mathbf{A})^{-1} \mathbf{W}^T \mathbf{W}) \\
&= \frac{1}{n} \min_{(\mathbf{\Lambda}_\mathbf{A} \mathbf{P}_\mathbf{A}) \in \mathcal{U}_{n \times n}} \text{trace}(\mathbf{\Lambda}_\mathbf{A}^2) \\
&\quad \cdot \text{trace}(\mathbf{P}_\mathbf{A}^T (\mathbf{\Lambda}_\mathbf{A}^{-1})^2 \mathbf{P}_\mathbf{A} \mathbf{P}_\mathbf{W}^T \mathbf{\Lambda}_\mathbf{W}^2 \mathbf{P}_\mathbf{W}) \\
&\geq \frac{1}{n} \min_{\mathbf{\Lambda}_\mathbf{A}, \mathbf{P}_\mathbf{A}} \text{trace}(\mathbf{\Lambda}_\mathbf{A}^2) \\
&\quad \cdot \text{trace}(\mathbf{\Lambda}_\mathbf{W} (\mathbf{P}_\mathbf{W}^T \mathbf{P}_\mathbf{A})^T (\mathbf{\Lambda}_\mathbf{A}^{-1})^2 (\mathbf{P}_\mathbf{W}^T \mathbf{P}_\mathbf{A}) \mathbf{\Lambda}_\mathbf{W}) \tag{8}
\end{aligned}$$

As the proof of Lemma 1, the minimum of (8) achieved when  $\mathbf{P}_\mathbf{W}^T \mathbf{P}_\mathbf{A}$  is a diagonal matrix. Thus  $\mathbf{P}_\mathbf{A} = \mathbf{P}_\mathbf{W}$  and

$$\begin{aligned}
(8) &= \frac{1}{n} \min_{\mathbf{\Lambda}_\mathbf{A}} \text{trace}(\mathbf{\Lambda}_\mathbf{A}^2) \text{trace}(\mathbf{\Lambda}_\mathbf{W}^2 (\mathbf{\Lambda}_\mathbf{A}^{-1})^2) \\
&\geq \frac{1}{n} \left( \sum_{i=1}^n \lambda_i \right)^2. \tag{9}
\end{aligned}$$

The equal sign in (9) is satisfied if and only if  $\mathbf{\Lambda}_\mathbf{A} = \sqrt{\mathbf{\Lambda}_\mathbf{W}}$ . Notice the inequality in (8) comes from removing the constraint that  $(\mathbf{\Lambda}_\mathbf{A} \mathbf{P}_\mathbf{A}) \in \mathcal{U}_{n \times n}$ , to satisfied the equal signs in (8) and (9) simultaneously, we need  $\sqrt{\mathbf{\Lambda}_\mathbf{W}} \mathbf{P}_\mathbf{W} \in \mathcal{U}_{n \times n}$ .  $\square$

## D.2 The LSA Algorithm and Extensions.

Lastly, we prove the complexity of the LSA algorithm and the basis of the workload separation technique.

**Lemma 2.** (SHERMAN-MORRISON-WOODBURY FORMULA[6]) *Given  $n \times n$  matrix  $\mathbf{X}$ ,  $n \times m$  matrix  $\mathbf{U}$ ,  $m \times m$  matrix  $\mathbf{\Lambda}$  and  $m \times n$  matrix  $\mathbf{V}$ ,*

$$(\mathbf{X} - \mathbf{U}\mathbf{\Lambda}\mathbf{V})^{-1} = \mathbf{X}^{-1} - \mathbf{X}^{-1}\mathbf{U}(\mathbf{\Lambda} + \mathbf{U}\mathbf{X}^{-1}\mathbf{V})^{-1}\mathbf{V}\mathbf{X}^{-1} \tag{10}$$

Using this lemma, the test of each split point in step 6 of Program 4.1 can be done in  $O(n^2)$  time, resulting in the following overall running time:

**Theorem 4.1.** *The LSA algorithm can be implemented in  $O(kn^4)$  time, where  $k$  is the maximum number of levels.*

*Proof.* Notice in step 6 of Program 4.1, each time we compute the total error of a possible split of row  $[v_1, v_2, \dots, v_n]$  on position  $i$ , it is equivalent to modify matrix  $\mathbf{A}'$  by removing this selected row and add the following two rows to matrix  $\mathbf{A}'$ .

$$\begin{bmatrix} v'_1 & v'_2 & \dots & v'_n \\ v''_1 & v''_2 & \dots & v''_n \end{bmatrix}, \quad v'_i, v''_i \in \{0, 1\}, v'_i + v''_i = 1, 1 \leq i \leq n.$$

To apply Lemma 2, let  $\mathbf{X} = \mathbf{A}'^T \mathbf{A}'$  and  $\mathbf{U}, \mathbf{\Lambda}, \mathbf{V}$  be the following matrices:

$$\mathbf{U} = \mathbf{V} = \begin{bmatrix} v_1 & v_2 & \dots & v_n \\ v'_1 & v'_2 & \dots & v'_n \\ v''_1 & v''_2 & \dots & v''_n \end{bmatrix}, \mathbf{\Lambda} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Denote the modified matrix as  $\mathbf{A}''$  and we can verify that  $\mathbf{A}''^T \mathbf{A}'' = \mathbf{A}'^T \mathbf{A}' - \mathbf{U}\mathbf{\Lambda}\mathbf{V} = \mathbf{X} - \mathbf{U}\mathbf{\Lambda}\mathbf{V}$ . Therefore the inverse of  $\mathbf{A}''^T \mathbf{A}''$  can be computed as the Eq. (10). Since  $\mathbf{U}$  and  $\mathbf{V}$  are  $n \times 3$  and  $3 \times n$  matrices respectively, the inverse of  $\mathbf{A}''^T \mathbf{A}''$  can be computed in  $O(n^2)$  time by first computing  $\mathbf{X}^{-1}\mathbf{U}$ ,  $\mathbf{U}\mathbf{X}\mathbf{V}$  and  $\mathbf{V}\mathbf{X}^{-1}$  and then finishing the evaluation from left to right.  $\square$

**Theorem 4.2.** Let  $\mathbf{W}_1, \dots, \mathbf{W}_k$  be sets of predicate queries over one-way marginals on  $k$  different dimensions. Let  $\mathbf{q}_0$  be the sum of all the entries on the domain. With the answer of  $\mathbf{q}_0$  given, the estimate of any query in  $\mathbf{W}_i$  does not involve queries in  $\mathbf{W}_1, \dots, \mathbf{W}_{i-1}, \mathbf{W}_{i+1}, \dots, \mathbf{W}_k$ .

*Proof.* For any  $i$ ,  $1 \leq i \leq k$  and a query  $\mathbf{q} \in \mathbf{W}_i$ . To estimate  $\mathbf{q}$ , one needs to find a set of queries whose linear combination is equal to  $\mathbf{q}_0$ . Let the representation be

$$\mathbf{q} = \sum_{j=1}^l \alpha_j \mathbf{q}_j + \sum_{j=1}^h \beta_j \mathbf{p}_j,$$

where  $\mathbf{q}_j \in \mathbf{W}_i$ ,  $1 \leq j \leq l$  and  $\mathbf{p}_j \in \bigcup_{t=1, t \neq i}^n \mathbf{W}_t$ ,  $1 \leq j \leq h$ . The equation above is equivalent to

$$\mathbf{q} - \sum_{j=1}^l \alpha_j \mathbf{q}_j = \sum_{j=1}^h \beta_j \mathbf{p}_j. \quad (11)$$

Notice the left hand side of Eq. (11) is the sum of one-way marginal queries on  $i$ -th dimension and the right hand side of Eq. (11) is the sum of one-way marginal queries that are not on  $i$ -th dimension. Since the only query that shared by the one-way marginal queries on  $i$ -th and not on  $i$ -th dimension is the total sum  $\mathbf{q}_0$ ,

$$\mathbf{q} - \sum_{j=1}^l \alpha_j \mathbf{q}_j = \alpha_0 \mathbf{q}_0,$$

where  $\alpha_0$  is a constant. Since the answer of  $\mathbf{q}_0$  is given, estimate with  $\alpha_0 \mathbf{q}_0$  always have better accuracy than with the combination of  $\sum_{j=1}^h \beta_j \mathbf{p}_j$  (which are noisy answers). Therefore estimating  $\mathbf{q}$  only relates to queries in  $\mathbf{W}_i$  and  $\mathbf{q}_0$  but not queries in  $\mathbf{W}_1, \dots, \mathbf{W}_{i-1}, \mathbf{W}_{i+1}, \dots, \mathbf{W}_k$ .  $\square$