

Optimizing Batch Linear Queries under Exact and Approximate Differential Privacy

Ganzhao Yuan, South China University of Technology, China

Zhenjie Zhang, Advanced Digital Sciences Center, Singapore

Marianne Winslett, Advanced Digital Sciences Center, Singapore; University of Illinois at Urbana-Champaign, USA

Xiaokui Xiao, Nanyang Technological University, Singapore

Yin Yang, Hamad Bin Khalifa University, Qatar

Zhifeng Hao, South China University of Technology, China; Guangdong University of Technology, China

Differential privacy is a promising privacy-preserving paradigm for statistical query processing over sensitive data. It works by injecting random noise into each query result, such that it is provably hard for the adversary to infer the presence or absence of any individual record from the published noisy results. The main objective in differentially private query processing is to maximize the accuracy of the query results, while satisfying the privacy guarantees. Previous work, notably [Li et al. 2010], has suggested that with an appropriate strategy, processing a batch of correlated queries as a whole achieves considerably higher accuracy than answering them individually. However, to our knowledge there is currently no practical solution to find such a strategy for an arbitrary query batch; existing methods either return strategies of poor quality (often worse than naive methods) or require prohibitively expensive computations for even moderately large domains. Motivated by this, we propose low-rank mechanism (LRM), the first practical differentially private technique for answering batch linear queries with high accuracy. LRM works for both exact (i.e., ϵ -) and approximate (i.e., (ϵ, δ) -) differential privacy definitions. We derive the utility guarantees of LRM, and provide guidance on how to set the privacy parameters given the user's utility expectation. Extensive experiments using real data demonstrate that our proposed method consistently outperforms state-of-the-art query processing solutions under differential privacy, by large margins.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications—Statistical databases

General Terms: Theory, Algorithms, Experimentation

Additional Key Words and Phrases: Linear Counting Query, Differential Privacy, Low-Rank, Matrix Approximation, Augmented Lagrangian Multiplier Algorithm

ACM Reference Format:

G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, Z. Hao. Optimizing Batch Linear Queries under Exact and Approximate Differential Privacy ACM Trans. Datab. Syst. V, N, Article A (January YYYY), 45 pages.

DOI = 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

Yuan is supported by NSF-61402182. Zhang, Winslett, Xiao are supported by SERC 102-158-0074 from Singapore's A*STAR. Xiao is also supported by SUG Grant M58020016 from Nanyang Technological University and AcRF Tier 2 Grant ARC19/14 from Ministry of Education, Singapore. Hao is supported by NSF-(61100148, 61202269, 61472089), Key Technology Research and Development Programs of Guangdong Province (2010B050400011, 2013B051000076), and Science & Technology Plan Project of Guangzhou City (201200000031, 2013Y2-00034, 2014Y2-00027). We thank Li and Miklau for sharing us code for the Adaptive Mechanism, and the anonymous referees for their valuable comments.

Author's addresses: Yuan (corresponding author), Department of Mathematics, South China University of Technology, Guangzhou, China, ecgzhyuan@scut.edu.cn. Zhang, Advanced Digital Sciences Center, Singapore, zhenjie@adsc.com.sg. Winslett, Advanced Digital Sciences Center, Singapore; Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA, winslett@illinois.edu. Xiao, School of Computer Engineering, Nanyang Technological University, Singapore, xkxiao@ntu.edu.sg. Yang, Division of Information and Communication Technologies, College of Science, Engineering and Technology, Hamad Bin Khalifa University, Qatar, yyang@qf.org.qa. Hao, Faculty of Computer Science, Guangdong University of Technology, China; School of Computer Science and Engineering, South China University of Technology, China, mazfhao@scut.edu.cn.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© YYYY ACM 0362-5915/YYYY/01-ARTA \$15.00

DOI 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

1. INTRODUCTION

Differential privacy [Dwork et al. 2006c] is an emerging paradigm for publishing statistical information over sensitive data, with strong and rigorous guarantees on individuals' privacy. Since its proposal, differential privacy has attracted extensive research efforts, such as in cryptography [Dwork et al. 2006c], algorithms [Dwork et al. 2010; Hardt and Talwar 2010; McSherry and Talwar 2007], database management [Ding et al. 2011; Hay et al. 2010; Li et al. 2010; Rastogi and Nath 2010; Xiao et al. 2011; Xiao et al. 2010; Peng et al. 2013], data mining [Bhaskar et al. 2010; Friedman and Schuster 2010], social network analysis [Rastogi et al. 2009; Hay et al. 2009; Sala et al. 2011] and machine learning [Blum et al. 2008; Chaudhuri et al. 2011; Rubinfeld et al. 2012]. The main idea of differential privacy is to inject random noise into aggregate query results, such that the adversary cannot infer, with high confidence, the presence or absence of any given record r in the dataset, even if the adversary knows all other records in the dataset besides r . The adversary's maximum confidence in inferring private information is controlled by a user-specified parameter ϵ , called the *privacy budget*. Given ϵ , the main goal of query processing under differential privacy is to maximize the utility/accuracy of the (noisy) query answers, while satisfying the above privacy requirements.

This work focuses on a common class of queries called *linear counting queries*, which is the basic operation in many statistical analyses. Similar ideas apply to other types of linear queries, e.g., linear sums. Figure 1(a) illustrates an example electronic medical record database, where each record corresponds to an individual. Figure 1(b) shows the exact number of HIV+ patients in each state, which we refer to as *unit counts*. A linear counting query in this example can be any linear combination of the unit counts. For instance, let $x_{NY}, x_{NJ}, x_{CA}, x_{WA}$ be the patient counts in states NY, NJ, CA, and WA respectively; one possible linear counting query is $x_{NY} + x_{NJ} + x_{CA} + x_{WA}$, which computes the total number of HIV+ patients in the four states listed in our example. Another example linear counting query is $x_{NY}/19 + x_{NJ}/8 + x_{CA}/37$, which calculates the weighted average of patient counts in states NY, NJ and CA, with weights set according to their respective population sizes. In general, we are given a database with n unit counts, and a batch QS of m linear counting queries. The goal is to answer all queries in QS under differential privacy, and maximize the expected overall accuracy of the queries.

Name	State	HIV+
Alice	NY	Yes
Bob	NJ	Yes
Carol	NY	Yes
Dave	CA	Yes
...		

State	# of HIV+ patients
NY	82,700
NJ	19,000
CA	67,000
WA	5,900
...	

(a) Patient records

(b) Statistics on HIV+ patients

Fig. 1. Example medical record database

Straightforward approaches to answering a batch of linear counting queries usually lead to sub-optimal result accuracy. Consider processing the query set $Q = \{q_1, q_2, q_3\}$ under the ϵ -differential privacy definition, detailed in Section 3. One naive solution, referred to as *noise on result (NOR)*, is to process each query independently, e.g., using the Laplace mechanism [Dwork et al. 2006c]. This method fails to exploit the *correlations* between different queries. Consider a batch of three different queries $q_1 = x_{NY} + x_{NJ} + x_{CA} + x_{WA}$, $q_2 = x_{NY} + x_{NJ}$, $q_3 = x_{CA} + x_{WA}$. Clearly, the three queries are correlated since $q_1 = q_2 + q_3$. Thus, an alternative strategy for answering these queries is to process only q_2 and q_3 , and use their sum to answer q_1 . As will be explained in Section 3, the amount of noise added to query results depends upon the *sensitivity* of the query set, which is defined as the maximum possible total change in query results caused by adding or

removing a single record in the original database. Under ϵ -differential privacy, the sensitivity of the query set $\{q_2, q_3\}$ is 1, because adding/removing a patient record in Figure 1(a) affects at most one of q_2 and q_3 (i.e., q_2 if the record is associated with state NY or NJ, and q_3 if the state is CA or WA), by exactly 1. On the other hand, the query set $\{q_1, q_2, q_3\}$ has a sensitivity of 2 (under the ϵ -differential privacy definition), since a record in the above 4 states affects both q_1 and one of q_2 and q_3 . According to the Laplace mechanism, the variance of the added noise to each query is $2\Delta^2/\epsilon^2$, where Δ is the sensitivity of the query set, and ϵ is the user-specified privacy budget. Therefore, processing $\{q_1, q_2, q_3\}$ directly incurs a noise variance of $(2 \times 2^2)/\epsilon^2$ for each query; on the other hand, executing $\{q_2, q_3\}$ leads to a noise variance of $(2 \times 1^2)/\epsilon^2$ for each of q_2 and q_3 , and their sum $q_1 = q_2 + q_3$ has a noise variance of $(2 \times 2)/\epsilon^2 = 4/\epsilon^2$. Clearly, the latter method obtains higher accuracy for all queries.

Another simple solution, referred to as *noise on data (NOD)*, is to process each unit count under differential privacy, and combine them to answer the given linear counting queries. Continuing the example, this method computes the noisy counts for x_{NY} , x_{NJ} , x_{CA} and x_{WA} , and uses their linear combinations to answer q_1 , q_2 , and q_3 . This approach overlooks the correlations between different unit counts. In our example, x_{NY} and x_{NJ} (and similarly, x_{CA} and x_{WA}) are either both present or both absent in every query, and, thus, can be seen as a single entity. Processing them as independent queries incurs unnecessary accuracy costs when re-combining them. In the example, NOD adds noise with variance $2/\epsilon^2$ to each unit count, and their combinations to answer q_1 , q_2 , and q_3 have noise variance $8/\epsilon^2$, $4/\epsilon^2$ and $4/\epsilon^2$, respectively. NOD's result utility is also worse than the above-mentioned strategy of processing q_2 and q_3 , and adding their results to answer q_1 .

In general, the query set Q may exhibit complex correlations among different queries and among different unit counts. As a consequence, it is non-trivial to obtain the best strategy to answer Q under differential privacy. For instance, consider the following query set:

$$\begin{aligned} q_1 &= 2x_{NJ} + x_{CA} + x_{WA} \\ q_2 &= x_{NJ} + 2x_{WA} \\ q_3 &= x_{NY} + 2x_{CA} + 2x_{WA} \end{aligned}$$

NOR is clearly a poor choice, since it incurs a sensitivity of 5 under the ϵ -differential privacy definition (e.g., a record of state WA affects q_1 by 1, and q_2 and q_3 by 2 each). The sensitivity of NOD remains 1, and it answers q_1 , q_2 , and q_3 with noise variance $2 \times (2^2 + 1^2 + 1^2)/\epsilon^2$, $2 \times (1^2 + 2^2)/\epsilon^2$ and $2 \times (1^2 + 2^2 + 2^2)/\epsilon^2$ respectively, leading to a sum-square error (SSE) of $40/\epsilon^2$. The optimal strategy in terms of SSE in this case computes the noisy results of $q'_1 = x_{NY}/8 + x_{WA}$, $q'_2 = -3x_{NY}/8 - x_{CA}$ and $q'_3 = x_{NY}/4 - x_{NJ}$. Then, it obtains the results for q_1 , q_2 , and q_3 as follows.

$$\begin{aligned} q_1 &= q'_1 - q'_2 - 2q'_3 \\ q_2 &= 2q'_1 - q'_3 \\ q_3 &= 2q'_1 - 2q'_2 \end{aligned}$$

The sensitivity of the above method is also 1, because (i) adding/removing a record of state NJ, CA and WA can only affect queries q'_3 , q'_2 and q'_1 , respectively, by at most 1; (ii) adding/removing a record of state NY causes the results of q'_1 , q'_2 and q'_3 to change by at most $1/8$, $3/8$, and $1/4$, respectively, leading to a maximum total change of $1/8 + 3/8 + 1/4 = 1$. We introduce the formal definition of sensitivity later in Section 3. Hence, independent random noise of variance $2 \times 1^2/\epsilon^2 = 2/\epsilon^2$ is injected to the results of each of q'_1 , q'_2 and q'_3 . Their combination $q_1 = q'_1 - q'_2 - 2q'_3$ thus has a noise variance of $2 \times (1^2 + (-1)^2 + (-2)^2)/\epsilon^2 = 12/\epsilon^2$. Similarly, combining $q'_1 - q'_3$ to answer q_2 and q_3 as above incur a noise variance of $2 \times (2^2 + (-1)^2)/\epsilon^2 = 10/\epsilon^2$ and $2 \times (2^2 + (-2)^2)/\epsilon^2 = 16/\epsilon^2$ respectively. The SSE for queries $q_1 - q_3$ is thus $12/\epsilon^2 + 10/\epsilon^2 + 16/\epsilon^2 = 38/\epsilon^2$.

Observe that there is no simple pattern in the query set or the optimal strategy. Since there is an infinite space of possible strategies, searching for the best one is a challenging problem.

Li et al. [Li et al. 2010] first formalize the above observations (i.e., answering a correlated query set with an effective strategy) into the *matrix mechanism*. However, as we explain in Section 2.2, the original matrix mechanism lacks a practical implementation, because the solutions in [Li et al. 2010] for finding a good strategy are either inefficient (which incur prohibitively high computational costs for even moderately large domains), or ineffective (which rarely obtain strategies that outperform naive methods NOD/NOR). Later, Li and Miklau [Li and Miklau 2012] propose the adaptive mechanism, which can be seen as an implementation of the matrix mechanism. This method, however, still incurs some drawbacks as discussed in Section 2.2, which limit its accuracy. Motivated by this, we propose the first practical realization of the matrix mechanism, called the *low-rank mechanism (LRM)*, based on the theory of low-rank matrix approximation. LRM applies to both ϵ -differential privacy and (ϵ, δ) -differential privacy, two most commonly used differential privacy definitions today. We analyze the utility of LRM under (ξ, η) -usefulness [Blum et al. 2008], a popular utility measure. Extensive experiments demonstrate that LRM significantly outperforms existing solutions in terms of result accuracy, sometimes by orders of magnitude.

The rest of the paper is organized as follows. Section 2 reviews previous studies on differential privacy. Section 3 provides formal definitions for our problem. Section 4 presents the mechanism formulation of LRM under ϵ -differential privacy. Section 5 discusses how to solve the optimization problem in LRM. Section 6 extends LRM to answer queries under (ϵ, δ) -differential privacy. Section 7 verifies the superiority of our proposal through an extensive experimental study. Finally, Section 8 concludes the paper.

2. RELATED WORK

Section 2.1 surveys general-purpose mechanisms for enforcing differential privacy. Section 2.2 presents two methods that are closely related to the proposed solution, namely the matrix mechanism and the adaptive mechanism.

2.1. Differential Privacy Mechanisms

Differential privacy was first formally presented in [Dwork et al. 2006c], though some previous studies have informally used similar models, e.g., [Dinur and Nissim 2003]. The Laplace mechanism [Dwork et al. 2006c] is the first generic mechanism for enforcing differential privacy, which works when the output domain is a multi-dimensional Euclidean space. McSherry and Talwar [McSherry and Talwar 2007] propose the exponential mechanism, which applies to any problem with a measurable output space. The generality of the exponential mechanism makes it an important tool in the design of many other differentially private algorithms, e.g., [Cormode et al. 2012; Xu et al. 2012; Xu et al. 2013; McSherry and Talwar 2007].

The original definition of differential privacy is ϵ -differential privacy, which focuses on providing a strong and rigorous definition of privacy. Besides this, another popular definition is (ϵ, δ) -differential privacy, which can be seen as an approximate version of ϵ -differential privacy. In many applications, (ϵ, δ) -differential privacy provides a similarly strong privacy definition, while enabling simpler and/or more accurate algorithms. One basic mechanism for enforcing (ϵ, δ) -differential privacy is the Gaussian mechanism, which injects Gaussian noise to the query results calibrated to the \mathcal{L}_2 sensitivity of the queries [Dwork et al. 2006a]. [Hardt and Roth 2012] employ k Gaussian measurements strategy to compute the low rank approximations of large matrices. However, (ϵ, δ) -differential privacy might be unsatisfactory in certain situations. For example, [De 2012] demonstrate that (ϵ, δ) -differential privacy is weaker than ϵ -differential privacy in terms of mutual information even when δ is negligible. The proposed solution applies to both definitions of differential privacy. We present details of these two privacy definitions in Section 3.

Linear query processing is of particular interest in both the theory and database communities, due to its wide range of applications. To minimize the error of linear queries under differential privacy requirements, several methods try to build a synopsis of the original database, such as Fourier transformations [Rastogi and Nath 2010], wavelets [Xiao et al. 2010] and hierarchical trees [Hay et al. 2010]. The compressive mechanism [Li et al. 2011] reduces the amount of noise neces-

sary to satisfy differential privacy, for datasets with a sparse representation. By publishing a noisy synopsis under ϵ -differential privacy, these methods are capable of answering an arbitrary number of linear queries. However, most of these methods obtain good accuracy only when the query selection criterion is a continuous range; meanwhile, since these methods are not workload-aware, their performance for a specific workload tends to be sub-optimal.

Workload-aware algorithms address this problem, which optimize the overall accuracy of a set of given linear queries. This work falls into this category. Notable workload-aware methods include (i) Multiplicative Weights / Exponential Mechanism (MWEM) [Hardt et al. 2012], (ii) the Matrix Mechanism [Li et al. 2010] and (iii) the Adaptive Mechanism [Li and Miklau 2012]. MWEM publishes a synthetic dataset optimized towards the given linear query set. In particular, it provides a beautiful theoretical bound on the maximum error of the given queries, which grows sublinearly to the number of records in the dataset, and logarithmically with the number of queries. In practice, however, this bound tends to be loose as it is derived from worst-case scenarios. Meanwhile, the target problem of MWEM is different from ours, as we focus on answering a given set of linear queries rather than publishing synthetic data. Nevertheless, MWEM can be applied to our problem, and we compare it against the proposed solution in the experiments. The Matrix Mechanism and the Adaptive Mechanism share some common features as the proposed solution, and we explain them in detail in Section 2.2. It is worth mentioning that as our experiments shows, the proposed solution outperforms all previous methods in terms of overall error, on a variety of datasets and workload types.

Recently, [Nikolov et al. 2013] proposes a workload decomposition method that injects *correlated* Gaussian noise to the query results to satisfy (ϵ, δ) -differential privacy. They prove that their solution provides an $\mathcal{O}((\log m)^2)$ approximation to the optimal mechanism, where m is the number of queries. However, this method is infeasible in practice, since it involves computing minimum enclosing ellipsoids (MEE), for which the current best algorithm takes $m^{\mathcal{O}(m)}n$ time, where n is the number of unit counts. [Nikolov et al. 2013] suggests using approximation method for computing MEE, e.g. Khachiyan's algorithm [Todd and Yildirim 2007]. This approximation algorithm still takes high order polynomial time to converge, which makes it prohibitively expensive for practical applications.

Several theoretical studies have derived lower bounds for the noise level for processing linear queries under differential privacy [Dinur and Nissim 2003; Hardt and Talwar 2010]. Notably, Dinur and Nissim [Dinur and Nissim 2003] prove that any perturbation mechanism with maximal noise of scale $\mathcal{O}(n)$ cannot possibly preserve personal privacy, if the adversary is allowed to ask all possible linear queries, and has exponential computation capacity. By reducing the computation capacity of the adversary to polynomial-bounded Turing machines, they show that an error scale $\Omega(\sqrt{n})$ is necessary to protect any individual's privacy. More recently, Hardt and Talwar [Hardt and Talwar 2010] have significantly tightened the error lower bound for answering a batch of linear queries under differential privacy. Given a batch of m linear queries, they prove that any ϵ -differential privacy mechanism leads to squared error of at least $\Omega(\epsilon^{-2}m^3\text{Vol}(W))$, where $\text{Vol}(W)$ is the volume of the convex body obtained by transforming the \mathcal{L}_1 -unit ball into m -dimensional space using the linear transformations in the workload W . This paper extends their analysis to low-rank workload matrices.

Another related line of research concerns answering queries *interactively* under differential privacy. In this setting, the system process queries one at a time, without knowing any future query. Clearly, this problem is more difficult than the non-interactive setting described so far, where the system knows all queries in the workload in advance. Most notably, Hardt et al. propose the Private Multiplicative Weights Mechanism (PMWM) [Hardt and Rothblum 2010], whose error is asymptotically optimal with respect to the number of queries answered. The MWEM method described above [Hardt et al. 2012] applies similar ideas to the non-interactive setting. Besides PMWM, Hardt et al. [Hardt and Talwar 2010] propose the K -norm Mechanism whose error level almost reaches the lower bound derived in the same paper. Roth et al. introduce the Median Mechanism

[Roth and Roughgarden 2010] for answering arbitrary queries interactively. However, both the K -norm Mechanism and the Median Mechanism rely on uniform sampling in a high-dimensional convex body [Dyer et al. 1991], which theoretically takes polynomial time, but is usually too expensive to be applied in practice.

Besides linear queries, differential privacy is also applicable to more complex queries in various research areas, due to its strong privacy guarantee. In the field of data mining, Friedman and Schuster [Friedman and Schuster 2010] propose the first algorithm for building a decision tree under differential privacy. Mohammed et al. [Mohammed et al. 2011] study the same problem, and propose an improved solution based on a generalization strategy coupled with the exponential mechanism. Ding et al. [Ding et al. 2011] investigate the problem of differentially private data cube publication. They present a randomized materialized view selection algorithm, which reduces the overall error, and preserves data consistency.

In the database literature, a plethora of methods have been proposed to optimize the accuracy of differentially private query processing. A tutorial on database-related differential privacy technologies can be found in [Yang et al. 2012]. Cormode et al. [Cormode et al. 2012] investigate the problem of multi-dimensional indexing under differential privacy, with the novel idea of assigning different amounts of privacy budget to different levels of the index. Peng et al. [Peng et al. 2012] propose the DP-tree, which obtains improved accuracy for higher dimensional data. Xu et al. [Xu et al. 2012; Xu et al. 2013] optimize the procedure of building a differentially private histogram, whose method combines dynamic programming for optimal histogram computation and the exponential mechanism. [Li et al. 2012] study the problem of how to perform frequent itemset mining on transaction databases while satisfying differential privacy, with the novel approach of constructing a basis set and then using it to find the most frequent patterns.

In addition, differential privacy for modeling security in social networks has also received much attention in recent literature. [Rastogi et al. 2009] considers answering subgraph counting queries in a social network. Their solution assumes a Bayesian adversary whose prior is drawn from a distribution. They compute a high probability upper bound on the local sensitivity of the data and then answer by adding noise proportional to that bound. [Hay et al. 2009] shows how to privately approximate the degree distribution in the edge adjacency model of a graph. Also, [Sala et al. 2011] develop a differentially private graph model based on dk -series reconstruction. Their approach mainly extracts a graph's detailed structure into degree correlation statistics and inject noise into the resulting dataset and generates a synthetic graph.

Lastly, differential privacy is also becoming a hot topic in the machine learning community, especially for learning tasks involving sensitive information, e.g., medical records. In [Chaudhuri et al. 2011], Chaudhuri et al. propose a generic differentially private learning algorithm, which requires strong convexity of the objective function. Rubinstein et al. [Rubinstein et al. 2012] study the problem of SVM learning on sensitive data, and propose an algorithm to perturb the kernel matrix with performance guarantees, when the gradient of the loss function satisfies the Lipschitz continuity property. Zhang et al. propose functional mechanism and for a large class of optimization-based analyses [Zhang et al. 2012]. Later, they propose the PrivGene framework, which combines genetic algorithms and an enhanced version of exponential mechanism for differentially private model fitting [Zhang et al. 2013]. General differential privacy techniques have also been applied to real systems, such as network trace analysis [McSherry and Mahajan 2010] and private recommender systems [McSherry and Mironov 2009].

2.2. Matrix Mechanism and Adaptive Mechanism

In the seminal work of [Li et al. 2010], Li et al. propose the matrix mechanism (MM), which formalizes the intuition that a batch of correlated linear queries can be answered more accurately under ϵ -differential privacy, by processing a different set of queries (called the *strategy*) and combining their results. Specifically, given a workload of linear counting queries, MM first constructs a *workload matrix* W of size $m \times n$, where m is the number of queries, and n is the number of unit counts. The construction of the workload matrix is elaborated further in Section 3. After that, MM searches

for a *strategy matrix* A of size $r \times n$, where r is a positive integer. Intuitively, A corresponds to another set of linear queries, such that every query in W can be expressed as a linear combination of the queries in A . The matrix mechanism then answers the queries in A under ϵ -differential privacy, and subsequently uses their noisy results to answer queries in W .

The main challenge for applying the matrix mechanism to practical workloads is to identify an appropriate strategy matrix A . Ref. [Li et al. 2010] provides two algorithms for this purpose. The first, based on iteratively solving a pair of related semidefinite programs, incurs $\mathcal{O}(m^3n^3)$ computational overhead, which is prohibitively expensive even for moderately large values of m and n . The second solution (called *approximate matrix mechanism (AMM)*) computes an \mathcal{L}_2 approximation of the optimal strategy matrix A . This method, though faster than the first one, still requires high CPU costs and memory consumption, and scales poorly with the domain size and query set cardinality. In order to test the approximate matrix mechanism with large data and query sets in our experiments, we have devised an improved solution, which we call the *exponential smoothing mechanism (ESM)*, based on the problem formulation of approximate matrix mechanism in [Li et al. 2010]. ESM is at least as accurate as the method in [Li et al. 2010], and yet much more efficient. Hence, in our experiments we use ESM in place of AMM. Appendix A.1 provides details of ESM.

There are, however, two main drawbacks of ESM (and also vanilla AMM). First, the \mathcal{L}_2 approximation of the optimal strategy matrix often has poor quality. In fact, due to this problem, in our experiments we found that under ϵ -differential privacy, the accuracy of ESM is often no better than the naive solution NOD that injects noise directly into the unit counts. A second and more subtle problem is that the formulation of the optimization program in AMM involves matrix inverse operators, which can cause numerical instability when the final solution (i.e., the strategy matrix) is of low rank, as explained in Appendix A.1. The proposed low-rank mechanism avoids both problems, and achieves significantly higher result accuracy as shown in our experiments.

The idea of matrix mechanism naturally extends to (ϵ, δ) -differential privacy, using the Gaussian mechanism instead of the Laplace mechanism as the fundamental building block. In this case, the optimization program is defined using \mathcal{L}_2 form, and the AMM formulation is equivalent to that of MM, meaning that AMM and ESM now solve the exact optimization program. Hence, in theory, AMM can obtain optimal results. However, in practice, both ESM and the AMM implementation in [Li et al. 2010] often fail to converge to the optimal strategy matrix, due to numerical instability incurred by the matrix inverse operator in the AMM formulation.

Recently, [Li and Miklau 2012] Li et al. propose another implementation of AMM, called the adaptive mechanism (AM). For any given workload W , AM attempts to find the best strategy matrix by computing the optimal nonnegative weights for the eigenvectors of the workload matrix W . Since the strategy matrix may have one or more columns whose \mathcal{L}_2 -norm are less than the sensitivity, they refine the strategy matrix by appending some completing columns to the candidate strategy matrix without raising the sensitivity. Therefore, this post-processing step can reduce the expected error. AM incurs two serious drawbacks. First, it involves solving a complicated semidefinite program, and it is not known whether their solution to the program converges to the optimal solution. Second and more importantly, such multistep strategy in AM does not offer any guarantee on optimality. The proposed method LRM is free from these problems, and obtains significantly better performance as we show in the experiments. Appendix A.2 provides details of AM.

3. PRELIMINARIES

We focus on answering a batch of linear counting queries $Q = \{q_1, q_2, \dots, q_m\}$ over a sensitive database D . Each query $q_i \in Q$ is a linear combination of *unit counts* in the data domain, denoted as x_1, x_2, \dots, x_n . In the example of Figure 1, the sensitive database D contains records corresponding to individual HIV+ patients; each unit count is the number of such patients in a state of the US; each query in the example is a linear combination of these state-level patient counts. Our goal is to answer Q with minimum overall error, while satisfying differential privacy. In particular, we consider two definitions of differential privacy, namely ϵ -differential privacy (i.e., the original definition of differential privacy) and (ϵ, δ) -differential privacy (a popular formulation of approximate

Table I. Summary of frequent notations

Symbol	Meaning
D	input database
n	number of unit counts
Q	input query set
m	number of queries in Q
W	workload matrix, i.e., the matrix representation of Q
B, L	a decomposition of W satisfying $W \approx B \cdot L$
s	rank of workload matrix W
r	number of columns in B (also number of rows in L)
$Q(D)$	exact answer of Q on database D
$\Delta(Q)$	\mathcal{L}_1 sensitivity of Q
$\Theta(Q)$	\mathcal{L}_2 sensitivity of Q
ϵ, δ	privacy parameters
ξ, η	utility parameters
$\kappa(W)$	generalized condition number of matrix W
$\rho(W)$	ρ -coherence of matrix W
$\ X\ _1$	maximum absolute column sum of matrix X
$\ X\ _2$	spectral norm, maximum singular value of matrix X
$\ X\ _\infty$	maximum absolute row sum of matrix X
$\ X\ _*$	nuclear norm, sum of the singular values of matrix X
$\ X\ _F$	Frobenius norm, square root of the sum of squared elements of matrix X

differential privacy). Our solutions use the Laplace mechanism (resp., the Gaussian mechanism) as a fundamental building block to enforce ϵ - (resp., (ϵ, δ) -) differential privacy. In the following, Section 3.1 presents the definition of ϵ -differential privacy and the Laplace mechanism. Section 3.2 covers (ϵ, δ) -differential privacy and the Gaussian mechanism. Section 3.3 describes naive approaches to answering a batch of linear counting queries. Section 3.4 explains important properties of low-rank matrices that are used in our solutions. Table I summarizes frequently used notations throughout the paper.

3.1. ϵ -Differential Privacy and the Laplace Mechanism

The basic idea behind the privacy guarantee of differential privacy is the indistinguishability between *neighbor databases*. Two databases D and D' are called neighbor databases, iff. D' can be obtained by adding or removing exactly one record from D . In the example of Figure 1, a neighbor database can be obtained by removing an individual from the original data, or by adding another one. For linear counting queries, the essential difference between two neighbor databases D and D' is that they differ on exactly one unit count, by exactly one. Formally, let $\{x_1, x_2, \dots, x_n\}$ be the set of unit counts corresponding to D and $\{x'_1, x'_2, \dots, x'_n\}$ be the unit counts for D' . Then, there exists an i , $1 \leq i \leq n$, such that $x_j = x'_j$ for all $j \neq i$, and $|x_i - x'_i| = 1$.

Given a set of queries Q , a randomized mechanism M for answering Q satisfies ϵ -differential privacy, iff. for every possible pair of neighbor databases D and D' , the following inequality holds:

$$\forall R : \Pr(M(Q, D) = R) \leq e^\epsilon \Pr(M(Q, D') = R) \quad (1)$$

where R is any possible output of M , and $M(Q, D)$ (resp. $M(Q, D')$) is the output of M given query set Q and input database D (resp., D'). This inequality indicates that given an output R of M , the adversary can only have limited confidence for inferring whether the input database is D or D' , regardless of his/her background knowledge. Since D and D' can be any two neighbor databases that differ in any record, the above inequality also limits the adversary's confidence for inferring the presence or absence of a record in the input database; hence, it provides plausible deniability to any individual involved in the sensitive data.

The Laplace mechanism [Dwork et al. 2006c] is a fundamental solution for enforcing ϵ -differential privacy, based on the concept of \mathcal{L}_1 sensitivity. Given a query set Q , its \mathcal{L}_1 sensitivity

$\Delta(Q)$ is the maximum \mathcal{L}_1 distance between the exact results of Q on any pair of neighbor databases D and D' , Formally, we have:

$$\Delta(Q) = \max_{D, D'} \|Q(D), Q(D')\|_1 \quad (2)$$

Note that in the above equation, D and D' can be any pair of neighbor databases. Hence, $\Delta(Q)$ is a property of the query set Q and the data domain, and it does not depend upon the actual sensitive data D . In the example of Figure 1, the \mathcal{L}_1 sensitivity of a single query $q_1 = x_{NY} + x_{NJ} + x_{CA} + x_{WA}$ is 1, because any two neighbor databases D and D' differ on only one unit count (which can be one of x_{NY}, x_{NJ}, x_{CA} or x_{WA}) by exactly 1. If we include $q_2 = x_{NY} + x_{NJ}$ and $q_3 = x_{CA} + x_{WA}$ to the query set Q , the \mathcal{L}_1 sensitivity of $Q = \{q_1, q_2, q_3\}$ is 2, because a change of 1 on any of x_{NY}, x_{NJ}, x_{CA} or x_{WA} affects the result of q_1 by 1, and either one (but not both) of q_2 and q_3 by 1, leading to a \mathcal{L}_1 distance of 2.

Given a database D and a query set Q , the Laplace mechanism (denoted as M_{Lap}) outputs a randomized result set R that follow the Laplace distribution with mean $Q(D)$ and scale $\frac{\Delta(Q)}{\epsilon}$, i.e.,

$$\Pr(M_{Lap}(Q, D) = R) \propto \exp\left(-\frac{\epsilon}{\Delta(Q)} \|R - Q(D)\|_1\right) \quad (3)$$

This is equivalent to adding independent Laplace noise to the exact result of each query in Q , i.e., $M(Q, D) = Q(D) + Lap\left(\frac{\Delta(Q)}{\epsilon}\right)^m$, where m is the number of queries in Q , and $Lap\left(\frac{\Delta(Q)}{\epsilon}\right)$ is a random variable following zero-mean Laplace distribution with scale $\lambda = \frac{\Delta(Q)}{\epsilon}$. The probability density function of zero-mean Laplace distribution is:

$$f(x) = \frac{1}{2\lambda} \exp\left(-\frac{\|x\|_1}{\lambda}\right) \quad (4)$$

According to properties of the Laplace distribution, the variance of $Lap(\lambda)$ is $2\lambda^2 = \frac{2\Delta(Q)^2}{\epsilon^2}$. Since the Laplace noise injected to each of the m query results is independent, the overall expected squared error of the query answers obtained by the Laplace mechanism is $\frac{2m\Delta(Q)^2}{\epsilon^2}$. In our running example in Figure 1, to answer the query set $Q = \{q_1 = x_{NY} + x_{NJ} + x_{CA} + x_{WA}, q_2 = x_{NY} + x_{NJ}, q_3 = x_{CA} + x_{WA}\}$ under ϵ -differential privacy, a direct application of the Laplace mechanism injects independent, zero-mean Laplace noise of scale $\frac{2}{\epsilon}$ to the exact result of each of q_1, q_2 and q_3 , since the \mathcal{L}_1 sensitivity for this set of queries is 2, as discussed in Section 1. The overall squared error for Q is thus $\frac{2 \times 3 \times 2^2}{\epsilon^2} = \frac{24}{\epsilon^2}$.

3.2. (ϵ, δ) -Differential Privacy and the Gaussian Mechanism

ϵ -differential privacy can be difficult to enforce, especially for queries with high \mathcal{L}_1 sensitivity, or those whose \mathcal{L}_1 sensitivity is difficult to analyze. Hence, relaxed versions of ϵ -differential privacy have been studied in the past, among which a popular definition is the (ϵ, δ) -differential privacy, also called approximate differential privacy. This definition involves an additional parameter δ , which is a non-negative real number controlling how closely this definition approximates ϵ -differential privacy. Formally, let $Range(M)$ be the set of all possible outputs of a mechanism M . A randomized mechanism M satisfies (ϵ, δ) -differential privacy, iff. for any two neighbor databases D and D' , the following holds:

$$\forall \mathbf{R} \subseteq Range(M) : \Pr(M(Q, D) \in \mathbf{R}) \leq e^\epsilon \Pr(M(Q, D') \in \mathbf{R}) + \delta \quad (5)$$

where \mathbf{R} is any set of possible results of M . It can be derived that when $\delta = 0$, (ϵ, δ) -differential privacy is equivalent to ϵ -differential privacy. Accordingly, since δ is non-negative, any mechanism

that satisfies ϵ -differential privacy also satisfies (ϵ, δ) -differential privacy for any value of δ . When $\delta > 0$, (ϵ, δ) -differential privacy relaxes ϵ -differential privacy by ignoring outputs of M with very small probability (controlled by parameter δ). In other words, an (ϵ, δ) -differentially private mechanism satisfies ϵ -differential privacy with a probability controlled by δ .

A basic mechanism for enforcing (ϵ, δ) -differential privacy is the *Gaussian mechanism* [Dwork et al. 2006b], which involves the concept of \mathcal{L}_2 sensitivity. For any two neighbor databases D and D' , the \mathcal{L}_2 sensitivity $\Theta(Q)$ of a query set Q is defined as:

$$\Theta(Q) = \max_{D, D'} \|Q(D) - Q(D')\|_2 \quad (6)$$

In the running example shown in Figure 1, the \mathcal{L}_2 sensitivity for the query set $Q = \{q_1 = x_{NY} + x_{NJ} + x_{CA} + x_{WA}, q_2 = x_{NY} + x_{NJ}, q_3 = x_{CA} + x_{WA}\}$ is $\sqrt{2}$, since the exact results of q_1 (as well as one of q_2 and q_3) differ by at most 1 for any two neighbor databases, leading to an \mathcal{L}_2 sensitivity of $\sqrt{1^2 + 1^2} = \sqrt{2}$. Similar to \mathcal{L}_1 sensitivity, the \mathcal{L}_2 sensitivity $\Theta(Q)$ depends on the data domain \mathbb{D} and the query set Q , not the actual data. Given a database D and a query set Q , the Gaussian mechanism (denoted by M_{Gau}) outputs a random result that follows the Gaussian distribution with mean $Q(D)$ and magnitude $\sigma = \frac{\Theta(Q)}{h(\epsilon, \delta)}$, where $h(\epsilon, \delta) = \frac{\epsilon}{\sqrt{8 \ln(2/\delta)}}$. This is equivalent to adding m -dimensional independent Gaussian noise $Gau\left(\frac{\Theta(Q)}{h(\epsilon, \delta)}\right)^m$, in which $Gau\left(\frac{\Theta(Q)}{h(\epsilon, \delta)}\right)$ is a random variable following a zero-mean Gaussian distribution with scale $\sigma = \frac{\Theta(Q)}{h(\epsilon, \delta)}$. The probability density function of zero-mean Gaussian distribution is:

$$g(x) = \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left(-\frac{\|x\|_2^2}{2\sigma^2}\right) \quad (7)$$

According to properties of the Gaussian distribution, the variance of $Gau(\sigma)$ is $\sigma^2 = \frac{\Theta(Q)^2}{h(\epsilon, \delta)^2}$. Since independent Gaussian noise is injected to each of the m query results, the total expected squared error for the query set is $\frac{m\Theta(Q)^2}{h(\epsilon, \delta)^2}$. In our running example in Figure 1, to answer the query set $Q = \{q_1 = x_{NY} + x_{NJ} + x_{CA} + x_{WA}, q_2 = x_{NY} + x_{NJ}, q_3 = x_{CA} + x_{WA}\}$ under (ϵ, δ) -differential privacy, a direct application of the Gaussian mechanism injects independent, zero-mean Laplace noise of scale $\frac{\sqrt{2}}{h(\epsilon, \delta)}$ to the exact result of each of q_1 , q_2 and q_3 , since the \mathcal{L}_2 sensitivity for this set of queries is $\sqrt{2}$, according to Equation (6). The overall squared error for Q is thus $\frac{3 \times (\sqrt{2})^2}{(h(\epsilon, \delta))^2} = \frac{48 \ln(2/\delta)}{\epsilon^2}$.

3.3. Naive Solutions for Answering a Batch of Linear Counting Queries

This paper focuses on answering a batch of linear counting queries, each of which is a linear combination of the unit counts of the input database D . Formally, given a weight vector $(w_1, w_2, \dots, w_n)^T \in \mathbb{R}^n$, a linear counting query can be expressed as:

$$q(D) = w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

We aim to answer a batch of m linear queries, $Q = \{q_1, q_2, \dots, q_m\}$. The query set Q thus can be represented by a *workload matrix* W with m rows and n columns. Each entry W_{ij} in W is the weight in query q_i on the j -th unit count x_j . Since we do not use any other information of the input database D besides the unit counts, in the following we abuse the notation by using D to represent the vector of unit counts, i.e., $D = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$. Hence, the query batch Q can be

answered by:

$$Q(D) = WD = \left(\sum_j W_{1j}x_j, \dots, \sum_j W_{mj}x_j \right)^T \in \mathbb{R}^{m \times 1}$$

Two naive solutions for enforcing differential privacy on a query batch are as follows.

Noise on data (NOD). The main idea of NOD is to add noise to each unit count. Then, the set of noisy unit counts are published, which can be used to answer any linear counting query. Because two neighbor databases differ on exactly one unit count, by exactly 1, both the \mathcal{L}_1 and the \mathcal{L}_2 sensitivity for the set of unit counts is 1, according to their respective definitions. NOD employs the Laplace mechanism to enforce ϵ -differential privacy (or the Gaussian mechanism to enforce (ϵ, δ) -differential privacy) on the published unit counts, and then combines the noisy unit counts to answer the query batch Q . Let $M_{NOD,\epsilon}$ and $M_{NOD,(\epsilon,\delta)}$ denote the NOD mechanism for enforcing ϵ -differential privacy and (ϵ, δ) -differential privacy, respectively. We have:

$$M_{NOD,\epsilon}(Q, D) = W \left(D + Lap \left(\frac{1}{\epsilon} \right)^n \right)$$

$$M_{NOD,(\epsilon,\delta)}(Q, D) = W \left(D + Gau \left(\frac{1}{h(\epsilon, \delta)} \right)^n \right)$$

where $h(\epsilon, \delta) = \frac{\epsilon}{\sqrt{8 \ln(2/\delta)}}$ as in the Gaussian mechanism.

Based on the analysis of the Laplace and Gaussian mechanisms, the expected squared error for $M_{NOD,\epsilon}$ and $M_{NOD,(\epsilon,\delta)}$ is $\frac{2}{\epsilon^2} \sum_{i,j} W_{ij}^2$ and $\frac{1}{(h(\epsilon,\delta))^2} \sum_{i,j} W_{ij}^2$, respectively. For both privacy definitions, the error of NOD is proportional to the squared sum of the entries in W .

Noise on results (NOR). NOR simply applies the Laplace mechanism (for ϵ -differential privacy) or the Gaussian mechanism (for (ϵ, δ) -differential privacy) directly on the query set Q . Recall that each query $q_i \in Q$ is a linear combination of the unit counts, i.e., $q_i = \sum_j W_{ij}x_j$. Meanwhile, two neighbor databases differ on exactly one unit count, by exactly 1. Therefore, the sensitivity (both \mathcal{L}_1 and \mathcal{L}_2) of q_i is $\max_j W_{ij}$, i.e., the maximum unit count weight in q_i . Regarding Q , its \mathcal{L}_1 sensitivity is $\Delta(Q) = \max_j \sum_i |W_{ij}|$, i.e., the highest column absolute sum [Li et al. 2010]. Similarly, its \mathcal{L}_2 sensitivity is $\Theta(Q) = \max_j \sqrt{\sum_i W_{ij}^2}$, i.e., the highest column \mathcal{L}_2 norm value [Li et al. 2010]. Thus, $M_{NOR,\epsilon}$ and $M_{NOR,(\epsilon,\delta)}$ output the following results.

$$M_{NOR,\epsilon}(Q, D) = WD + Lap \left(\frac{\Delta(Q)}{\epsilon} \right)^m$$

$$M_{NOR,(\epsilon,\delta)}(Q, D) = WD + Gau \left(\frac{\Theta(Q)}{h(\epsilon, \delta)} \right)^m$$

where $\Delta(Q) = \max_j \sum_i |W_{ij}|$, $\Theta(Q) = \max_j \sqrt{\sum_i W_{ij}^2}$, and $h(\epsilon, \delta) = \frac{\epsilon}{\sqrt{8 \ln(2/\delta)}}$.

Similar to the analysis of the Laplace and the Gaussian mechanisms, the expected squared error of the $M_{NOR,\epsilon}$ on query Q is $\frac{2m\Delta(Q)^2}{\epsilon^2} = \frac{2m \max_j \sum_i W_{ij}^2}{\epsilon^2}$, and the expected squared error of $M_{NOR,(\epsilon,\delta)}$ is $\frac{m\Theta(Q)^2}{h(\epsilon,\delta)^2} = \frac{m \max_j \sum_i W_{ij}^2}{h(\epsilon,\delta)^2}$. An interesting observation is that under (ϵ, δ) -differential privacy, NOR obtains lower expected squared error than NOD, iff. $m \max_j \sum_i W_{ij}^2 < \sum_j \sum_i W_{ij}^2$. Note that when $m \geq n$, this inequality can never hold, implying that NOR is more effective for when the number of queries m is smaller than the number of unit counts n .

3.4. Low-Rank Matrices and Matrix Norms

The rank of a real-value matrix W is the number of non-zero *singular values* obtained by performing *singular value decomposition* (SVD) of W . SVD decomposes W of size $m \times n$ into the product of three matrices: $W = U\Sigma V$. U and V are row-wise and column-wise orthogonal matrices respectively, and Σ is a diagonal matrix with positive real diagonal values, which are the singular values of W . Let s be the number of such singular values, i.e., the rank of W . Then, Matrices U , Σ , and V are of sizes $m \times s$, $s \times s$, and $s \times n$ respectively. SVD guarantees that $s \leq \min\{m, n\}$.

A matrix W of size $m \times n$ whose rank is less than $\min\{m, n\}$ is called a *low-rank matrix*. This happens when the rows and columns of W are correlated. In the running example of Figure 1, the workload matrix corresponding to the query set $Q = \{q_1 = x_{NY} + x_{NJ} + x_{CA} + x_{WA}, q_2 = x_{NY} + x_{NJ}, q_3 = x_{CA} + x_{WA}\}$ is a low-rank matrix, since the queries in Q are correlated (i.e., $q_1 = q_2 + q_3$, and the unit counts are also correlated (e.g., x_{NY} and x_{NJ}). The main idea of the proposed low-rank mechanism is to exploit the low-rank property of the workload matrix to reduce the necessary amount of noise required to satisfy differential privacy.

An important concept used in the proposed solution is the matrix norm, which is an extension of the notion of vector norms to matrices. Two common definitions of the matrix norm are: (i) Entrywise norm, which treats a matrix W of size $m \times n$ simply as a vector of size $m \times n$ consisting of all entries of W , and applies one of the vector norm definitions. For example, applying the \mathcal{L}_2 -norm to all entries in W obtains $\|W\|_2 = (\sum_{i=1}^m \sum_{j=1}^n |W_{ij}|^2)^{1/2}$, which is also called the *Frobenius norm*, written as $\|W\|_F$. (ii) Induced norm (or Operator norm), defined by $\|W\|_p = \max_{x \neq 0} \|Wx\|_p / \|x\|_p$, where x is a vector of size n , and $\|x\|_p$ is the \mathcal{L}_p norm of x . Notably, $\|W\|_1$ is simply the maximum absolute column sum of W , and $\|W\|_\infty$ is simply the maximum absolute row sum of the matrix W .

4. WORKLOAD DECOMPOSITION

Recall that the example in Figure 1 shows that sometimes it is best to answer a batch of linear counting queries Q *indirectly*, by first answering a set of intermediate linear counting queries under differential privacy, and combine their results to answer Q . The proposed low-rank mechanism (LRM) follows this idea. Specifically, given a workload matrix W corresponding to the query set Q , LRM decomposes W into the product of two matrices $W = BL$. B is of size $m \times r$ and L is of size $r \times n$. Here, r is a parameter to be determined which specifies the number of intermediate queries; L corresponds to the set of intermediate linear counting queries to answer under differential privacy; B indicates how the results of these intermediate queries are combined to answer Q . The main challenge lies in how to choose the best decomposition that minimizes the overall error of Q , as there is a vast search space for possible decompositions. In this section, we model the search for the optimal matrix decomposition as a constrained optimization program, which is solved in the next section. For the ease of presentation, we focus on ϵ -differential privacy in this and the next section, and defer the discussion of (ϵ, δ) -differential privacy until Section 6. In addition, we provide asymptotic error bounds for LRM in Appendix B.

In the following, Section 4.1 formalize LRM and the optimization program of workload decomposition. Section 4.2 analyzes the result utility of LRM with the optimal workload decomposition, and discusses the selection of the privacy parameter ϵ . Finally, Section 4.3 presents a relaxed optimization program for workload decomposition which can further improve the accuracy of LRM for certain workloads.

4.1. Optimization Program Formulation

We first formalize LRM under ϵ -differential privacy. Given W and its decomposition $W = BL$, LRM first applies the Laplace mechanism to the intermediate queries specified by L . Let $\Delta(L)$ denote the \mathcal{L}_1 sensitivity of these intermediate queries. Similar to the case of NOR discussed in Section 3.3, $\Delta(L)$ is the maximum sum of absolute values of a column in L , which is:

$$\Delta(L) = \max_j \sum_i |L_{ij}|$$

Applying the Laplace mechanism, we obtain the noisy results of the intermediate queries:

$$LD + \text{Lap}\left(\frac{\Delta(L)}{\epsilon}\right)^r$$

where D denotes the vector of unit counts. Next, LRM multiplies matrix B with the noisy intermediate results, which essentially recombines the intermediate results to answer Q . Let $M_{LRM,\epsilon}(Q, D)$ denote LRM under ϵ -differential privacy, we have:

$$M_{LRM,\epsilon}(Q, D) = B \left(LD + \text{Lap}\left(\frac{\Delta(L)}{\epsilon}\right)^r \right) \quad (8)$$

Since $W = BL$, we have $Q(D) = WD = BLD$. Hence, the output $M_{LRM,\epsilon}(Q, D)$ can be seen as the sum of two components: BLD and $B \cdot \text{Lap}\left(\frac{\Delta(L)}{\epsilon}\right)^r$. The former is the exact result of Q , and the latter is the noise added in order to satisfy differential privacy. Next we analyze the error of LRM. First we define the *scale* of a decomposition, as follows.

Definition 4.1. Scale of a workload decomposition. Given a workload decomposition $W = BL$, its scale $\Phi(B)$ is the squared sum of the entries in B , i.e., $\Phi(B) = \sum_{i,j} B_{ij}^2$.

Meanwhile, we call $\Delta(L)$ the \mathcal{L}_1 sensitivity of the decomposition $W = BL$. The following lemma shows that the expected squared error of LRM is linear to the scale of the decomposition, and quadratic to the \mathcal{L}_1 sensitivity of the decomposition.

LEMMA 4.2. *The expected squared error of $M_{LRM,\epsilon}(Q, D)$ using decomposition $W = BL$ is $\frac{2\Phi(B)\Delta(L)^2}{\epsilon^2}$.*

PROOF. According to Equation (8), $M_{LRM,\epsilon}(Q, D) - Q(D) = B \cdot \text{Lap}\left(\frac{\Delta(L)}{\epsilon}\right)^r$. The expected squared error of the mechanism is thus $\left(\sum_{i,j} B_{ij}^2\right) \frac{2(\Delta(L))^2}{\epsilon^2}$. Since $\Phi(B) = \sum_{i,j} B_{ij}^2$, the error can be rewritten as $\frac{2\Phi(B)\Delta(L)^2}{\epsilon^2}$. \square

Therefore, to find the best workload decomposition, it suffice to solve the optimal B and L that minimize $\Phi(B) (\Delta(L))^2$, while satisfying $W = BL$. However, this optimization program is difficult to solve, because (i) the objective function involves the product of $\Phi(B)$ and the square of $\Delta(L)$, and (ii) $\Delta(L)$ may not be differentiable. To address this problem, we first prove an important property of workload decomposition, which implies that the exact value of $\Delta(L)$ is not important.

LEMMA 4.3. *Given a workload decomposition $W = BL$, we can always construct another decomposition $W = B'L'$ satisfying (i) $\Delta(L') = 1$ and (ii) (B', L') lead to the same expected squared error of $M_{LRM,\epsilon}$ as (B, L) , i.e.,*

$$\Phi(B)\Delta(L)^2 = \Phi(B') (\Delta(L'))^2 = \Phi(B')$$

PROOF. We obtain B' and L' by $B' = \Delta(L)B$, $L' = \frac{1}{\Delta(L)}L$. Based on the definition of \mathcal{L}_1 sensitivity, we have

$$\Delta(L') = \max_j \sum_i |L'_{ij}| = \max_j \sum_i \left| \frac{L_{ij}}{\Delta(L)} \right| = \frac{1}{\Delta(L)} \Delta(L) = 1$$

Meanwhile, according to Definition 4.1, we have:

$$\Phi(B') = \sum_{ij} (B'_{ij})^2 = \sum_{ij} \Delta(L)^2 (B_{ij})^2 = \Phi(B) \Delta(L)^2$$

This leads to the conclusion of the lemma. \square

It follows from the above lemma is that there must be an optimal decomposition with \mathcal{L}_1 sensitivity equal to 1, because we can always apply Lemma 4.3 to transform an optimal decomposition whose \mathcal{L}_1 sensitivity is not 1 to another optimal decomposition whose \mathcal{L}_1 sensitivity is 1. Therefore, it suffices to fix $\Delta(L)$ to 1 in the optimization program. Meanwhile, according to properties of the matrix trace, we have $\Phi(B) = \text{tr}(B^T B)$. Thus, we arrive at the following theorem.

THEOREM 4.4. *Given the workload W , a workload decomposition $W = BL$ minimizes the expected squared error of the queries, if (B, L) is the optimal solution to the following program:*

$$\begin{aligned} \min_{B, L} \quad & \frac{1}{2} \text{tr}(B^T B) \\ \text{s.t.} \quad & W = BL \\ & \forall j \sum_i^r |L_{ij}| \leq 1 \end{aligned} \tag{9}$$

The constant factor $1/2$ in the objective function above simplifies the notations in the following sections; it does not affect the optimal solution of the program. We omit the proof since it is already clear from the discussions above. Solving the above optimization program is rather difficult, since it involves a non-linear objective function and complex constraints. We present a relaxation of the problem in Section 4.3, and our solution in Section 5.

4.2. Utility Analysis and Budget Selection

In practice, users are often unsure about how to set the privacy parameter ϵ involved in ϵ -differential privacy. Instead, setting the desired *utility* level of the query results is much more intuitive. Given the user-specified utility, this subsection derives the smallest ϵ value for LRM that satisfies the utility requirement. Note that smaller values of ϵ corresponds to stronger privacy protection. We use a common definition of query result utility called (ξ, η) -usefulness [Blum et al. 2008], as follows.

Definition 4.5. Given a mechanism M , query set Q , sensitive data D , and parameters $\xi > 0$ and $0 < \eta < 1$, we say that M is (ξ, η) -useful with respect to Q and D under the $\|\cdot\|_*$ -norm if the following inequality holds:

$$\Pr(\|M(Q, D) - Q(D)\|_* \geq \xi) \leq \eta$$

where $\|\cdot\|_*$ -norm can be any vector norm definition. In our analysis, we consider the $\|\cdot\|_1$ -norm and the $\|\cdot\|_\infty$ -norm.

Given user specified values of ξ and η , we now derive the minimum value for ϵ with which LRM achieves (ξ, η) -usefulness. The derivation uses Markov's inequality and the Chernoff bound, as follows.

LEMMA 4.6. *Markov's Inequality and the Chernoff Bound [Billingsley 2012]. Given a non-negative random variable X and $t > 0$, the following inequality holds:*

$$\Pr(X \geq t) \leq \frac{\mathbb{E}[X]}{t}$$

Moreover, for any $s \geq 0$, we have:

$$\Pr(X \geq t) = \Pr(e^{sX} \geq e^{st}) \leq \frac{\mathbb{E}[e^{sX}]}{e^{st}}$$

The minimum ϵ value is given in the following theorem.

THEOREM 4.7. Utility of LRM under ϵ -differential privacy. *Given query set Q , database D , and user-specified parameters $\xi > 0$ and $0 < \eta < 1$, (i) $M_{LRM,\epsilon}$ with the optimal decomposition $W = BL$ solved from Program (9) returns (ξ, η) -useful results of Q on D under the $\|\cdot\|_1$ -norm, when the privacy parameter ϵ satisfies $\epsilon \geq (2\|B\|_1(s \cdot \ln 2 - \ln \eta)) / \xi$. (ii) Meanwhile, $M_{LRM,\epsilon}$ with the optimal decomposition achieves (ξ, η) -usefulness under the $\|\cdot\|_\infty$ -norm, when $\epsilon \geq (2\|B\|_\infty(\sum_{i=1}^s \ln(\frac{i}{i-0.5}) - \ln \eta)) / \xi$.*

PROOF. (i) We first prove the utility of LRM under the $\|\cdot\|_1$ -norm. Let X be the Laplace noise vector injected to the results of intermediate queries corresponding to L . We have:

$$\begin{aligned} \|M_P(Q, D) - Q(D)\|_1 &= \|B(LD + X) - WD\|_1 \\ &= \|B \cdot X\|_1 = \|B \cdot X\|_1 \leq \|B\|_1 \cdot \|X\|_1 = \|B\|_1 \cdot \|X\|_1 \end{aligned}$$

According to the Laplace mechanism, X_1, X_2, \dots, X_r are i.i.d. random variables following the zero-mean Laplace distribution with scale $\Delta(L)/\epsilon$. Since L is obtained by solving Program (9), we have $\Delta(L) = 1$. Therefore, the scale of each of the Laplace variable $X_i, 1 \leq i \leq r$ is $1/\epsilon$. According to properties of the Laplace distribution, $|X_i|$ follows the exponential distribution with rate parameter equal to ϵ . Let $Y = \|X\|_1 = |X_1| + |X_2| + \dots + |X_r|$. Then, according to properties of the exponential distribution, Y follows the Erlang distribution. Specifically, the probability distribution function of Y is:

$$\Pr(Y = x) = \frac{\epsilon^r x^{r-1} e^{-\epsilon x}}{(r-1)!} dx$$

For any positive number t such that $\mathbb{E}[e^{tY}]$ exists, we have:

$$\mathbb{E}[e^{tY}] = \int_0^\infty e^{tx} \cdot \frac{\epsilon^r x^{r-1} e^{-\epsilon x}}{(r-1)!} dx = (1 - \frac{t}{\epsilon})^{-r}, t < \epsilon$$

Moreover, for any real number c , according to Lemma 4.6, we have:

$$\Pr(Y > c) = \Pr(e^{tY} > e^{tc}) \leq \frac{\mathbb{E}[e^{tY}]}{e^{tc}} = \frac{(1 - \frac{t}{\epsilon})^{-r}}{e^{tc}}$$

Setting $t = \frac{\xi}{2}$ and $c = \frac{\xi}{\|B\|_1}$, we obtain:

$$\Pr(Y > \frac{\xi}{\|B\|_1}) \leq \frac{(\frac{1}{2})^{-r}}{e^{\frac{\xi \epsilon}{2\|B\|_1}}}$$

Therefore, we have:

$$\begin{aligned} \|M_P(Q, D) - Q(D)\|_1 &\leq \|B\|_1 \cdot Y \\ \Rightarrow \forall \xi, \Pr(\|M_P(Q, D) - Q(D)\|_1 \geq \xi) &\leq \Pr(Y \geq \frac{\xi}{\|B\|_1}) \leq \frac{(\frac{1}{2})^{-r}}{e^{\frac{\xi \epsilon}{2\|B\|_1}}} \end{aligned} \tag{10}$$

When $\epsilon \geq (2\|B\|_1(r \cdot \ln 2 - \ln \eta)) / \xi$, the above probability is thus bound by η . This finishes the proof for claim (i) in the theorem.

(ii) Next we focus on the $\|\cdot\|_\infty$ -norm. Let X denote the same meaning as in the proof of part (i). Then, we have:

$$\|M_P(Q, D) - Q(D)\|_\infty = \|B \cdot X\|_\infty \leq \|B\|_\infty \cdot \|X\|_\infty$$

The inequality above holds due to the fact that: $\|Rx\|_\infty \leq \|R\|_\infty \cdot \|x\|_\infty$ for any matrix R and vector x . Let $Y = \|X\|_\infty = \max(|X_1|, |X_2|, \dots, |X_r|)$. Similar to part (i) of the proof, each $|X_i|$, $1 \leq i \leq r$ follows the exponential distribution with rate ϵ . According to the memoryless property of the exponential distribution, we create a chain of variables, as follows:

$$Y = \max(|X_1|, |X_2|, \dots, |X_r|) = X_{\lambda=r\epsilon} + X_{\lambda=(r-1)\epsilon} + \dots + X_{\lambda=\epsilon} \quad (11)$$

where each $X_{\lambda=x}$ denotes an independent exponential random variable with rate x . Intuitively, $X_{\lambda=r\epsilon}$ models the distribution of the smallest value among $|X_1|, |X_2|, \dots, |X_r|$; $X_{\lambda=(r-i+1)\epsilon}$, $1 < i \leq r$ models the difference between the i -th smallest value and the $(i-1)$ -th smallest value among $|X_1|, |X_2|, \dots, |X_r|$. The sum thus yields the maximum value among $|X_1|, |X_2|, \dots, |X_r|$.

Similar to the part (i) of the proof, we further derive:

$$\mathbb{E}[e^{tY}] = \mathbb{E}[e^{t(X_{\lambda=r\epsilon} + X_{\lambda=(r-1)\epsilon} + \dots + X_{\lambda=\epsilon})}] = \mathbb{E}[e^{t(X_{\lambda=r\epsilon})}] \cdot \mathbb{E}[e^{t(X_{\lambda=(r-1)\epsilon})}] \cdot \dots \cdot \mathbb{E}[e^{t(X_{\lambda=\epsilon})}]$$

Because $\mathbb{E}[e^{tX_{\lambda=a}}] = \int_0^\infty e^{tx} \cdot ae^{-ax} dx = \frac{a}{a-t}$ for any $t < a$, we reach:

$$\forall t < \epsilon, \mathbb{E}[e^{tY}] = \prod_{i=1}^r \frac{i\epsilon}{i\epsilon - t}$$

Finally, according to Lemma 4.6, we have the following inequality:

$$\begin{aligned} \Pr(Y > c) &= \Pr(e^{tY} > e^{tc}) \\ &\leq \frac{\mathbb{E}[e^{t(Y)}]}{e^{ct}} \\ &= \prod_{i=1}^r \frac{i\epsilon}{i\epsilon - t} / e^{ct} \end{aligned}$$

With the choice of $t = \frac{\xi}{2}$ and $c = \frac{\xi}{\|B\|_\infty}$, we obtain:

$$\begin{aligned} \|M_P(Q, D) - Q(D)\|_\infty &\leq \|B\|_\infty \cdot \|X\|_\infty \\ \Rightarrow \forall \xi, \Pr(\|M_P(Q, D) - Q(D)\|_\infty > \xi) &\leq \Pr(\|X\|_\infty > \frac{\xi}{\|B\|_\infty}) \\ \Rightarrow \forall \xi, \Pr(\|M_P(Q, D) - Q(D)\|_\infty > \xi) &\leq \left(\prod_{i=1}^r \frac{i\epsilon}{i\epsilon - \xi/2} \right) / e^{c\xi} = \left(\prod_{i=1}^r \frac{i\epsilon}{i\epsilon - \xi/2} \right) / e^{2\frac{\xi\epsilon}{\|B\|_\infty}} \end{aligned}$$

When $\epsilon \geq \left(2\|B\|_\infty \left(\sum_{i=1}^r \ln\left(\frac{i}{i-0.5}\right) - \ln \eta \right) \right) / \xi$, the above probability is bounded by η . \square

4.3. Relaxed Workload Decomposition

Program 9 is rather difficult to solve, since it contains a non-linear objective and complex constraints. To devise a stable numerical solution, we relax the formulation so that BL does not necessarily match W exactly, but within a small error tolerance. To do this, we introduce a new parameter γ to bound the difference between W and BL in terms of the Frobenius norm. This leads to the following optimization program:

$$\begin{aligned} \min_{B, L} & \frac{1}{2} \text{tr}(B^T B) \\ \text{s.t.} & \|W - BL\|_F \leq \gamma \\ & \forall j \sum_i^r |L_{ij}| \leq 1 \end{aligned} \quad (12)$$

The following theorem analyzes the error of LRM with the optimal decomposition obtained by solving Program (12).

THEOREM 4.8. *The expected squared error of $M_{LRM,\epsilon}(Q, D)$ using the optimal decomposition (B, L) solved from Program (12) is at most*

$$2\text{tr}(B^T B)/\epsilon^2 + \gamma \sum_i x_i^2$$

PROOF. When $W \neq BL$, there are two sources of error. The first is the added Laplace noise. According to Lemma 4.2, the error incurred by the Laplace noise is at most $\frac{2}{\epsilon^2}\Phi(B)(\Delta(L))^2 \leq \frac{2}{\epsilon^2}\text{tr}(B^T B)$.

The second source of the error is due to the difference between W and BL . The incurred expected squared error is bounded by:

$$\begin{aligned} & ((W - BL)D)^T(W - BL)D \\ & \leq \|W - BL\|_F^2 D^T D = \|W - BL\|_F^2 \sum_{i=1}^n x_i^2 \end{aligned}$$

The inequality above is due to the Cauchy-Schwartz inequality. By linearity of expectation, the expected squared errors can be simply summed up. This leads to the conclusion of the theorem.

□

While Theorem 4.8 implies the possibility of estimating the optimal γ , it is not practical to implement it directly, because this estimation depends on the data, i.e., $\sum_i x_i^2$. In our experiments, we test different values of γ , report their relative performance, and describe guidelines for setting the appropriate γ independently of the underlying data.

5. SOLVING FOR THE OPTIMAL WORKLOAD DECOMPOSITION

This section solves the relaxed workload decomposition problem defined in Program (12). This program is rather difficult to solve, because it is neither convex nor differentiable. In the following, Section 5.1 describes an effective and efficient solution, based on the inexact Augmented Lagrangian method [Conn et al. 1997; Lin et al. 2010]. Section 5.2 proves that the proposed solution always converges, and analyzes its convergence rate.

5.1. Solution Based on Augmented Lagrangian Method

Observe that Program (12) is a constrained optimization problem with a large number of unknowns, a non-linear objective and rather complex constraints. Since there is no known analytic solution to such a problem, we focus on numerical solutions. Furthermore, Program (12) is difficult to tackle even with numerical methods, due to three main challenges. First and foremost, there are a set of *non-differentiable* constraints $\forall j \sum_i |L_{ij}| \leq 1$, which rules out many generic techniques for solving constrained optimization problems, such as the Lagrange multiplier method, which are limited to problems with differentiable constraints. Second, the non-differentiable constraints involve the unknown matrix L , whereas the objective function involves another unknown matrix B , whose relationship to L is rather complex (i.e., in constraint $\|W - BL\|_F \leq \gamma$); consequently, it is non-trivial to apply specialized methods for handling the non-differentiable constraints. Finally, Program (12) is not convex with respect to the unknowns B and L .

The main idea of the proposed solution is to break down Program (12) into simpler, solvable sub-problems. Since the most difficult part of Program (12) is the existence of the non-differentiable constraints $\forall j \sum_i |L_{ij}| \leq 1$, we aim to break down the whole problem into subproblems with only these constraints, and an objective function that only involves the unknown L , not B . Then, we use a specialized technique to solve each of these subproblems. Specifically, we first eliminate the constraint $\|W - BL\|_F \leq \gamma \rightarrow 0$ using the augmented Lagrangian method, which runs in multiple iterations, each of which solves a subproblem with only the constraints $\forall j \sum_i |L_{ij}| \leq 1$. Then, inside each iteration, we remove B from the objective function of the subproblem, by alternatively optimizing for

ALGORITHM 1: Workload Matrix Decomposition

```

1: Initialize  $\pi^{(0)} = \mathbf{0} \in \mathbb{R}^{m \times n}, \beta^{(0)} = 1, k = 1$ 
2: while not converged do
3:   while not converged do
4:      $B^{(k)} \leftarrow$  update  $B$  using Equation (14)
5:      $L^{(k)} \leftarrow$  run Algorithm 3 to update  $L$  according to Program (15)
6:     Compute  $\tau = \|W - B^{(k)}L^{(k)}\|_F$ 
7:     if  $\tau$  is sufficiently small or  $\beta$  is sufficiently large then
8:       return  $B^{(k)}$  and  $L^{(k)}$ 
9:     if  $k$  is a multiple of 10 then
10:       $\beta^{(k+1)} = 2\beta^{(k)}$ 
11:       $\pi^{(k+1)} = \pi^{(k)} + \beta^{(k+1)} (W - B^{(k)}L^{(k)})$ 
12:       $k = k + 1$ 

```

B and L . The result are subproblems with only the constraints $\forall j \sum_i |L_{ij}| \leq 1$ as well as an objective function that has only L as unknowns. Each of these subproblems are then solved by applying a special solver called Nesterov's first order optimal gradient method [Nesterov 2003]. An important optimization is that we apply the *inexact* augmented Lagrangian method [Conn et al. 1997; Lin et al. 2010], which does not solve the subproblem exactly in each iteration exactly, leading both increased efficiency and stability.

Algorithm 1 shows the proposed solution for Program (12). First, we apply the inexact augmented Lagrangian method to eliminate the linear constraint $\|W - BL\|_F \leq \gamma \rightarrow 0$, as follows: we add to the objective function (i) a positive penalty item $\beta \in \mathbb{R}$ and (ii) the Lagrange multiplier $\pi \in \mathbb{R}^{m \times n}$. β and π are iteratively updated, following the strategy in [Conn et al. 1997; Lin et al. 2010]. In each iteration, the values of β and π are fixed, and the algorithm aims to find values for B and L that minimize the following subproblem:

$$\min_{B,L} \mathcal{J}(B, L, \beta, \pi) = \frac{1}{2} \text{tr}(B^T B) + \langle \pi, W - BL \rangle + \frac{\beta}{2} \|W - BL\|_F^2 \quad (13)$$

$$\text{s.t. } \forall j \sum_i |L_{ij}| \leq 1$$

Next we eliminate unknowns B from the objective function of the above subproblem, Program . Observe that this is a bi-convex optimization problem with respect to B and L , meaning that it is convex with respect to B (resp. L), once we fix L (resp. B) to a constant. Hence, we solve it by alternately optimizing B and L (lines 3-5 of Algorithm 1). Note that following the inexact Augmented Lagrangian Multiplier methodology, it is not necessary to obtain the exact optimal values of B and L , instead, a small number of iterations of the while-loop in lines 4-5 suffices. We first focus on optimizing B , treating L as constant. Observe that $\mathcal{J}(\cdot)$ is convex with respect to B . Hence, the optimal B can be obtained by solving $\frac{\partial \mathcal{J}}{\partial B} = 0$. In particular, the gradient with respect to B is:

$$\frac{\partial \mathcal{J}}{\partial B} = B - \pi L^T + \beta B L L^T - \beta W L^T$$

Solving B from $\frac{\partial \mathcal{J}}{\partial B} = 0$, we obtain:

$$B = (\beta W L^T + \pi L^T) (\beta L L^T + I)^{-1} \quad (14)$$

Next we show how to optimize L with a fixed B . This is equivalent to the following quadratic program:

$$\begin{aligned} \mathcal{G}(L) &= \frac{\beta}{2} \text{tr}(L^T B^T B L) - \text{tr}\left((\beta W + \pi)^T B L\right) \\ &\text{s.t. } \forall j \sum_i |L_{ij}| \leq 1 \end{aligned} \quad (15)$$

The gradient of the objective $\mathcal{G}(L)$ respect to L in (15) can be computed as:

$$\frac{\partial \mathcal{G}}{\partial L} = \beta B^T B L - \beta B^T W - B^T \pi \quad (16)$$

For all L', L'' with $\forall j \sum_i |L'_{ij}| \leq 1, \forall j \sum_i |L''_{ij}| \leq 1$, we have the following inequalities:

$$\begin{aligned} \frac{\|\mathcal{G}(L') - \mathcal{G}(L'')\|_F}{\|L' - L''\|_F} &= \frac{\|\beta B^T B L' - \beta B^T B L''\|_F}{\|L' - L''\|_F} \\ &\leq \frac{\|\beta B^T B\|_2 \cdot \|L' - L''\|_F}{\|L' - L''\|_F} = \beta \cdot \|B^T B\|_2 \end{aligned}$$

Therefore, the gradient of $\mathcal{G}(L)$ is Lipschitz continuous with parameter $\omega = \beta \cdot \|B^T B\|_2$.

We employ Nesterov's first order optimal gradient method [Nesterov 2003] to solve the program in (15). Nesterov's method has a much faster convergence rate than traditional methods such as the subgradient method or naive projected gradient descent. The updating rule in the projected gradient method is expressed as follows:

$$L^{(t+1)} = \mathcal{P}(L^{(t)} - \eta^{(t)} \frac{\partial \mathcal{G}}{\partial L^{(t)}})$$

where t denotes the iteration counter, $\mathcal{P}(L)$ denotes the \mathcal{L}_1 projection operator on any $L \in \mathbb{R}^{r \times n}$, $\eta > 0$ denotes the appropriate step size. One typical choice for η is the inverse of the gradient lipschitz constant $1/\omega$, however, this can be sub-optimal when the gradient lipschitz constant is large. One can cooperate Beck et al.'s backtracking line search strategy to further accelerate the convergence of the projected gradient algorithm [Beck and Teboulle 2009]. We adopt this line search strategy in our algorithm.

L is updated by gradient descent while ensuring that the \mathcal{L}_1 regularized constraint on L is satisfied. This is done by the \mathcal{L}_1 projection operator, formulated as the following optimization problem:

$$\mathcal{P}(L) = \arg \min_{L \in \mathbb{R}^{r \times n}} \|\bar{L} - L\|_F^2, \text{ s.t. } \forall j \sum_i |\bar{L}_{ij}| \leq 1, \quad (17)$$

We observe that Equation (17) can be decoupled into n independent \mathcal{L}_1 regularized sub-problems:

$$\arg \min_{l \in \mathbb{R}^{r \times 1}} \|\bar{l} - l\|_2^2, \text{ s.t. } \sum_i |\bar{l}_i| \leq 1$$

where $l = L_j^{(t)}, j = 1, 2, \dots, n, L_j^{(t)}$ is the j^{th} column of $L^{(t)}$. Such a projection operator can be solved efficiently by \mathcal{L}_1 projection methods in $\mathcal{O}(r \log r)$ time [Duchi et al. 2008], as described in Algorithm 2. The complete algorithm for solving Program (15) is summarized in Algorithm 3.

5.2. Convergence Analysis

This subsection analyzes the convergence properties of the proposed workload decomposition algorithm. In each iteration, Algorithm 1 solves a sequence of Lagrangian subproblems by optimizing B (step 4) and L (step 5) alternately. The algorithm stops when a sufficiently small γ is obtained or the penalty parameter β is sufficiently large. It suffices to guarantee that L converges to a locally optimal solution [Lin et al. 2010; Wen et al. 2012a; Wen et al. 2012b].

ALGORITHM 2: Algorithm for \mathcal{L}_1 Ball Projection

-
- 1: input: A vector $l \in \mathbb{R}^{r \times 1}$
 - 2: sort l into v such that $v_1 \geq v_2 \geq \dots \geq v_r$
 - 3: find $\rho = \max\{i \in [r] : v_i - \frac{1}{i} \left(\sum_{k=1}^i v_k - 1 \right) > 0\}$
 - 4: compute $\theta = \frac{1}{\rho} \left(\sum_{i=1}^{\rho} v_i - 1 \right)$
 - 5: output $\bar{l} \in \mathbb{R}^{r \times 1}$, s.t. $\bar{l}_i = \max(l_i - \theta, 0)$, $i \in [r]$
-

ALGORITHM 3: Nesterov's Projected Gradient Method

-
- 1: input: $\mathcal{G}(L)$, $\frac{\partial \mathcal{G}}{\partial L}$, $L^{(0)}$
 - 2: $\chi = r \cdot n \cdot 10^{-12}$, Lipschitz parameter: $\omega^{(0)} = 1$
 - 3: Initializations: $L^{(1)} = L^{(0)}$, $\tau^{(-1)} = 0$, $\tau^{(0)} = 1$, $t = 1$
 - 4: **while** not converged **do**
 - 5: $\alpha = \frac{\tau^{(t-2)} - 1}{\tau^{(t-1)}}$, $S = L^{(t)} + \alpha(L^{(t)} - L^{(t-1)})$
 - 6: **for** $j = 0$ to \dots **do**
 - 7: $\omega = 2^j \omega^{(t-1)}$, $U = S - \frac{1}{\omega} \nabla_S$
 - 8: Project U to the feasible set to obtain $L^{(t)}$ (i.e., solve Equation (17))
 - 9: **if** $\|S - L^{(t)}\|_F < \chi$ **then**
 - 10: return;
 - 11: Define function: $\mathcal{J}_{\omega, S}(U) = \mathcal{G}(S) + \langle \frac{\partial \mathcal{G}}{\partial U}, U - S \rangle + \frac{\omega}{2} \|U - S\|_F^2$
 - 12: **if** $\mathcal{G}(L^{(t)}) \leq \mathcal{J}_{\omega, S}(U)$ **then**
 - 13: $\omega^{(t)} = \omega$; $L^{(t+1)} = L^{(t)}$; **break**;
 - 14: Set $\tau^{(t)} = \frac{1 + \sqrt{1 + 4(\tau^{(t-1)})^2}}{2}$
 - 15: $t = t + 1$
 - 16: **return** $L^{(t)}$
-

In general, penalty methods have the property that when the global (or local) minimizers of the subproblem are found, every limit point is a global (or local) minimizer of the original problem [Fiacco and McCormick 1968]. This property is preserved by the Augmented Lagrangian Multiplier counterparts. Therefore, the proposed solution for the workload decomposition problem converges, whenever the bi-convex optimization subproblem in Program (5.1) converges. Regarding the convergence properties of the bi-convex optimization subproblem, past study [Bertsekas 1999] on bi-convex optimization has shown that block coordinate descent is guaranteed to converge to the stationary point for *strictly convex* problems. However, the subproblem in Program (5.1) is not strictly convex (though it is convex); meanwhile, the subproblem may have multiple optimal solutions, which may cause problems to its convergence. Fortunately, for bi-convex optimizations which only involves two blocks, [Grippio and Sciandrone 2000] shows that the strict convexity of the subproblem is not required; every limit point of $\{B^{(k)}, L^{(k)}\}$ is a stationary point. Accordingly, the bi-convex optimization subproblem exhibits nice convergence properties. In the following, we formalize and prove the convergence results of the proposed algorithm.

We first present the first order KKT conditions of the optimization problem in Program (12). Introducing Lagrange multipliers $\mu \in \mathbb{R}^{n \times 1}$ and $\pi \in \mathbb{R}^{m \times n}$ for the inequality constraints $\forall j \sum_i |L_{ij}| \leq 1$ and linear constraints $W = BL$ respectively, we derive the following KKT conditions of the optimization problem:

$$\begin{aligned}
& \mu \geq 0 \text{ (Non-Negativity)} \\
& W = BL, \forall j \sum_i^r |L_{ij}| \leq 1 \text{ (Feasibility)} \\
& B = \pi L^T, 0 \in \sum_j^n \mu_j \frac{\partial \sum_i^r |L_{ij}|}{\partial L} - B^T \pi \text{ (Optimality)} \\
& \forall j \mu_j (\sum_i^r |L_{ij}| - 1) = 0 \text{ (Complementary Slackness)}
\end{aligned} \tag{18}$$

The following theorem establishes the convergence properties of the proposed algorithm, under the assumption that the iterates generated by Algorithm 1 exhibit no jumping behavior. Remark that the similar condition was used in [Wen et al. 2012a; Wen et al. 2012b].

THEOREM 5.1. Convergence of Algorithm 1. *Let $X \triangleq (B, L, \pi)$ and $\{X^{(k)}\}_{k=1}^\infty$ be the intermediate results of Algorithm 1 after the k -th iteration. Assume that $\{X^{(k)}\}_{k=1}^\infty$ is bounded and $\lim_{k \rightarrow \infty} (X^{(k+1)} - X^{(k)}) = 0$. Then any accumulation point of $\{X^{(k)}\}_{k=1}^\infty$ satisfies the KKT conditions presented in Equation (18). In other words, whenever $\{X^{(k)}\}_{k=1}^\infty$ converges, it converges to a first-order KKT optimal point.*

PROOF. Since $L^{(k+1)}$ is the global optimal solution of Program (15), by the KKT optimal condition, there exist $\mu \geq 0, \mu \in \mathbb{R}^{n \times 1}$ and $L^{(k+1)}$ such that the following equation holds:

$$0 \in \frac{\partial \mathcal{G}}{\partial L^{(k+1)}} + \sum_j^n \mu_j \frac{\partial \sum_i^r |L_{ij}^{(k+1)}|}{\partial L^{(k+1)}} \tag{19}$$

Note that \mathcal{G} is a convex function with respect to $L^{(k+1)}$. Hence, the KKT conditions are both necessary and sufficient conditions for global optimality. Combining Equations (16) and (19), we obtain:

$$\begin{aligned}
& \beta B^{(k+1)T} \left(B^{(k+1)T} (L^{(k+1)} - L^{(k)}) \right) \\
& = \beta B^{(k+1)T} (W - B^{(k+1)T} L^{(k)}) + B^{(k+1)T} \pi - \sum_j^n \mu_j \frac{\partial \sum_i^r |L_{ij}^{(k+1)}|}{\partial L^{(k+1)}}
\end{aligned} \tag{20}$$

We derive the following equations according to the update rule for B (at Line 4 in Algorithm 1) and the Lagrangian multiplier update rule for π (at Line 11 in Algorithm 1), respectively:

$$B^{(k+1)} - B^{(k)} = \left(\beta W L^{(k)T} + \pi L^{(k)T} - B^{(k)} (\beta L^{(k)} L^{(k)T} + I) \right) \left(\beta L^{(k)} L^{(k)T} + I \right)^{-1} \tag{21}$$

$$\pi^{(k+1)} - \pi^{(k)} = -\beta^{(k+1)} \left(W - B^{(k+1)} L^{(k+1)} \right) \tag{22}$$

Since $\{X^{(k)}\}_{k=1}^\infty$ is bounded according to our assumption, the sequences $\{B^{(k)}\}_{k=1}^\infty$ and $\{L^{(k)}\}_{k=1}^\infty$ are also bounded. Hence, $\lim_{k \rightarrow \infty} (X^{(k+1)} - X^{(k)}) = 0$ implies that both sides of Equation (20, 21, 22) converge to zero as k approaches infinity. Consequently,

$$\begin{aligned}
& W - B^{(k)} L^{(k)} \rightarrow 0, \quad \pi L^{(k)T} - B^{(k)} \rightarrow 0 \\
& \exists \mu : -B^{(k+1)T} \pi + \sum_j^n \mu_j \frac{\partial \sum_i^r |L_{ij}^{(k+1)}|}{\partial L^{(k+1)}} \rightarrow 0
\end{aligned} \tag{23}$$

where the first limit in Equation (23) is used to derive other limits. Therefore, the sequence $\{X^{(k)}\}_{k=1}^{\infty}$ asymptotically satisfies the KKT conditions in Equation (18). This completes the proof. \square

Next we focus on the convergence rate of the proposed algorithm. The following theorem states that it converges linearly.

THEOREM 5.2. Convergence Rate of Algorithm 1. *Let $X \triangleq (B, L, \pi)$ and $\{X^{(k)}\}_{k=1}^{\infty}$ be the intermediate results of Algorithm 1 after the k -th iteration. Assume that $\{X^{(k)}\}_{k=1}^{\infty}$ is bounded and $\lim_{k \rightarrow \infty} (X^{(k+1)} - X^{(k)}) = 0$. Let $(B^{(k)}, L^{(k)})$ be the solution obtained after the k -th iteration and (B^*, L^*) be the optimal solution to Program (12), we have*

$$\min_{i=1,2,\dots,k} |\text{tr}(B^{(i)T} B^{(i)}) - \text{tr}(B^{*T} B^*)| \leq \mathcal{O}\left(\frac{1}{k}\right) \quad (24)$$

In other words, Algorithm 1 converges to the stationary point linearly.

PROOF. Let $B^{(k)}$ denote the solution of the Lagrangian sub-problem in the k^{th} iteration. The following inequality holds on the sequence of the Lagrangian subproblems:

$$\begin{aligned} & \mathcal{J}(B^{(k+1)}, L^{(k+1)}, \pi^{(k)}, \beta^{(k)}) \\ & \leq \min_{\substack{W=B^T L, \\ \forall j \sum_i |L_{ij}| \leq 1}} \mathcal{J}(B, L, \pi^{(k)}, \beta^{(k)}) \\ & \leq \min_{\substack{W=B^T L, \\ \forall j \sum_i |L_{ij}| \leq 1}} \mathcal{J}(B, L, \pi^*, \beta^{(k)}) \\ & = \min_{\substack{W=B^T L, \\ \forall j \sum_i |L_{ij}| \leq 1}} \frac{1}{2} \text{tr}(B^T B) = \frac{1}{2} \text{tr}(B^{*T} B^*) \end{aligned} \quad (25)$$

By the definition of $\mathcal{J}(\cdot)$ and the inequality above, we derive the following inequality:

$$\begin{aligned} & \frac{1}{2} \text{tr}(B^{(k+1)T} B^{(k+1)}) \\ & = \mathcal{J}(B^{(k+1)}, L^{(k+1)}, \pi^{(k)}, \beta^{(k)}) - \langle \pi^{(k)}, W - B^{(k+1)} L^{(k+1)} \rangle + \frac{\beta^{(k)}}{2} \|W - B^{(k+1)} L^{(k+1)}\|_F^2 \\ & = \mathcal{J}(B^{(k+1)}, L^{(k+1)}, \pi^{(k)}, \beta^{(k)}) - \frac{1}{2\beta^{(k)}} \left(\|\pi^{(k)} + \beta^{(k)}(W - B^{(k+1)} L^{(k+1)})\|_F^2 - \|\pi^{(k)}\|_F^2 \right) \\ & = \mathcal{J}(B^{(k+1)}, L^{(k+1)}, \pi^{(k)}, \beta^{(k)}) - \frac{1}{2\beta^{(k)}} \left(\|\pi^{(k+1)}\|_F^2 - \|\pi^{(k)}\|_F^2 \right) \\ & \leq \frac{1}{2} \text{tr}(B^{*T} B^*) - \frac{1}{2\beta^{(k)}} \left(\|\pi^{(k+1)}\|_F^2 - \|\pi^{(k)}\|_F^2 \right) \end{aligned} \quad (26)$$

The third equality holds because of the Lagrangian multiplier update rule:

$$W - B^{(k+1)} L^{(k+1)} = \frac{1}{\beta^{(k)}} \left(\pi^{(k+1)} - \pi^{(k)} \right).$$

By the non-negativity of norms, we have:

$$\begin{aligned} \frac{1}{2}\text{tr}(B^{(k+1)T} B^{(k+1)}) &\geq \frac{1}{2}\text{tr}(B^{(k+1)T} B^{(k+1)}) - \|W - B^{(k+1)} L^{(k+1)}\|_F^2 \\ &\geq \frac{1}{2}\text{tr}(B^{*T} B^*) - \|W - B^{(k+1)} L^{(k+1)}\|_F^2 \\ &= \frac{1}{2}\text{tr}(B^{*T} B^*) - \frac{1}{2\beta^{(k)}} \left(\|\pi^{(k+1)}\|_F^2 - \|\pi^{(k)}\|_F^2 \right) \end{aligned} \quad (27)$$

Combining Equations (26) and (27), we obtain:

$$\beta^{(i+1)} \left(\text{tr} \left(B^{(i+1)T} B^{(i+1)} \right) - \text{tr} \left(B^{*T} B^* \right) \right) = \|\pi^{(i+1)}\|_F^2 - \|\pi^{(i)}\|_F^2, \forall i$$

Summing this equality above over $i = 0, 1, \dots, k-1$, we have:

$$\sum_{i=0}^{k-1} \beta^{(i+1)} \left(\text{tr} \left(B^{(i+1)T} B^{(i+1)} \right) - \text{tr} \left(B^{*T} B^* \right) \right) = \|\pi^{(k)}\|_F^2 - \|\pi^{(0)}\|_F^2 \quad (28)$$

Since $\beta^{(k)}$ is non-decreasing, we have:

$$\min_{i=0,1,\dots,k-1} \left| \text{tr} \left(B^{(i+1)T} B^{(i+1)} \right) - \text{tr} \left(B^{*T} B^* \right) \right| \leq \frac{(\|\pi^{(k)}\|_F^2 - \|\pi^{(0)}\|_F^2) / \beta^{(0)}}{k} \quad (29)$$

By the boundedness of $\|\pi^{(k)}\|_F^2 - \|\pi^{(0)}\|_F^2$, we complete the proof. \square

Note that although our convergence proof assumes that each subproblem is solved exactly, this is not required in practise, because the inexact augmented Lagrange multipliers method has been shown to converge practically as fast as the exact augmented Lagrange multipliers [Lin et al. 2010]. Meanwhile, inexact augmented Lagrange multipliers require significantly fewer iterations when solving the subproblem, leading to much higher efficiency.

Complexity Analysis: Each update on B in Equation (14) takes $\mathcal{O}(r^2 m)$ time, while each update on L consumes $\mathcal{O}(r^2 n)$ time. Assuming that Algorithm 1 converges to a local minimum within N_{in} inner iterations (at line 3 in Algorithm 1) and N_{out} outer iterations (line 2 in Algorithm 1), the overall complexity of Algorithm 1 is $\mathcal{O}(N_{in} \times N_{out} \times (r^2 m + r^2 n))$.

6. LRM UNDER (ϵ, δ) -DIFFERENTIAL PRIVACY

This section extends LRM to (ϵ, δ) -differential privacy. Section 6.1 formulates the workload decomposition as an optimization program. Section 6.2 analyzes the utility of LRM. Section 6.3 discusses the algorithm for solving optimal workload decomposition.

6.1. Workload Decomposition

Similar to the case of ϵ -differential privacy described in section 4, LRM decomposes the workload matrix W into $W = BL$. Then, LRM applies the Gaussian mechanism to the intermediate queries corresponding to L to enforce (ϵ, δ) -differential privacy. Finally, LRM combines the noisy results of the intermediate queries according to B , to obtain the results of Q . Formally, let $\Theta(L)$ be the \mathcal{L}_2 sensitivity of L , i.e., $\Theta(L) = \max_j (\sum_i L_{ij}^2)^{1/2}$. LRM under (ϵ, δ) -differential privacy is defined as follows.

$$M_{LRM,(\epsilon,\delta)}(Q, D) = B \left(LD + \text{Gau} \left(\frac{\Theta(L)}{h(\epsilon, \delta)} \right)^r \right) \quad (30)$$

where $h(\epsilon, \delta) = \frac{\epsilon}{\sqrt{8 \ln(2/\delta)}}$.

Let $\Phi(B)$ be scale of the decomposition as defined in Definition 4.1, i.e., $\Phi(B) = \sum_{i,j} B_{ij}^2$. The following lemma shows that the error of LRM is linear to $\Phi(B)$, and quadratic to $\Theta(L)$.

LEMMA 6.1. *The expected squared error of $M_{LRM,(\epsilon,\delta)}(Q, D)$ with respect to the decomposition $W = BL$ is $8 \ln(2/\delta) \Phi_B \Theta(L)^2 / \epsilon^2$.*

PROOF. According to Equation (30), $Q(D) - M_{LRM,(\epsilon,\delta)}(Q, D) = B \left(\text{Gau} \left(\frac{\Theta(L)}{h(\epsilon,\delta)} \right)^r \right)$. The expected squared error of LRM is thus $\sum_{i,j} B_{ij}^2 \frac{2(\Theta(L))^2}{h(\epsilon,\delta)^2}$. Since $\Phi_B = \sum_{i,j} B_{ij}^2$ and $h(\epsilon,\delta) = \frac{\epsilon}{\sqrt{8 \ln(2/\delta)}}$, the error can be rewritten as $8 \ln(2/\delta) \Phi_B \Theta(L)^2 / \epsilon^2$. \square

Therefore, the best decomposition is the one that minimizes $\Phi_B \Theta(L)^2$. Similar to the case of ϵ -differential privacy, the particular value of $\Theta(L)$ is not important, as stated in the following lemma.

LEMMA 6.2. *Given a workload decomposition $W = BL$, we can always construct another decomposition $W = B'L'$ satisfying (i) $\Theta(L') = 1$ and (ii) (B', L') lead to the same expected squared error of $M_{LRM,(\epsilon,\delta)}$ as (B, L) .*

The proof is similar to that of Lemma 4.3, and omitted for brevity. Based on Lemma 6.2, we formulate the following optimization program for finding the best decomposition for $M_{LRM,(\epsilon,\delta)}$:

$$\begin{aligned} \min_{B,L} \quad & \frac{1}{2} \text{tr}(B^T B) \\ \text{s.t.} \quad & W = BL \\ & \forall j \sum_i^r L_{ij}^2 \leq 1 \end{aligned} \tag{31}$$

6.2. Utility Analysis and Budget Selection

This subsection analyzes the utility $M_{LRM,(\epsilon,\delta)}$, as well as the choice of the privacy parameters (ϵ, δ) given a user-specified utility constraint. We use (ξ, η) -usefulness (Definition 4.5) as the utility measure. The result is stated in the following theorem.

THEOREM 6.3. Utility of LRM under (ϵ, δ) -differential privacy. *Given database D and workload W , for any $\xi > 0$ and $0 < \eta < 1$, mechanism $M_{LRM,(\epsilon,\delta)}$ using the optimal decomposition $W = BL$ solved from Program (31) has the following utility guarantees: (i) when $\epsilon \geq \sqrt{6 \cdot \ln \frac{2}{\delta} \cdot (\frac{r}{2} \ln 3 - \ln \eta)} \|B\|_2 / \xi$, the output of $M_{LRM,(\epsilon,\delta)}$ is (ξ, η) -useful under the $\|\cdot\|_2$ -norm; (ii) when $\epsilon \geq \sqrt{(6 \ln r - 3 \ln 3)(\ln 2 - \ln \delta) / \eta} \|B\|_\infty / \xi$, the output of $M_{LRM,(\epsilon,\delta)}$ is (ξ, η) -useful under the $\|\cdot\|_\infty$ -norm.*

PROOF. (i) Let X be the Gaussian noise vector injected to the intermediate results in LRM. According to Equation (30), we have:

$$\|M_{LRM,(\epsilon,\delta)}(Q, D) - Q(D)\|_2^2 = \|B(LD + X) - WD\|_2^2 = \|B \cdot X\|_2^2 \leq \|B\|_2^2 \cdot \|X\|_2^2$$

The inequality above is due to the fact that $\|Rx\|_2 \leq \|R\|_2 \cdot \|x\|_2$, for any matrix R and vector x . Accordingly, we derive the following:

$$\begin{aligned} & \|M_{LRM,(\epsilon,\delta)}(Q, D) - Q(D)\|_2^2 \leq \|B\|_2^2 \cdot \|X\|_2^2 \\ \Rightarrow \forall \xi, \Pr(\|M_{LRM,(\epsilon,\delta)}(Q, D) - Q(D)\|_2^2 \geq \xi^2) & \leq \Pr(\|X\|_2^2 \cdot \|B\|_2^2 \geq \xi^2) \\ \Rightarrow \forall \xi, \Pr(\|M_{LRM,(\epsilon,\delta)}(Q, D) - Q(D)\|_2 \geq \xi) & \leq \Pr(\|X\|_2^2 \geq \frac{\xi^2}{\|B\|_2^2}) \end{aligned}$$

Next we focus on properties of X . According to the Gaussian mechanism, the elements of X , i.e., X_1, X_2, \dots, X_r follow i.i.d. zero-mean Gaussian distribution with scale $\sigma = \frac{\Theta(L)}{h(\epsilon, \delta)}$. Since the decomposition $W = BL$ is solved from Program 31, we have $\Theta(L) = 1$. Thus, $\sigma = \frac{1}{h(\epsilon, \delta)} = \frac{\epsilon}{\sqrt{2 \ln(2/\delta)}}$.

Let t, c be any positive number, we have:

$$\Pr(\|X\|_2^2 \geq c) = \Pr\left(\frac{\|X\|_2^2}{t\sigma^2} > \frac{c}{t\sigma^2}\right) = \Pr\left(e^{\frac{\|X\|_2^2}{t\sigma^2}} > e^{\frac{c}{t\sigma^2}}\right) \leq \frac{\mathbb{E}\left[e^{\frac{\|X\|_2^2}{t\sigma^2}}\right]}{e^{\frac{c}{t\sigma^2}}}$$

where the last inequality holds due to Markov's inequality.

Consider the random variable $Y_i = \exp\left(\frac{X_i^2}{t\sigma^2}\right)$, where t is an arbitrary positive number such that $\mathbb{E}[Y_i]$ exists. According to the probability density function of the Gaussian distribution (Equation (7)), we have:

$$\mathbb{E}[Y_i] = \int_{-\infty}^{\infty} g(x) e^{\left(\frac{x^2}{t\sigma^2}\right)} dx = \int_{-\infty}^{\infty} \sqrt{\frac{1}{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} e^{\frac{x^2}{t\sigma^2}} dx = \sqrt{\frac{t}{t-2}}, \forall t > 2$$

Based on the above derivations, and the fact that X_i 's are independent variables, we obtain:

$$\Pr(\|X\|_2^2 \geq c) \leq \frac{\prod_{i=1}^r (\mathbb{E} e^{\frac{X_i^2}{t\sigma^2}})}{e^{\frac{c}{t\sigma^2}}} = \frac{\prod_{i=1}^r \mathbb{E}[Y_i]}{e^{\frac{c}{t\sigma^2}}} = \frac{\left(\frac{t}{t-2}\right)^{r/2}}{e^{\frac{c}{t\sigma^2}}}$$

With the choice of $t = 3$, $c = \frac{\xi^2}{\|B\|_2^2}$, and $\sigma = \frac{\sqrt{2 \ln(2/\delta)}}{\epsilon}$, this leads to:

$$\Pr(\|M_{\epsilon, \delta}(Q, D) - Q(D)\|_2 \geq \xi) \leq \frac{\left(\frac{t}{t-2}\right)^{r/2}}{e^{\frac{c}{t\sigma^2}}} = \frac{3^{r/2}}{e^{\frac{\xi^2 \epsilon^2}{6 \ln(2/\delta) \|B\|_2^2}}}$$

When $\epsilon \geq \sqrt{6 \cdot \ln \frac{2}{\delta} \cdot \left(\frac{r}{2} \ln 3 - \ln \eta\right) \|B\|_2 / \xi}$, the above probability is bound by η .

(ii) Let X be the Gaussian noise vector injected to the intermediate results as in part (i) of the proof. We have:

$$\|M_{\epsilon, \delta}(Q, D) - Q(D)\|_{\infty}^2 = \|B \cdot X\|_{\infty}^2 \leq \|B\|_{\infty}^2 \cdot \|X\|_{\infty}^2 = \|\sigma B\|_{\infty}^2 \cdot \left\|\frac{1}{\sigma} X\right\|_{\infty}^2$$

The above inequality holds due to the fact that $\|Rx\|_{\infty} \leq \|R\|_{\infty} \cdot \|x\|_{\infty}$ for any matrix R and vector x . Let $Z = \left\|\frac{1}{\sigma} X\right\|_{\infty}^2 = \left(\max\left(\frac{1}{\sigma} X_1, \dots, \max\left(\frac{1}{\sigma} X_r\right)\right)\right)^2$. We derive:

$$\begin{aligned} & \|M_{\epsilon, \delta}(Q, D) - Q(D)\|_{\infty}^2 \leq \|\sigma B\|_{\infty}^2 \cdot \left\|\frac{1}{\sigma} X\right\|_{\infty}^2 \\ \Rightarrow \forall \xi, & \Pr(\|M_{\epsilon, \delta}(Q, D) - Q(D)\|_{\infty}^2 \geq \xi^2) \leq \Pr(\|\sigma B\|_{\infty}^2 \cdot Z \geq \xi^2) \\ \Rightarrow \forall \xi, & \Pr(\|M_{\epsilon, \delta}(Q, D) - Q(D)\|_{\infty} \geq \xi) \leq \Pr\left(Z \geq \frac{\xi^2}{\|\sigma B\|_{\infty}^2}\right) \end{aligned}$$

By Markov's inequality, we obtain:

$$\Pr\left(Z \geq \frac{\xi^2}{\|\sigma B\|_{\infty}^2}\right) \leq \frac{\mathbb{E}[Z]}{\frac{\xi^2}{\|\sigma B\|_{\infty}^2}}$$

Note that the above bound is tight, even though Chernoff bound can not be applied here.

Next we derive an upper bound for the expected value of Z . Let $Y = \frac{1}{\sigma} X$. Clearly, Y_1, Y_2, \dots, Y_r are independent, standard normal random variables. Hence, Y_i^2 's ($1 \leq i \leq r$) are i.i.d. χ_1^2 variables,

i.e., Chi-square random variables with 1-degree of freedom. The probability density function f_i for Y_i is thus:

$$f_i(x) = \frac{1}{\sqrt{2\pi}} x^{-\frac{1}{2}} e^{-\frac{x}{2}}$$

Since the function $\exp(\cdot)$ is convex and positive, by Jensen's inequality, for any t such that $\mathbb{E}[e^{tZ}]$ exists, we have:

$$e^{t\mathbb{E}[Z]} \leq \mathbb{E}[e^{tZ}] = \mathbb{E}[\max_i e^{tY_i^2}] \leq \sum_{i=1}^r \mathbb{E}[e^{tY_i^2}] \quad (32)$$

Meanwhile, for any $t < \frac{1}{2}$, we have $\mathbb{E}[e^{tY_i^2}] = \int_0^{+\infty} e^{tx} \frac{1}{\sqrt{2\pi}} x^{-\frac{1}{2}} e^{-\frac{x}{2}} dx = (1 - 2t)^{-\frac{1}{2}}$. Combine this with Equation (32), we obtain an upper bound of the expected value of Z :

$$\mathbb{E}[Z] \leq \frac{\ln r}{t} - \frac{1}{2t} \ln(1 - 2t)$$

With the choice of $t = \frac{1}{3}$, we have $\mathbb{E}[Z] \leq 3 \ln r + \frac{3}{2} \ln 3$. Since $\forall j \sum_i L_{ij}^2 \leq 1$, the sensitivity over the batch query workload Q is 1. Since $\sigma = \frac{\sqrt{2 \ln(2/\delta)}}{\epsilon}$, we obtain the following:

$$\begin{aligned} \forall \xi, \Pr(\|M_{\epsilon, \delta}(Q, D) - Q(D)\|_{\infty} \geq \xi) &\leq \mathbb{E}[Z] \cdot \|\sigma B\|_{\infty}^2 / \xi^2 \\ &\leq \left(3 \ln r + \frac{3}{2} \ln 3\right) \cdot \|\sigma B\|_{\infty}^2 / \xi^2 \\ &= \left(3 \ln r + \frac{3}{2} \ln 3\right) \cdot (2 \ln(2/\delta)) \cdot \|B\|_{\infty}^2 / (\xi \epsilon)^2 \end{aligned}$$

When $\epsilon \geq \sqrt{(6 \ln r - 3 \ln 3)(\ln 2 - \ln \delta) / \eta} \|B\|_{\infty} / \xi$, the above probability is bound by η . \square

6.3. Solving for the Optimal Workload Decomposition

The optimization program (i.e., Program (31)) for workload decomposition under (ϵ, δ) -differential privacy is identical to the one under ϵ -differential privacy (Program (9)), except that the former uses \mathcal{L}_2 sensitivity in the constraints $\forall j \sum_i L_{ij}^2 \leq 1$ whereas the latter uses \mathcal{L}_1 sensitivity. Hence, to solve Program (31), we simply adapt Algorithm 1 by modifying the parts related to these constraints.

The only major modification of Algorithm 1 lies in the projection step, which now needs to project every column in L onto the \mathcal{L}_2 ball of radius 1, instead of the \mathcal{L}_1 unit ball as in Section 5. Specifically, the \mathcal{L}_2 ball projection is performed by solving the following optimization program:

$$\min_{\bar{L} \in \mathbb{R}^{r \times n}} \|\bar{L} - L\|_F^2, \text{ s.t. } \forall j \sum_i \bar{L}_{ij}^2 \leq 1 \quad (33)$$

The above program can be decoupled into n independent \mathcal{L}_2 regularized sub-problems:

$$\arg \min_{\bar{l} \in \mathbb{R}^{r \times 1}} \|\bar{l} - l\|_2^2, \text{ s.t. } \sum_i \bar{l}_i^2 \leq 1$$

where $l = L_j^{(t)}$, $j = 1, 2, \dots, n$, $L_j^{(t)}$ is the j^{th} column of $L^{(t)}$. Such a projection can be computed by $\bar{l} = \frac{l}{\max(1, \|l\|_2)}$. Therefore, the projection can be computed efficiently in linear time. Finally, by adapting the proofs in section 5.2, we can draw the conclusion that the modified Algorithm 1 for optimizing workload decomposition for LRM under (ϵ, δ) -differential privacy also converges to the a local KKT optimal point linearly. We omit the complete proofs for brevity.

7. EXPERIMENTS

This section experimentally evaluate the effectiveness of LRM under ϵ - and (ϵ, δ) - differential privacy definitions. For ϵ -differential privacy, we compares LRM against six state-of-the-art methods: Laplace mechanism (LM) [Dwork et al. 2006c], Privlet (WM) [Xiao et al. 2010], hierarchical mechanism (HM) [Hay et al. 2010], exponential smoothing (ESM) [Yuan et al. 2012] (an implementation of the approximate matrix mechanism [Li et al. 2010], described in Appendix A.1), adaptive mechanism (AM) [Li and Miklau 2012] (another implementation of the approximate matrix mechanism [Li et al. 2010; Li and Miklau 2012], described in Appendix A.2) and the exponential mechanism with multiplicative weights update (MWEM) [Hardt et al. 2012], whose performance depends on the dataset. For (ϵ, δ) -differential privacy, we compare LRM against WM, HM, ESM, AM, and the Gaussian mechanism (GM) [McSherry and Mironov 2009].

Implementations: For AM, we employ the Python implementation that can be obtained from the authors' website (<http://cs.umass.edu/~chaoli>). We use the default stopping criterion provided by the authors. For MWEM, we used Hardt et al's C# code listed in the Appendix of [Hardt et al. 2012]). Note that MWEM needs to tune an additional parameter T which denotes the number of iterations in order to ensure its performance. We follow the experimental setting in [Hardt et al. 2012]. Specifically, we choose $T \in \{10, 12, 14, 16\}$ in our experiments and reported the values for the best setting of T in each case (Strictly speaking, such parameter tuning violates differential privacy; hence, the reported results are in favor on MWEM). For all remaining methods, we implemented them in Matlab, and published all code online (<http://yuanganzhao.weebly.com/>). We performed all experiments on a desktop PC with an Intel quad-core 2.50 GHz CPU and 4GBytes RAM. In each experiment, every algorithm is executed 20 times and the average performance is reported.

Datasets: We use four real-world data sets in our experiments [Hay et al. 2010; Xu et al. 2013; Hardt et al. 2012]: *Search Log*, *Net Trace*, *Social Network* and *UCI Adult*. *Search Log* includes search keyword statistics collected from *Google Trends* and *American Online* between 2004 and 2010. Each unit count is the number of appearances of a particular keyword. *Social Network* contains information about users in a social network, where each unit count is the number of users with a specific degree in the social graph. *Net Trace* is collected from a university intranet, where each unit count is the number of TCP packets related to a particular IP address. The total number of unit counts in *Search Logs*, *Net Trace* and *Social Network* are 65, 536, 32, 768 and 11, 342 respectively. The UCI Adult data was extracted from the census bureau database in the U.S. Department of Commerce, it contains 14 features, among which six are continuous and eight are categorical. We use the following strategies to generate the sensitive data with varying domain size n . For the $\{\textit{Search Log}, \textit{Net Trace}, \textit{Social Network}\}$ data sets, we transform the original counts into a vector of fixed size n (domain size), by merging consecutive counts in order. For the *UCI Adult* data set, we only consider the combined $\{\textit{workclass}, \textit{education}, \textit{occupation}, \textit{race}\}$ attributes (with their total corresponding domain of size $\{8 \times 16 \times 14 \times 5 = 8960\}$) and uniformly choose n domains. The counting numbers of their corresponding records are used as the domain data. We observed that all the data sets $\{\textit{Search Log}, \textit{Net Trace}, \textit{Social Network}\}$ are dense with their sparsity exactly equals to 100%, while the *UCI Adult* data set is sparse with its sparsity roughly $12\% \sim 17\%$.

Workloads: We generated four different types of workloads, namely *WDiscrete*, *WRange*, *WMarginal* and *WRelated*. In *WDiscrete*, for each W_{ij} (i.e., the coefficient of the i -th query on the j -th unit count), we set $W_{ij} = 1$ with probability 0.02 and $W_{ij} = -1$ otherwise. In *WRange*, each query q_i sums the unit counts in a range $[s_i, t_i] \subset [1, n]$, i.e., $W_{ij} = 1$ for $s_i \leq j \leq t_i$, and $W_{ij} = 0$ otherwise. The start and end points s_i and t_i of each query q_i is randomly generated, following the uniform distribution. *WMarginal* is used in [Li and Miklau 2012], which contain queries that are uniformly sampled from the set of all 2-way marginals. For *WRelated*, we generate s independent linear counting queries (called *base queries*) with random weights following $(0, 1)$ -normal distribution. Let A (of size $s \times n$) denote the workload matrix of the s queries. We also generate

another matrix C of size $m \times s$ in a similar way. The workload matrix W is then the product of C and A , i.e., the linear combination of base queries according to C .

Parameters: We test the impact of five parameters in our experiments: γ , r , n , m and s . γ is the relaxation factor defined in Program (12). r is the number of intermediate queries in LRM, i.e., the number of columns in B (and also the number of rows in L). n is the number of unit counts and m is the number of queries in the batch. Finally, s is the number of base queries during the generation of $W_{Related}$. The ranges and defaults (shown in bold) of the parameters are summarized in Table II. Moreover, we test three different values of the privacy budget: $\epsilon = 1, 0.1$ and 0.01 . For (ϵ, δ) -differential privacy, following [Li and Miklau 2012], we set $\delta = 0.0001$.

Table II. Parameters used in the experiments.

γ	0.0001, 0.001, 0.01 , 0.1, 1, 10
r	$\{0.8, \mathbf{1.0}, \mathbf{1.2}, 1.4, 1.7, 2.1, 2.5, 3.0, 3.6\} \times rank(W)$
n	128, 256, 512, 1024 , 2048, 4096, 8192
m	64, 128, 256 , 512, 1024
s (during the generation of $W_{Related}$)	$\{0.1, 0.2, 0.3, 0.4, \mathbf{0.5}, 0.6, 0.7, 0.8, 0.9, 1.0\} \times \min(m, n)$

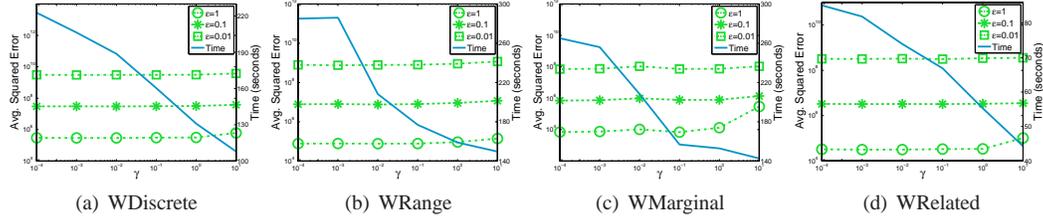
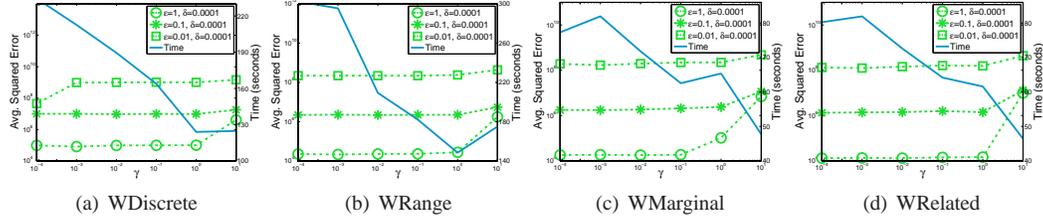
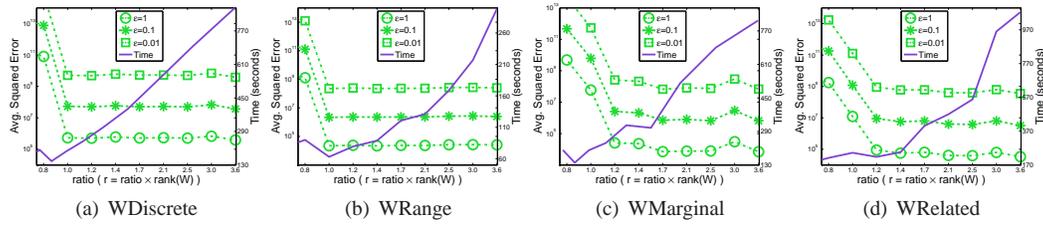
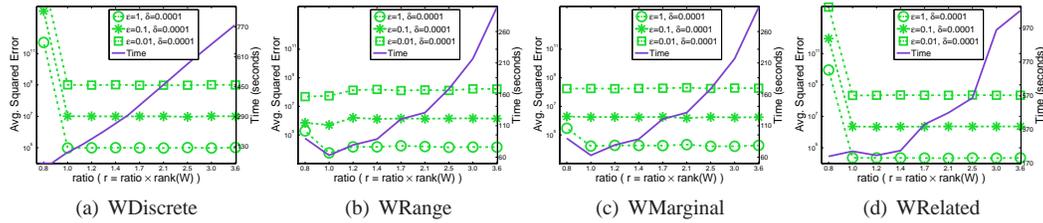
In the experiments, we measure average squared error and computation time of the methods. Specifically, the average squared error is the average squared \mathcal{L}_2 distance between the exact query answers and the noisy answers. In the following, Section 7.1 examines the impact of γ and r , which are only used in LRM. The results provide important insights on how to set these two parameters to maximize the utility of LRM. Then, Sections 7.2 to 7.5 compare LRM against existing methods.

7.1. Impact of γ and r on LRM

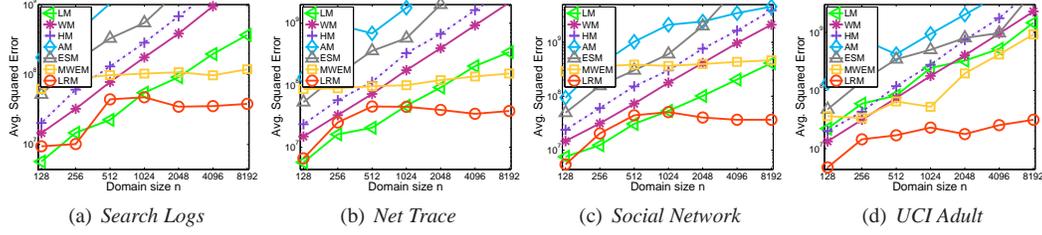
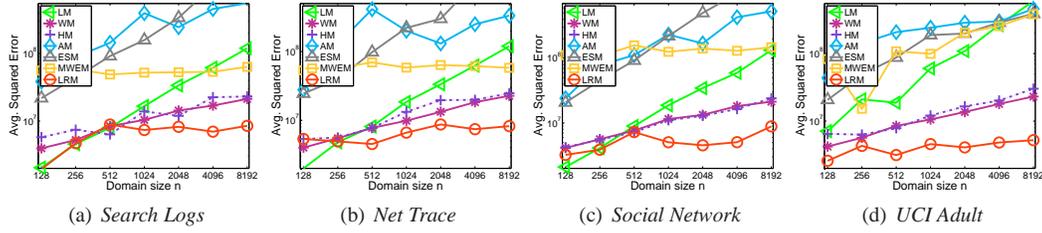
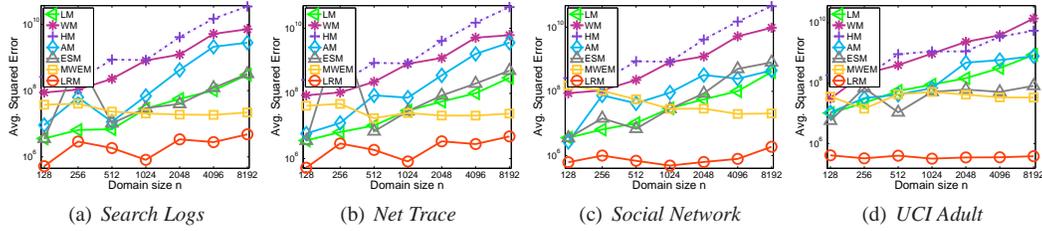
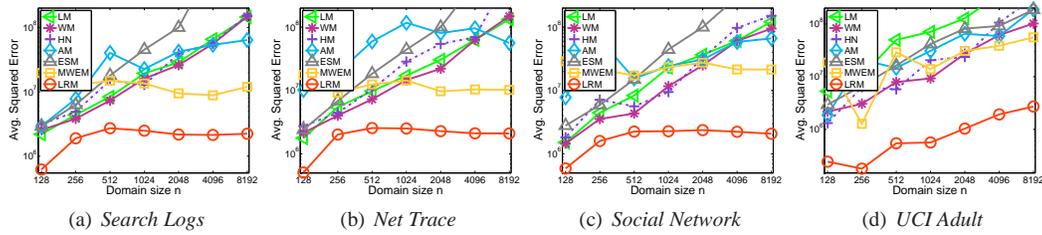
In LRM, the relaxation factor γ controls the difference between BL and W . In our first set of experiments, we investigate the impact of γ on the accuracy and efficiency of LRM. Figure 2 and Figure 3 report the performance of LRM with varying values for γ under ϵ -differential privacy and (ϵ, δ) -differential privacy respectively, using the *Search Logs* dataset. Results on other datasets lead to similar conclusions, and are omitted for brevity.

The results in the Figure 2 and Figure 3 show that when ϵ is relatively low (meaning strong privacy), the error of LRM is not sensitive to γ regardless of the workload, for all values of γ tested in the experiments (10^{-4} to 10). Only when ϵ reaches 1 does large values of γ (e.g., $\gamma > 1$) show negative impact on the performance of LRM. This negative effect is relatively small under ϵ -differential privacy; it is more pronounced under (ϵ, δ) -differential privacy. The reason is that the error of LRM comes from two sources: the added noise and the difference between the decomposition BL and the original workload W . When the privacy requirement is strong (i.e., when ϵ is relatively low, or when ϵ -differential privacy is used), the error introduced by inexact decomposition is negligible compared to the noise added to satisfy differential privacy. Conversely, with looser privacy requirement (high ϵ and (ϵ, δ) -differential privacy definition), the noise level becomes low, and the error in decomposition becomes more evident. Nevertheless, when $\gamma \leq 0.1$, its impact is insignificant in all settings. Meanwhile, LRM runs much faster with a larger γ . Overall, $\gamma \leq 0.1$ is a safe choice, and a larger value of γ is recommended for applications with strong privacy requirements. In the following experiments, we fix γ to 0.01.

r is another important parameter in LRM that determines the rank of the matrix BL that approximates the workload W . r affects both the approximation accuracy and the optimization speed. When r is too small, e.g., when $r < rank(W)$, our optimization formulation may fail to find a good approximation, leading to suboptimal accuracy for the query batch. On the other hand, an overly large r leads to poor efficiency, as the search space expands dramatically. We thus test LRM with varying r , by controlling the ratio of r to the actual rank $rank(W)$, on the *Search Log* dataset. We record the average squared error and running time of LRM for all the workloads under ϵ and (ϵ, δ) -differential privacy, and report them in Figure 4 and Figure 5 respectively.

Fig. 2. Effect of relaxation parameter γ on *Search Logs* under ϵ -differential privacyFig. 3. Effect of relaxation parameter γ on *Search Logs* under (ϵ, δ) -differential privacyFig. 4. Effect of r on *Search Logs* under ϵ -differential privacyFig. 5. Effect of r on *Search Logs* under (ϵ, δ) -differential privacy

There are several important observations in Figure 4 and Figure 5. First, a value of r below $\text{rank}(W)$ leads to far worse accuracy (up to two orders of magnitude) compared to settings with higher values of r . Second, the performance of LRM becomes stable when r exceeds $1.2 \cdot \text{rank}(W)$ for ϵ -differential privacy, and $1.0 \cdot \text{rank}(W)$ for (ϵ, δ) -differential privacy. This is because the optimization formulation has enough freedom to find the optimal decomposition when $r > \text{rank}(W)$. For (ϵ, δ) -differential privacy, this result is expected, because any decomposition $W = BL$ with $r > \text{rank}(W)$ can be transformed into a decomposition $B'L'$ with $r = \text{rank}(W)$, by projecting the columns of L and the rows of B onto the range of L , which does not affect the \mathcal{L}_2 -sensitivity of B . Finally, the amount of computations for workload decomposition increases linearly with r (note that both axes are in logarithmic scale). Thus, to balance the efficiency and effectiveness of LRM, a good value for r is between $\text{rank}(W)$ and $1.2 \cdot \text{rank}(W)$. In subsequent experiments, we set $r = 1.2 \cdot \text{rank}(W)$ and $r = 1.0 \cdot \text{rank}(W)$ for ϵ and (ϵ, δ) -differential privacy, respectively.

Fig. 6. Effect of domain size n on workload $WDiscrete$ under ϵ -differential privacy with $\epsilon = 0.1$ Fig. 7. Effect of domain size n on workload $WRange$ under ϵ -differential privacy with $\epsilon = 0.1$ Fig. 8. Effect of domain size n on workload $WMarginal$ under ϵ -differential privacy with $\epsilon = 0.1$ Fig. 9. Effect of domain size n on workload $WRelated$ under ϵ -differential privacy with $\epsilon = 0.1$

7.2. Impact of Varying Domain Size n

We now evaluate the accuracy performance of all mechanisms with varying domain size n . We perform all experiments with $\epsilon = 0.1$, since the specific value of ϵ has negligible impact on the relative performance of different mechanisms. For ϵ -differential privacy, we report the results of all mechanisms on the 4 different workloads in Figures 6, 7, 8 and 9, respectively. On workloads $WMarginal$ and $WRelated$, the performance of AM and ESM is comparable to the naive Laplace mechanism, and significantly worse than the other methods, sometimes by more than an order of magnitude. This is mainly because the \mathcal{L}_2 approximation used by AM and ESM does not lead to a good optimization of the actual objective function formulated using \mathcal{L}_1 sensitivity. On $WDiscrete$,

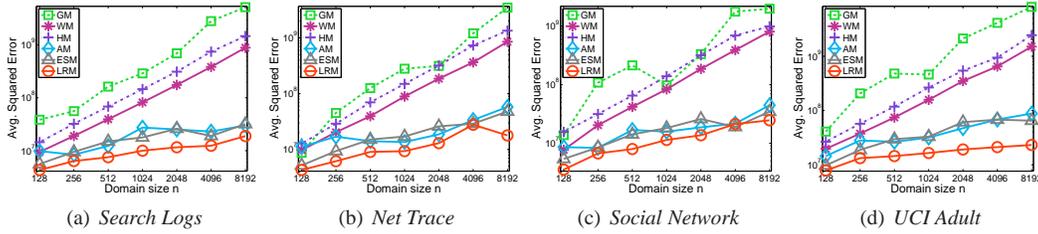


Fig. 10. Effect of domain size n on workload $WDiscrete$ under (ϵ, δ) -differential privacy with $\epsilon = 0.1$ and $\delta = 0.0001$

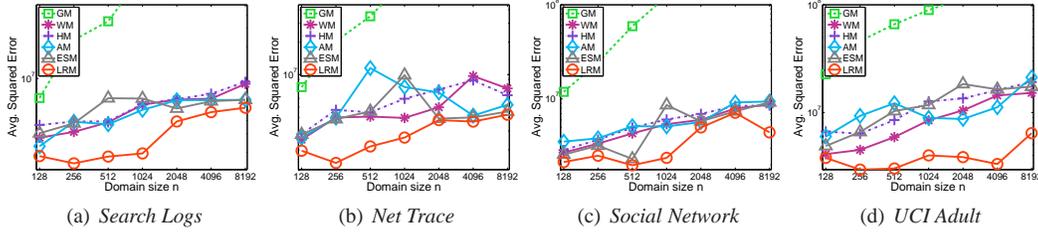


Fig. 11. Effect of domain size n on workload $WRange$ under (ϵ, δ) -differential privacy with $\epsilon = 0.1$ and $\delta = 0.0001$

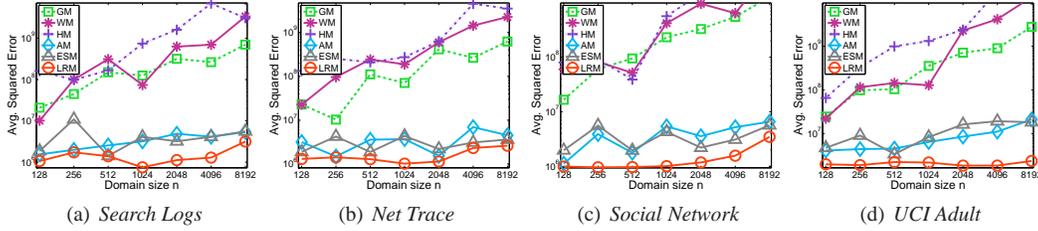


Fig. 12. Effect of domain size n on workload $WMarginal$ under (ϵ, δ) -differential privacy with $\epsilon = 0.1$ and $\delta = 0.0001$

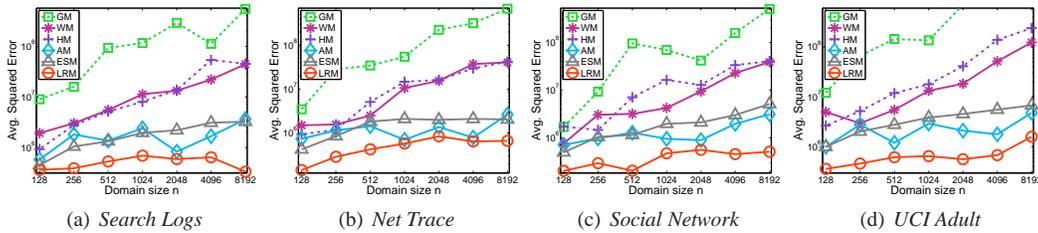


Fig. 13. Effect of domain size n on workload $WRelated$ under (ϵ, δ) -differential privacy with $\epsilon = 0.1$ and $\delta = 0.0001$

the Laplace mechanism outperforms all other mechanisms when the data is non-sparse and domain size is relatively small. This is in part due to the fact that the queries in $WDiscrete$ are generally independent when $m \geq n$. Since the other mechanisms do not gain from correlations among queries, Laplace mechanism is optimal in such a situation. Whereas all other data-independent mechanisms incur an error linear to the domain size n , LRM's error stops increasing when the domain size reaches 512. This is because LRM's error rate depends on the rank of the workload matrix W , which is no larger than $\min(m, n)$. This explains the excellent performance of LRM in larger domains. On $WRange$, the errors of WM and HM are smaller than that of the Laplace mechanism when the domain size is no smaller than 512. Moreover, WM and HM perform better on $WRange$ than on

the others workloads, since they are designed to optimize mainly for range queries. Nonetheless, LRM's performance is significantly better than any of them, since it fully utilizes the correlations between the range queries on large domains. On *WMarginal* and *WRelated*, LRM achieves the best performance in all settings. The performance gap between LRM and other methods is over two orders of magnitude when the domain size reaches 8192. Since *WRelated* naturally leads to a low rank workload matrix W , this result verifies LRM's vast benefit from exploiting the low-rank property of the workload. Finally, we observe some interesting behaviors of the data-dependent method MWEM. The error incurred by MWEM does not scale well with the domain size n on non-sparse data sets. Moreover, MWEM performs comparably to LRM on *Search Logs* and *Net Trace* when the n is very large ($n \geq 4096$). However, the performance of MWEM is rather unstable; it incurs much larger error than LRM on *Social Network* and *UCI Adult*, in some cases by more than two order of magnitude.

Regarding (ϵ, δ) -differential privacy, we report the accuracy of all methods in Figures 10, 11, 12 and 13. LRM obtains the best performance in all settings, especially when n is large. Its improvement over the naive Gaussian mechanism is over two orders of magnitude. AM and ESM have similar accuracy. For range queries, the performance of ESM and AM is comparable to that of WM and HM, which are optimized for range counts. However, the accuracy of AM and ESM is rather unstable on workloads *WRange* and *WMarginal*. For ESM, this instability is caused by numerical errors in the matrix inverse operations, which can be high when the final solution matrix is low-rank. For AM, the problem is with its post-processing step, which gives approximation solutions with unstable quality. The performance of LRM, on the other hand, is consistently good in all settings.

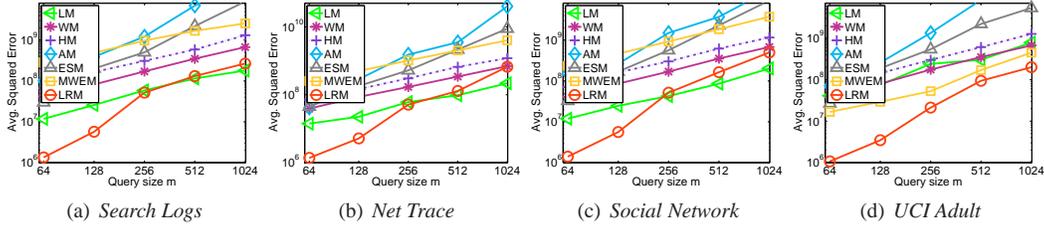
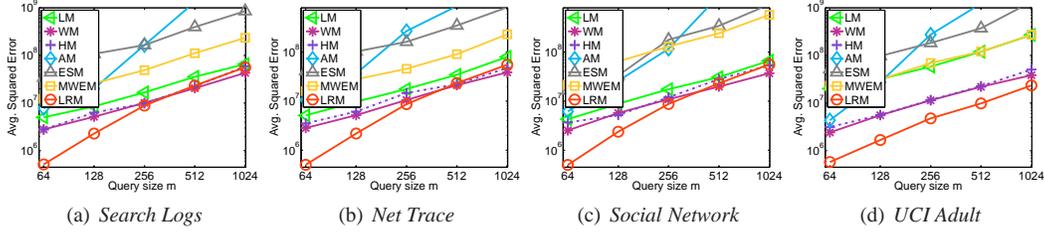
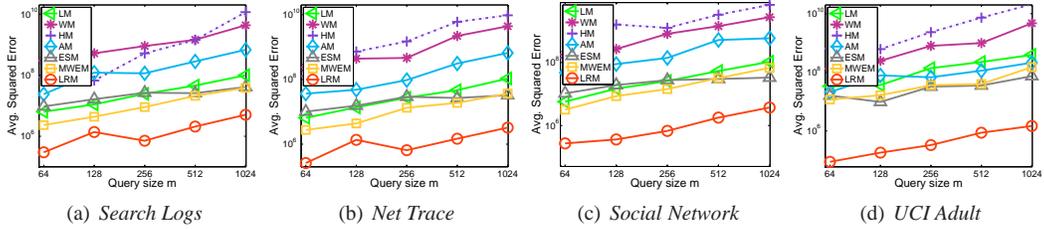
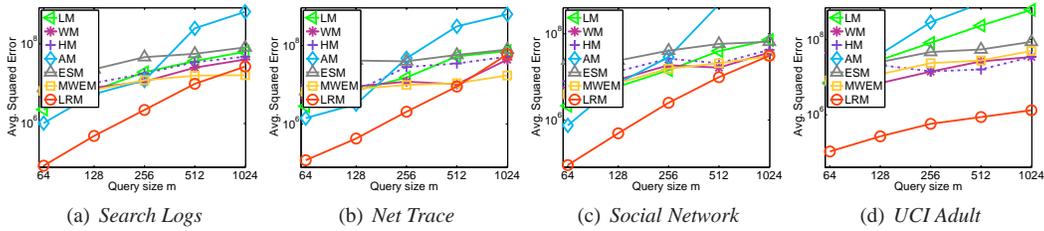
7.3. Impact of Number of Queries m

In this subsection, we test the impact of the query set cardinality m on the performance of the mechanisms. We mainly focus on settings when the number of queries m is no larger than the domain size n . For ϵ -differential privacy, the accuracy results are reported in Figures 14, 15, 16 and 17. On *WRange* and *WMarginal*, LRM outperforms all other mechanisms, when m is significantly smaller than n . As m grows, the performance of all mechanisms on *WRange* tends to converge. The degeneration in performance of LRM is due to the lack of low rank property when the batch contains too many random range queries. When m is no less than 256, both the WM and HM achieve comparable accuracy to LRM, since they are optimized for range queries. On *WDiscrete*, MWEM is comparable to LRM on *UCI Adult* data set, one possible reason is that MWEM can make use of the sparsity of the data on *WDiscrete* workload. On *WRelated* workload, the accuracy of LRM is dramatically higher than the other methods, for all values of m . This is because the rank of the *WRelated* workload is fixed to s , regardless of the number of queries. Finally, we observe that on *WDiscrete* and *WRange*, while the performance of other mechanisms does not differ much from data to data, the data-dependent method MWEM generally performs better on the *UCI Adult* dataset compared to on other datasets, due to the high sparsity of *UCI Adult*.

For (ϵ, δ) -differential privacy, we report the results in Figures 18, 19, 20 and 21. We have the following observations from these results. On *WDiscrete*, *WRange* and *WRelated* workload, WM and HM improve upon the naive Gaussian mechanism; however, on *WMarginal*, WM and HM incur higher errors than GM. AM and ESM again exhibit similar performance, which is often better than that of WM, HM, and GM. LRM consistently outperforms its competitors in all test cases.

7.4. Impact of Varying Query Rank s

The previous experiments demonstrate LRM's substantial performance advantages when the workload matrix has low rank. In this set of experiments, we manually control the rank of workload W to verify this observation. Recall that the parameter s determines the size of the matrix $C_{m \times s}$ and the size of the matrix $A_{s \times n}$ during the generation of the *WRelated* workload. When C and A contain only independent rows/columns, s is exactly the rank of the workload matrix $W = CA$. In Figure 22 and 23, we vary s from $0.1 \times \min(m, n)$ to $1 \times \min(m, n)$.


 Fig. 14. Effect of number of queries m on workload $W_{Discrete}$ under ϵ -differential privacy with $\epsilon = 0.1$

 Fig. 15. Effect of number of queries m on workload W_{Range} under ϵ -differential privacy with $\epsilon = 0.1$

 Fig. 16. Effect of number of queries m on workload $W_{Marginal}$ under ϵ -differential privacy with $\epsilon = 0.1$

 Fig. 17. Effect of number of queries m on workload $W_{Related}$ under ϵ -differential privacy with $\epsilon = 0.1$

For ϵ -differential privacy, LRM outperforms all other methods by at least one order of magnitude when s is low. With increasing s , the performance gap gradually closes. This phenomenon confirms that the low rank property is the main reason behind LRM's advantages. For (ϵ, δ) -differential privacy, LRM also gives the best performance in all test cases; its performance advantage decreases with s , though at a much slower rate compared to the case of ϵ -differential privacy.

7.5. Scalability of the Low-Rank Mechanism

Finally, we demonstrate the efficiency and scalability of LRM under ϵ - and (ϵ, δ) -differential privacy. The running time of LRM is dominated by the optimization module that solves the best workload

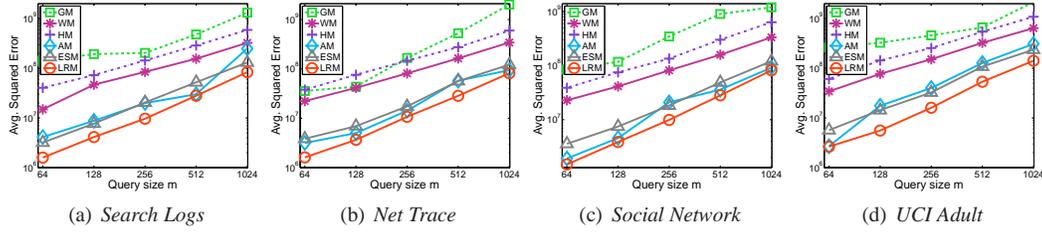


Fig. 18. Effect of number of queries m on workload $WDiscrete$ under (ϵ, δ) -differential privacy with $\epsilon = 0.1$ and $\delta = 0.0001$

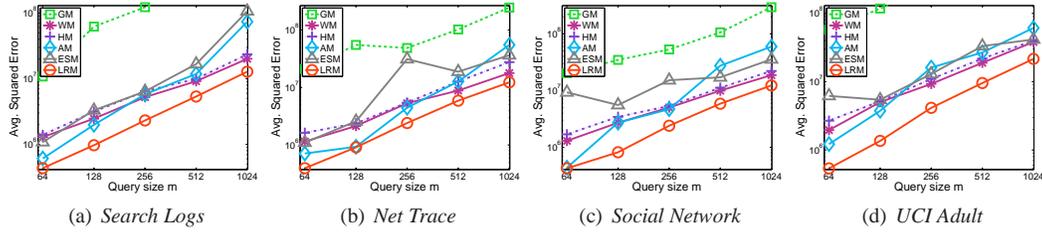


Fig. 19. Effect of number of queries m on workload $WRange$ under (ϵ, δ) -differential privacy with $\epsilon = 0.1$ and $\delta = 0.0001$

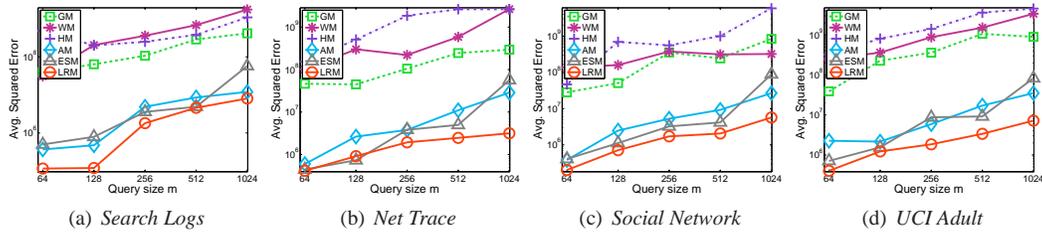


Fig. 20. Effect of number of queries m on workload $WMarginal$ under (ϵ, δ) -differential privacy with $\epsilon = 0.1$ and $\delta = 0.0001$

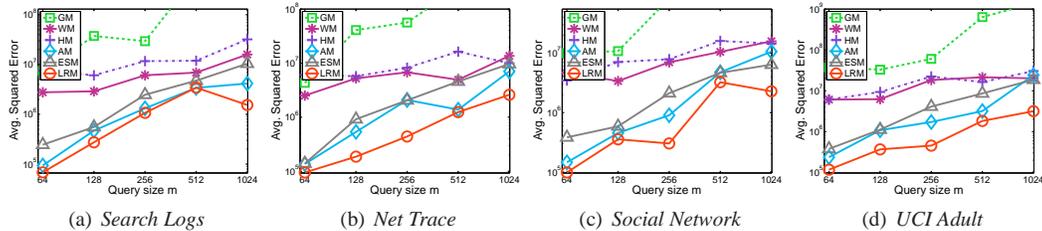


Fig. 21. Effect of number of queries m on workload $WRelated$ under (ϵ, δ) -differential privacy with $\epsilon = 0.1$ and $\delta = 0.0001$

decomposition, which is independent of the dataset. In Figure 24 and Figure 25, we vary the domain size n from 128 to 8192 and the number of queries m from 64 to 256, respectively, and report the total running time of LRM for the 4 different types of workloads in our experiments. LRM scales roughly linearly with the domain size n and the number of queries m (note that both axes are in logarithmic scale). Moreover, we observe that for workload $WRelated$, LRM runs faster when the rank s of the workload is lower, given the same values of n and m . LRM under (ϵ, δ) -differential privacy is slightly more efficient than under ϵ -differential privacy. This is expected, since we set a smaller

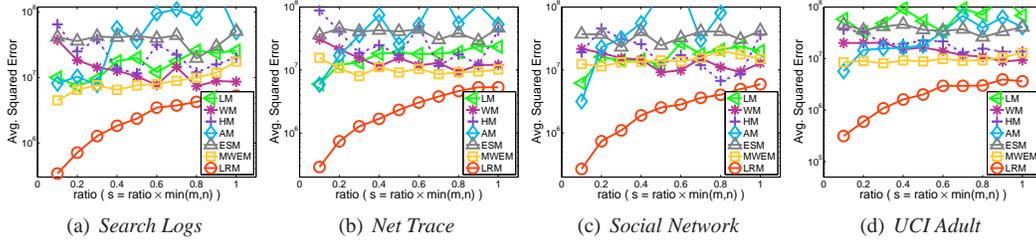


Fig. 22. Effect of parameter s with under ϵ -differential privacy with $\epsilon = 0.1$

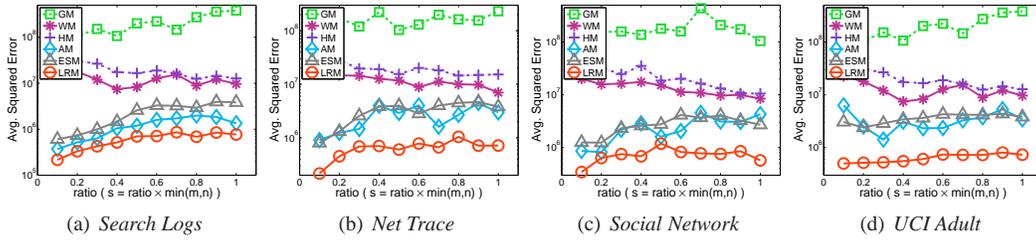


Fig. 23. Effect of parameter s under (ϵ, δ) -differential privacy with $\epsilon = 0.1$ and $\delta = 0.0001$

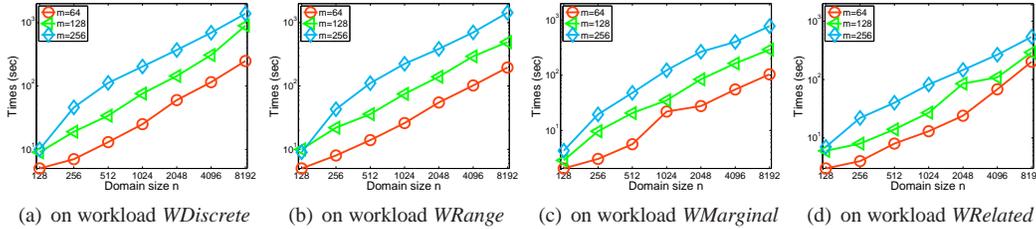


Fig. 24. Scalability of LRM under ϵ -differential privacy

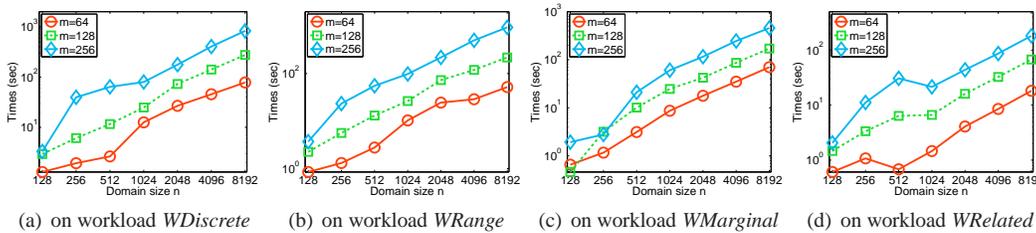


Fig. 25. Scalability of LRM under (ϵ, δ) -differential privacy

value of r for (ϵ, δ) -differential privacy. In all settings, LRM always terminates within 20 minutes for each experiment. In practice, this computation time pays off as LRM achieves significantly higher accuracy than existing methods.

8. CONCLUSIONS AND FUTURE WORK

This paper presents the low rank mechanism (LRM), an optimization framework that minimizes the overall error of the results for a batch of linear queries under differential privacy. The proposed method is the first practical method for a large number of linear queries, with an efficient and effective implementation using well established optimization techniques. Experiments show that

LRM significantly outperforms other state-of-the-art differentially private query processing mechanisms, often by orders of magnitude. The current design of LRM focuses on exploiting the correlations between different queries. One interesting direction for future work is to further optimize LRM by utilizing also the correlations between data values, e.g., as is done in [Xu et al. 2013; Rastogi and Nath 2010; Li et al. 2011].

References

- BALL, K. 1997. An elementary introduction to modern convex geometry. *Flavors of geometry* 31, 1–58.
- BECK, A. AND TEOULLE, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences (SIIMS)* 2, 1, 183–202.
- BELTRÁN, C. 2011. Estimates on the condition number of random rank-deficient matrices. *IMA Journal of Numerical Analysis (IMAJNA)* 31, 1, 25–39.
- BERTSEKAS, D. P. 1999. *Nonlinear programming*. Athena Scientific.
- BHASKAR, R., LAXMAN, S., SMITH, A., AND THAKURTA, A. 2010. Discovering frequent patterns in sensitive data. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 503–512.
- BILLINGSLEY, P. 2012. *Probability and Measure*. Vol. 939. John Wiley & Sons, New York.
- BIRGIN, E. G., MARTÍNEZ, J. M., AND RAYDAN, M. 2000. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization (SIOPT)* 10, 4, 1196–1211.
- BLUM, A., LIGETT, K., AND ROTH, A. 2008. A learning theory approach to non-interactive database privacy. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*. 609–618.
- CANDÈS, E. J. AND RECHT, B. 2009. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics (FoCM)* 9, 6, 717–772.
- CHAUDHURI, K., MONTELEONI, C., AND SARWATE, A. D. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research (JMLR)* 12, 1069–1109.
- CHEN, Z. AND DONGARRA, J. J. 2005. Condition numbers of gaussian random matrices. *SIAM Journal on Matrix Analysis and Applications (SIMAX)* 27, 3, 603–620.
- CONN, A. R., GOULD, N., PH, AND TOINT, L. 1997. A globally convergent lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. *Mathematics of Computation* 66, 217, 261–288.
- CORMODE, G., PROCOPIUC, C. M., SHEN, E., SRIVASTAVA, D., AND YU, T. 2012. Differentially private spatial decompositions. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)*. 20–31.
- D’ASPROMONT, A., GHAOUI, L. E., JORDAN, M. I., AND LANCKRIET, G. R. G. 2007. A direct formulation for sparse pca using semidefinite programming. *SIAM Review (SIREV)* 49, 3, 434–448.
- DE, A. 2012. Lower bounds in differential privacy. In *Theory of Cryptography*. Springer, 321–338.
- DING, B., WINSLETT, M., HAN, J., AND LI, Z. 2011. Differentially private data cubes: optimizing noise sources and consistency. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 217–228.
- DINUR, I. AND NISSIM, K. 2003. Revealing information while preserving privacy. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*. 202–210.
- DUCHI, J., SHALEV-SHWARTZ, S., SINGER, Y., AND CHANDRA, T. 2008. Efficient projections onto the l_1 -ball for learning in high dimensions. In *Proceedings of International Conference on Machine Learning (ICML)*. 272–279.
- DWORK, C., KENTHAPADI, K., MCSHERRY, F., MIRONOV, I., AND NAOR, M. 2006a. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Lecture Notes in Computer Science Series, vol. 4004. Springer, 486–503.
- DWORK, C., KENTHAPADI, K., MCSHERRY, F., MIRONOV, I., AND NAOR, M. 2006b. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. 486–503.
- DWORK, C., MCSHERRY, F., NISSIM, K., AND SMITH, A. 2006c. Calibrating noise to sensitivity in private data analysis. In *Proceedings of Theory of Cryptography Conference (TCC)*. 265–284.
- DWORK, C., ROTHBLUM, G. N., AND VADHAN, S. P. 2010. Boosting and differential privacy. In *Proceedings of Symposium on Foundations of Computer Science (FOCS)*. 51–60.
- DYER, M. E., FRIEZE, A. M., AND KANNAN, R. 1991. A random polynomial time algorithm for approximating the volume of convex bodies. *Journal of the Association for Computing Machinery (JACM)* 38, 1, 1–17.
- FIACCO, A. V. AND MCCORMICK, G. P. 1968. *Nonlinear programming: Sequential unconstrained minimization techniques*. John Wiley & Sons, New York.
- FRIEDMAN, A. AND SCHUSTER, A. 2010. Data mining with differential privacy. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 493–502.

- GHOSH, A., ROUGHGARDEN, T., AND SUNDARARAJAN, M. 2012. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing (SICOMP)* 41, 6, 1673–1693.
- GRIPPO, L. AND SCIANDRONE, M. 2000. On the convergence of the block nonlinear gauss-seidel method under convex constraints. *Operations Research Letters* 26, 3, 127–136.
- HARDT, M., LIGETT, K., AND MCSHERRY, F. 2012. A simple and practical algorithm for differentially private data release. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*. 2348–2356.
- HARDT, M. AND ROTH, A. 2012. Beating randomized response on incoherent matrices. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*. ACM, 1255–1268.
- HARDT, M. AND ROTHBLUM, G. N. 2010. A multiplicative weights mechanism for privacy-preserving data analysis. In *Proceedings of Symposium on Foundations of Computer Science (FOCS)*. IEEE, 61–70.
- HARDT, M. AND TALWAR, K. 2010. On the geometry of differential privacy. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*. 705–714.
- HAY, M., LI, C., MIKLAU, G., AND JENSEN, D. 2009. Accurate estimation of the degree distribution of private networks. In *Proceedings of IEEE International Conference on Data Mining (ICDM)*. 169–178.
- HAY, M., RASTOGI, V., MIKLAU, G., AND SUCIU, D. 2010. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the Very Large Data Bases Endowment (PVLDB)* 3, 1, 1021–1032.
- LI, C., HAY, M., RASTOGI, V., MIKLAU, G., AND MCGREGOR, A. 2010. Optimizing linear counting queries under differential privacy. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*. 123–134.
- LI, C. AND MIKLAU, G. 2012. An adaptive mechanism for accurate query answering under differential privacy. *Proceedings of the Very Large Data Bases Endowment (PVLDB)* 5, 6, 514–525.
- LI, C. AND MIKLAU, G. 2013. Optimal error of query sets under the differentially-private matrix mechanism. In *International Conference on Database Theory (ICDT)*. 272–283.
- LI, N., QARDAJI, W. H., SU, D., AND CAO, J. 2012. Privbasis: Frequent itemset mining with differential privacy. *Proceedings of the Very Large Data Bases Endowment (PVLDB)* 5, 11, 1340–1351.
- LI, Y. D., ZHANG, Z., WINSLETT, M., AND YANG, Y. 2011. Compressive mechanism: Utilizing sparse representation in differential privacy. In *Proceedings of ACM Workshop on Privacy in the Electronic Society (WPEC)*. 177–182.
- LIN, Z., CHEN, M., WU, L., AND MA, Y. 2010. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *Arxiv preprint arXiv:1009.5055*.
- MCSHERRY, F. AND MAHAJAN, R. 2010. Differentially-private network trace analysis. In *Proceedings of ACM SIGCOMM International Conference on Data Communication (SIGCOMM)*. 123–134.
- MCSHERRY, F. AND MIRONOV, I. 2009. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 627–636.
- MCSHERRY, F. AND TALWAR, K. 2007. Mechanism design via differential privacy. In *Proceedings of Symposium on Foundations of Computer Science (FOCS)*. 94–103.
- MOHAMMED, N., CHEN, R., FUNG, B. C. M., AND YU, P. S. 2011. Differentially private data release for data mining. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 493–501.
- NESTEROV, Y. E. 2003. *Introductory lectures on convex optimization: a basic course*. Applied Optimization Series, vol. 87. Kluwer Academic Publishers.
- NIKOLOV, A., TALWAR, K., AND ZHANG, L. 2013. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*. ACM, 351–360.
- PENG, S., YANG, Y., ZHANG, Z., WINSLETT, M., AND YU, Y. 2012. Dp-tree: indexing multi-dimensional data under differential privacy (abstract only). In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 864.
- PENG, S., YANG, Y., ZHANG, Z., WINSLETT, M., AND YU, Y. 2013. Query optimization for differentially private data management systems. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)*. 1093–1104.
- RASTOGI, V., HAY, M., MIKLAU, G., AND SUCIU, D. 2009. Relationship privacy: output perturbation for queries with joins. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*. 107–116.
- RASTOGI, V. AND NATH, S. 2010. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 735–746.
- ROTH, A. AND ROUGHGARDEN, T. 2010. Interactive privacy via the median mechanism. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*. 765–774.
- RUBINSTEIN, B. I., BARTLETT, P. L., HUANG, L., AND TAFT, N. 2012. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *Journal of Privacy and Confidentiality* 4, 1, 65–100.
- SALA, A., ZHAO, X., WILSON, C., ZHENG, H., AND ZHAO, B. Y. 2011. Sharing graphs using differentially private graph models. In *Proceedings of the ACM Internet Measurement Conference (IMC)*. 81–98.

- SREBRO, N., RENNIE, J. D., AND JAAKKOLA, T. 2004. Maximum-margin matrix factorization. In *Neural Information Processing Systems (NIPS)*. Vol. 17. 1329–1336.
- TODD, M. J. AND YILDIRIM, E. A. 2007. On khachiyan’s algorithm for the computation of minimum-volume enclosing ellipsoids. *Discrete Applied Mathematics* 155, 13, 1731–1744.
- WEN, Z., YANG, C., LIU, X., AND MARCHESINI, S. 2012a. Alternating direction methods for classical and ptychographic phase retrieval. *Inverse Problems* 28, 11, 115010.
- WEN, Z., YIN, W., AND ZHANG, Y. 2012b. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Mathematical Programming Computation* 4, 4, 333–361.
- XIAO, X., BENDER, G., HAY, M., AND GEHRKE, J. 2011. ireduct: differential privacy with reduced relative errors. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 229–240.
- XIAO, X., WANG, G., AND GEHRKE, J. 2010. Differential privacy via wavelet transforms. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)*. 225–236.
- XU, J., ZHANG, Z., XIAO, X., YANG, Y., AND YU, G. 2012. Differentially private histogram publication. In *International Conference on Data Engineering (ICDE)*. 32–43.
- XU, J., ZHANG, Z., XIAO, X., YANG, Y., YU, G., AND WINSLETT, M. 2013. Differentially private histogram publication. *The VLDB Journal* 22, 6, 797–822.
- YANG, Y., ZHANG, Z., MIKLAU, G., WINSLETT, M., AND XIAO, X. 2012. Differential privacy in data publication and analysis. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 601–606.
- YUAN, G., ZHANG, Z., WINSLETT, M., XIAO, X., YANG, Y., AND HAO, Z. 2012. Low-rank mechanism: Optimizing batch queries under differential privacy. *Proceedings of the Very Large Data Bases (VLDB) Endowment* 5, 11, 1352–1363.
- ZHANG, J., XIAO, X., YANG, Y., ZHANG, Z., AND WINSLETT, M. 2013. Privgene: differentially private model fitting using genetic algorithms. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*. ACM, 665–676.
- ZHANG, J., ZHANG, Z., XIAO, X., YANG, Y., AND WINSLETT, M. 2012. Functional mechanism: regression analysis under differential privacy. *Proceedings of the Very Large Data Bases Endowment (PVLDB)* 5, 11, 1364–1375.

A. IMPLEMENTATION OF THE APPROXIMATE MATRIX MECHANISM

Li et al. [Li et al. 2010] describes two implementations of the Matrix Mechanism, which optimizes the accuracy of a batch of linear counting queries under ϵ -differential privacy. The first directly solves the optimization program of the matrix mechanism which can be formulated as follows:

$$\min_{A \in \mathbb{R}^{r \times n}} \|A\|_{1,\infty}^2 \text{tr}(WA^\dagger A^{\dagger T} W^T) \quad (34)$$

where A^\dagger denotes the pseudo-inverse of matrix A , and $\|A\|_{1,\infty}$ is the maximum \mathcal{L}_1 norm of column vectors of A . It is shown that this problem can be formulated as a semidefinite program with rank constraint and solved by a sequence of semidefinite programs. However, it incurs high computational overhead, which is prohibitively expensive even for moderate-sized workload. The second implementation solves an approximate version of Program (34), as follows:

$$\min_{A \in \mathbb{R}^{r \times n}} \|A\|_{2,\infty}^2 \text{tr}(WA^\dagger A^{\dagger T} W^T) \quad (35)$$

where $\|A\|_{2,\infty}$ is the maximum \mathcal{L}_2 norm of column vectors of A . Under ϵ -differential privacy, Program (35) is essentially the \mathcal{L}_2 approximate of the original matrix mechanism formulation. The solution to Program (35) presented in [Li et al. 2010], however, is rather complicated, and incurs high computational costs. In the following two subsections, we describe two implementation of the approximate matrix mechanism, the *exponential smoothing mechanism (ESM)* [Yuan et al. 2012] and the adaptive mechanism (AM) [Li and Miklau 2012] for solving Program (35).

A.1. Exponential Smoothing Mechanism

In this subsection, we present a simpler and more efficient solution, referred to as the *exponential smoothing mechanism (ESM)*, based on the methodology of *exponential smoothing*. Observe that $\|A\|_{2,\infty}^2 = \max(\text{diag}(A^T A))^1$, and $(A^T A)^{-1} = (A^T A)^\dagger$ (A has full column rank). Let $M = A^T A$, we reformulate Program (35) as the following positive definite optimization problem:

$$\min_{M \in \mathbb{R}^{n \times n}} G(M) = \max(\text{diag}(M)) \text{tr}(WM^{-1}W^T) \quad \text{s.t. } M \succ 0$$

A is given by $A = \sum_i^n \sqrt{\lambda_i} v_i v_i^T$, where λ_i, v_i are the i th eigenvalue and eigenvector of M , respectively. Calculating the second term $\text{tr}(WM^{-1}W^T)$ is relatively straightforward. Since it is smooth, its gradient can be computed as $-M^{-1}W^T W M^{-1}$. However, calculating the first term $\max(\text{diag}(M))$ is harder since it is non-smooth. Fortunately, inspired by [d'Aspremont et al. 2007], we can still use a logarithmic and exponential function to approximate this term.

Approximate the maximum positive number: Since M is positive definite, $v = \text{diag}(M) > 0$. we let $\mu > 0$ be a sufficient small parameter and define:

$$f_\mu(v) = \mu \log \sum_i^n \left(\exp\left(\frac{v_i}{\mu}\right) \right) \quad (36)$$

We then have $\max(v) \leq f_\mu(v) \leq \max(v) + \mu \log n$. The gradient of the objective function in Equation (36) with respect to v can be computed as:

$$\frac{\partial f}{\partial v_i} = \frac{\exp\left(\frac{v_i - \max(v)}{\mu}\right)}{\sum_j^n \left(\exp\left(\frac{v_j - \max(v)}{\mu}\right) \right)} = \frac{\exp\left(\frac{v_i}{\mu}\right)}{\sum_j^n \left(\exp\left(\frac{v_j}{\mu}\right) \right)}, \quad \forall i \quad (37)$$

¹We use the Matlab notations in this paper. When Δ is a matrix, $\text{diag}(\Delta)$ denotes a column vector formed from the main diagonal of Δ , when Δ is a vector, $\text{diag}(\Delta)$ denotes a diagonal matrix with Δ in the main diagonal entries. Moreover, $\max(\cdot)$ retrieves the largest element of an array.

Since the second order hessian matrix of the objective function in Equation (36) can be computed as:

$$\frac{\partial^2 f}{\partial v \partial v} = \frac{\text{diag}(\exp(\frac{v}{\mu}))}{\mu \sum_j^n (\exp(\frac{v_j}{\mu}))} - \frac{\exp(\frac{v}{\mu}) \exp(\frac{v}{\mu})^T}{\mu \left(\sum_j^n (\exp(\frac{v_j}{\mu})) \right)^2} = \mathbb{S} - \mathbb{T},$$

we have the upper bound of the spectral norm of the hessian: $\|\|\frac{\partial^2 f}{\partial v \partial v}\|\|_2 = \|\|\mathbb{S} - \mathbb{T}\|\|_2 \leq \|\|\mathbb{S}\|\|_2 + \|\|\mathbb{T}\|\|_2 \leq \frac{1}{\mu} + \frac{1}{\mu} = \frac{2}{\mu}$. Therefore, the gradient of $f_\mu(v)$ is Lipschitz continuous with parameter $\omega = \frac{2}{\mu}$. If we set $\mu = \frac{\epsilon}{\log n}$, this becomes a uniform ϵ -approximation of $\max(v)$ with a Lipschitz continuous gradient with constant $\omega = \frac{2}{\mu} = \frac{2 \log n}{\epsilon}$. In our experiments, we use $\mu = \frac{0.01}{\log n}$.

To mitigate the problems with large numbers, using the property of the logarithmic and exponential functions, we can rewrite Equation (36) and Equation (37) as:

$$f_\mu(v) = \max(v) + \mu \log \left(\sum_i^n \exp \left(\frac{v_i - \max(v)}{\mu} \right) \right)$$

$$\frac{\partial f}{\partial v_i} = \left(\sum_j^n \exp \left(\frac{v_j - v_i}{\mu} \right) \right)^{-1}, \quad \forall i$$

By the chain rule of differentiation in calculus, the gradient of $G(M)$ can be computed as:

$$\frac{\partial G}{\partial M} = \text{diag} \left(\frac{\partial f}{\partial v} \right) \cdot \text{tr} (W M^{-1} W^T) + f_\mu(v) \cdot (-M^{-1} W^T W M^{-1})$$

Here $\text{diag}(\frac{\partial f}{\partial v})$ denotes a diagonal matrix with $\frac{\partial f}{\partial v} \in \mathbb{R}^n$ as the main diagonal entries. This formulation allows us to run the non-monotone spectral projected gradient descent algorithm [Birgin et al. 2000] on the cone of positive semidefiniteness. We use eigenvalue decomposition to trim the negative eigenvalues to maintain positive semidefiniteness of M , and iteratively improve the result. After the algorithm terminates, we return the final M as the optimal solution to the program.

A.2. Adaptive Mechanism

In this subsection, we briefly review the adaptive mechanism (AM) proposed in [Li and Miklau 2012], a heuristic solution for the problem in Program (35). AM considers the following optimization problem:

$$\min_{\lambda \in \mathbb{R}^n} \sum_{i=1}^n \frac{d_i^2}{\lambda_i^2}, \text{ s.t. } (Q \odot Q)(\lambda \odot \lambda) \leq \mathbf{1}_m \quad (38)$$

where Q is from the singular value decomposition of the workload matrix $W = QDP$ with $Q \in \mathbb{R}^{m \times n}$, $D \in \mathbb{R}^{n \times n}$, $P \in \mathbb{R}^{n \times n}$, and $d = \text{diag}(D) \in \mathbb{R}^n$, i.e., the diagonal values of D . Furthermore, \odot is the Hadamard (entry-wise) product, $\mathbf{1}_m$ is a column vector of all entries equal to one. AM then computes the strategy matrix A by

$$A = Q \text{diag}(\lambda) \in \mathbb{R}^{m \times n} \quad (39)$$

where $\text{diag}(\lambda)$ is a diagonal matrix with λ as its diagonal values.

The optimization problem in (40) is non-convex since it contains quadratic term both in the objective and the constraint. By changing variable to $\lambda \odot \lambda = u$, we have the following equivalent optimization problem:

$$\min_{u \in \mathbb{R}^n} \sum_{i=1}^n \frac{d_i^2}{u_i}, \text{ s.t. } (Q \odot Q)u \leq \mathbf{1}_m, \quad u \geq 0. \quad (40)$$

By introducing an auxiliary variable $v \in \mathbb{R}^n$, the optimization above can be reformulated as the following semidefinite program:

$$\min_{u \in \mathbb{R}^n, v \in \mathbb{R}^n} \sum_{i=1}^n v_i d_i^2, \text{ s.t. } (Q \odot Q) u \leq \mathbf{1}_m, \begin{bmatrix} u_i & 1 \\ 1 & v_i \end{bmatrix} \succeq 0, \forall i \in [n] \quad (41)$$

which can be solved by off-the-shelf interior-point solvers.

ALGORITHM 4: Adaptive Mechanism for Approximately Solving Problem (35)

- 1: Input: workload matrix $W \in \mathbb{R}^{m \times n}$
- 2: Compute the SVD decomposition $W = QDP$ to obtain $Q \in \mathbb{R}^{m \times n}$ and $d = \text{diag}(D) \in \mathbb{R}^n$.
- 3: Solve the semidefinite program in Equation (41) and obtain u .
- 4: Compute $A' = Q \text{diag}(\sqrt{u}) \in \mathbb{R}^{m \times n}$ and $A'' = \text{diag}(\sqrt{\max(o) \mathbf{1}_n - o}) \in \mathbb{R}^{n \times n}$ where $o_i = \|A'_i\|_2^2$, $i = 1, \dots, n$, $o \in \mathbb{R}^n$.
- 5: Output the strategy matrix A :

$$A = \begin{bmatrix} A' \\ A'' \end{bmatrix} \in \mathbb{R}^{(m+n) \times n}$$

The complete AM algorithm is summarized in Algorithm 4. Given a workload matrix W , AM automatically selects a different set of “eigen-queries” Q and use a nonnegative combination of Q to compute the strategy matrix A with respect to the workload matrix. First, in Step 2 the algorithm performs the SVD decomposition of W to derive the eigen-queries Q . Based on the eigen-queries Q , AM aims to find the optimal linear combination $\lambda (\lambda \geq 0)$ with $\lambda = \sqrt{u}$ by solving the semidefinite program in Step 3. In Step 4, the matrix A' that is constructed is a candidate strategy but may have one or more columns whose norm is less than the sensitivity. In this case, AM adds queries or completes columns in order to further reduce the expected error without raising the sensitivity. Essentially AM searches over a reduced subspace of A . Hence, the candidate strategy matrix A' solved from the optimization problem in (35) does not guarantee to be the optimal strategy since it is limited to a weighted nonnegative combination of the fixed eigen-queries Q in Equation (39).

B. ASYMPTOTIC ERROR BOUNDS FOR LRM

B.1. LRM Error Bounds under ϵ -Differential Privacy

In this subsection, we prove the lower bound and upper bound of the error incurred by the optimal workload decomposition solved from Program (9), and analyze the gap between the two bounds. First, we establish an error upper bound for LRM in the following lemma.

LEMMA B.1. Error upper bound under ϵ -differential privacy. *Given a workload matrix W of rank s with singular values $\{\lambda_1, \dots, \lambda_s\}$, an upper bound of the expected squared error of $M_{LRM, \epsilon}(Q, D)$ w.r.t. the optimal decomposition $W = B^* L^*$ is $2 \sum_{k=1}^s \lambda_k^2 / \epsilon^2$.*

PROOF. Consider the naive method NOD, which can be considered as a special case of LRM by setting $B = W$ and $L = I$ (i.e., identity matrix). Clearly, $\Delta(L) = 1$. According to Lemma 4.2, the expected squared error of this decomposition is:

$$2\Phi(B)\Delta(L)^2/\epsilon^2 = 2\|W\|_F^2/\epsilon^2 = 2 \sum_{k=1}^s \lambda_k^2/\epsilon^2$$

We reach the conclusion of the lemma. \square

Next we derive a lower bound on the squared error for linear counting queries under ϵ -differential privacy, using geometric analysis under orthogonal projection [Hardt and Talwar 2010]. To do this, we first present the following lemma, which is used later in our geometric analysis.

LEMMA B.2. *For all orthogonal $V \in \mathbb{R}^{s \times n}$, we have the following inequality:*

$$\text{Vol}(VB_1^n) \geq \text{Vol}(B_2^s) \cdot n^{-\frac{s}{2}}$$

where $\text{Vol}(B_2^s)$ denotes the volume of unit Euclidean ball, and $\text{Vol}(VB_1^n)$ denotes the volume of unit ball of the \mathcal{L}_1 norm on \mathbb{R}^n after the orthogonal transformation under V .

PROOF. By Cauchy-Schwarz inequality we have $\|x\|_1 \leq \sqrt{n}\|x\|$ for all $x \in \mathbb{R}^n$, therefore, the n -dimensional ℓ_1 ball contains an ℓ_2 ball of radius $n^{-\frac{1}{2}}$, i.e. $B_1^n \supseteq n^{-\frac{1}{2}}B_2^n$. Given an orthogonal transformation V , we obtain $VB_1^n \supseteq n^{-\frac{1}{2}}VB_2^n$. Moreover, because the orthogonal projection of a Euclidean ball is a lower-dimensional Euclidean ball of the same radius, it holds that $n^{-\frac{1}{2}}VB_2^n = n^{-\frac{1}{2}}B_2^s$. Therefore, the volume of VB_1^n is bounded from below by:

$$\begin{aligned} \text{Vol}(VB_1^n) &\geq \text{Vol}(n^{-\frac{1}{2}}B_2^s) \\ &= \text{Vol}(B_2^s) \cdot n^{-\frac{s}{2}}. \end{aligned}$$

□

We are now ready to prove the error lower bound of LRM.

LEMMA B.3. **Error Lower Bound under ϵ -differential privacy.** *Given a workload matrix W of rank s with singular values $\{\lambda_1, \dots, \lambda_s\}$, the expected squared error of any ϵ -differential privacy mechanism is at least*

$$\Omega \left(\frac{s^4}{n} \left(\frac{2^s}{s!} \prod_{k=1}^s \lambda_k \right)^{2/s} / \epsilon^2 \right)$$

PROOF. Corollary 3.4 in [Hardt and Talwar 2010] proves that any ϵ -differential privacy mechanism for linear counting queries incurs expected squared error no less than:²

$$\Omega \left(k^3 (\text{Vol}(PW B_1^n))^{2/k} / \epsilon^2 \right)$$

In the formula above, B_1^n is the \mathcal{L}_1 -unit ball. $\text{Vol}(PW B_1^n)$ is the volume of the unit ball after the linear transformation PW , in which P is any orthogonal linear transformation matrix from $\mathbb{R}^n \mapsto \mathbb{R}^s$. To prove the lemma, we construct an orthogonal transformation $P = U^T$, where U is obtained from the SVD decomposition of W ($W = U\Sigma V$). According to properties of SVD decomposition, $U^T U$ and $V V^T$ are identity matrices. Thus, we have $\text{Vol}(PW B_1^n) = \text{Vol}(P U V V^T \Sigma V B_1^n) = \text{Vol}(V(V^T \Sigma V) B_1^n) = \text{Vol}(V B_1^n) \prod_{k=1}^s \lambda_k$. The last equality holds due to Lemma 7.5 in [Hardt and Talwar 2010]. Consider the the convex body VB_1^n . By Lemma B.2, it has a lower bound $\text{Vol}(B_2^s) \cdot (n^{-\frac{s}{2}})$. Note that $\text{Vol}(B_2^s)$ can be computed using the Gamma function [Ball 1997]: $\frac{\pi^{s/2}}{\Gamma(1+s/2)}$. Using the Stirling's formula, we know that $\Gamma(1+s/2)$ is roughly $\sqrt{2\pi} e^{-s/2} (s/2)^{s/2+1/2}$, so that $\text{Vol}(B_2^s)$ is roughly $(\frac{2\pi e}{s})^{\frac{s}{2}}$. Therefore, the lower bound can be computed as: $\Omega \left(\frac{s^4}{n} \left(\frac{2^s}{s!} \prod_{k=1}^s \lambda_k \right)^{2/s} / \epsilon^2 \right)$. We thus reach the conclusion of the lemma. □

Next we compare the error upper and lower bounds. The analysis involves a matrix-theory concept called the *generalized condition number*.

²[Hardt and Talwar 2010] used absolute errors, from which which we derived the squared errors.

Definition B.4. Generalized condition number. Given a workload matrix W , the generalized condition number $\kappa(W)$ of W defined as the product of the spectral norm of W and that of its pseudo-inverse or equivalently, the ratio between the largest singular value of W to the *nonzero* smallest [Chen and Dongarra 2005; Beltrán 2011].

$$\kappa(W) \triangleq \|W\|_2 \cdot \|W^\dagger\|_2 = \frac{\lambda_1}{\lambda_s}$$

Note that we always have $\kappa(W) \geq 1$.

THEOREM B.5. *When $s > 5$, the gap between the upper and lower bounds of the error incurred by mechanism $M_{LRM,\epsilon}(Q, D)$ with the optimal decomposition $W = B^*L^*$ is $\mathcal{O}\left((\kappa(W))^2 \frac{n}{s}\right)$.*

PROOF. The theorem is established by comparing the upper and lower bounds in Lemmas B.1 and B.3, as follows.

$$\begin{aligned} \frac{2 \sum_{k=1}^s \lambda_k^2 / \epsilon^2}{\frac{s^4}{n} \left(\frac{2^s}{s!} \prod_{k=1}^s \lambda_k\right)^{2/s} / \epsilon^2} &\leq \frac{2 \sum_{k=1}^s \lambda_1^2}{\frac{s^4}{n} \left(\frac{2^s}{s!} \prod_{k=1}^s \lambda_s\right)^{2/s}} \\ &\leq \frac{2ns\lambda_1^2}{\left(\frac{2^s}{s!}\right)^{2/s} \lambda_s^2 s^4} \\ &= \frac{2n\kappa(W)^2}{\left(\frac{2^s}{s!}\right)^{2/s} s^3} \\ &\leq \frac{1}{8} \kappa(W)^2 \frac{n}{s} \end{aligned}$$

The last inequality holds due to the fact that $s! < \left(\frac{s}{2}\right)^s$ when $s > 5$. Note that all the inequalities above are tight, and the equalities hold when $\kappa(W) = 1$, i.e. $\lambda_1 = \lambda_2 = \dots = \lambda_s$. \square

From the theorem above, we draw the following interesting observations. (i) When the rank of the matrix is low (i.e., s is small) and the batch queries are highly correlated ($\kappa(W) \gg 1$), then the ratio of the upper bound to the lower bound is large, meaning that LRM can potentially achieve lower error than NOD. (ii) Conversely, when the rank of the matrix is full rank ($s \rightarrow n$ and $n \leq m$) and the batch queries are almost random or independent ($\kappa(W) \rightarrow 1$), then the achievable error rate of LRM converges to the upper error bound obtained by NOD. Therefore, in this situation, NOD might be good enough and no sophisticated algorithm is needed, which is validated by the experimental results in Section 7.3). These results are consistent with the work of [Ghosh et al. 2012], who show that Laplace mechanism is optimal in a strong sense when answering a single linear query.

B.2. LRM Error Bounds under (ϵ, δ) -Differential Privacy

We first derive an upper bound for the error of LRM. Unlike the case of ϵ -differential privacy, we have a tighter error upper bound than that obtained by naive methods. We introduce the concept of ρ -coherence of a matrix, which is similar to μ -coherence [Candès and Recht 2009] and C -coherence [Hardt and Roth 2012] of a matrix in the low-rank optimization literature.

Definition B.6. ρ -coherence of a matrix. Given a matrix W with its SVD decomposition that $W = U\Sigma V$, where $U \in \mathbb{R}^{m \times s}$, $\Sigma \in \mathbb{R}^{s \times s}$, $V \in \mathbb{R}^{s \times n}$. We say the matrix W is ρ -coherent if

$$\rho(W) = \max_i \|V_i\|_2, \quad i = 1, \dots, n$$

where V_i is the i -th column of V . Note that we have $0 < \rho(W) \leq 1$.

LEMMA B.7. Error Upper Bound under (ϵ, δ) -differential privacy. Given a workload matrix W of rank s with singular values $\{\lambda_1, \dots, \lambda_s\}$, an upper bound of the expected squared error of $M_{LRM,(\epsilon,\delta)}(Q, D)$ w.r.t. the optimal decomposition $W = B^*L^*$ is $(\rho(W))^2 \sum_{k=1}^s \lambda_k^2 / h(\epsilon, \delta)^2$.

PROOF. To prove the lemma, we perform SVD decomposition of W , obtaining $W = U\Sigma V$. Then, we build a decomposition $B = \rho(W)U\Sigma$ and $L = \frac{1}{\rho(W)}V$. This is a valid decomposition of W , because $BL = \rho(W)U\Sigma \frac{1}{\rho(W)}V = U\Sigma V = W$.

Next we prove that $\Delta(L) = 1$. According to properties of the SVD transformation, column vectors in V are orthogonal vectors; hence, for every column V_j in V , we have $\|V_j\|_2 \leq \rho(W)$. Therefore, $\Theta(L) = \max_j (\sum_i L_{ij}^2)^{1/2} = \max_j \frac{1}{\rho(W)} \|V_j\|_2 = 1$.

The expected squared error of this decomposition is then bounded by:

$$\begin{aligned} \Phi(B) &= \text{tr}(B^T B) / h(\epsilon, \delta)^2 \\ &= \text{tr}((\rho(W)U\Sigma)^T (\rho(W)U\Sigma)) / h(\epsilon, \delta)^2 \\ &= \rho(W)^2 \text{tr}(\Sigma^T U^T U \Sigma) / h(\epsilon, \delta)^2 \\ &= \rho(W)^2 \sum_{k=1}^s \lambda_k^2 / h(\epsilon, \delta)^2 \end{aligned}$$

We thus reach the conclusion of the lemma.

□

Note that since $\rho(W) \leq 1$, the above error bound is no worse than the error obtained by NOD. Meanwhile, the proof essentially describes another simple solution whose accuracy is no worse than NOD.

We now focus on the error lower bound of LRM under (ϵ, δ) -differential privacy. This has already been studied in [Li and Miklau 2013], and we summarize their results with our notations in the following lemma.

LEMMA B.8. Error Lower Bound under (ϵ, δ) -differential privacy [Li and Miklau 2013]. Given a workload matrix W of rank s with singular values $\{\lambda_1, \dots, \lambda_s\}$, the expected squared error of $M_{LRM,(\epsilon,\delta)}(Q, D)$ w.r.t. the optimal decomposition $W = B^*L^*$ is at least

$$\frac{1}{nh(\epsilon, \delta)^2} \left(\sum_{i=1}^s \lambda_i \right)^2$$

The proof of the above result in [Li and Miklau 2013] is rather complicated. In the following we provide a simple proof.

PROOF.

$$\begin{aligned} \min_{\substack{W=BL, \\ \forall j \sum_i L_{ij}^2 \leq 1}} \frac{1}{h(\epsilon, \delta)^2} \|B\|_F^2 &\geq \frac{1}{nh(\epsilon, \delta)^2} \min_{W=BL} \|L\|_F^2 \cdot \|B\|_F^2 \\ &= \frac{1}{nh(\epsilon, \delta)^2} (\|W\|_*)^2 \\ &= \frac{1}{nh(\epsilon, \delta)^2} \left(\sum_{i=1}^s \lambda_i \right)^2 \end{aligned}$$

The first inequality is due to $\sum_j^n (\sum_i^r L_{ij}^2) \leq n$. Note that this inequality above is tight, and the equality holds when every column of L lies on the surface of the unit ball. The first equality is due

to the variational formulation of nuclear norm (see, e.g., [Srebro et al. 2004]) that

$$\|W\|_* = \min_{B,L} \|L\|_F \cdot \|B\|_F, \quad s.t. \quad W = BL.$$

We thus reach the conclusion of the lemma. \square

We next compare the error upper bound and the error lower bound for LRM under (ϵ, δ) -differential privacy.

THEOREM B.9. *The ratio between the error upper and lower bounds of mechanism $M_{LRM,(\epsilon,\delta)}(Q, D)$ with the optimal decomposition $W = B^*L^*$ is bounded by $\mathcal{O}\left((\kappa(W))^2 \frac{n}{s}\right)$.*

PROOF.

We compare the upper and lower bounds in B.7 and B.8, as follows.

$$\begin{aligned} \frac{\rho(W)^2 \sum_{k=1}^s \lambda_k^2 / h(\epsilon, \delta)^2}{\frac{1}{n} (\sum_{i=1}^s \lambda_i)^2 / h(\epsilon, \delta)^2} &= \frac{\rho(W)^2 \sum_{k=1}^s \lambda_k^2}{\frac{1}{n} (\sum_{i=1}^s \lambda_i)^2} \\ &\leq \frac{s \lambda_1^2 \rho(W)^2}{\frac{1}{n} \lambda_s^2 s^2} \\ &= (\kappa(W) \rho(W))^2 \frac{n}{s} \end{aligned}$$

We thus reach the conclusion of the theorem. \square

The above theorem leads to similar conclusions as in the case of ϵ -differential privacy, except that here we compare LRM with an improved version of NOD described in the proof of Lemma B.7. Meanwhile, the above ratio also involves an additional parameter ρ , i.e., the coherence number of the workload matrix.