

# Principled Evaluation of Differentially Private Algorithms using DPBench

Michael Hay\*, Ashwin Machanavajjhala\*\*, Gerome Miklau†, Yan Chen\*\*, Dan Zhang†

\* Colgate University  
Department of Computer  
Science  
mhay@colgate.edu

\*\* Duke University  
Department of Computer  
Science  
{ashwin,yanchen}@cs.duke.edu

† University of Massachusetts  
Amherst  
School of Computer Science  
{miklau,dzhang}@cs.umass.edu

## ABSTRACT

Differential privacy has become the dominant standard in the research community for strong privacy protection. There has been a flood of research into query answering algorithms that meet this standard. Algorithms are becoming increasingly complex, and in particular, the performance of many emerging algorithms is *data dependent*, meaning the distribution of the noise added to query answers may change depending on the input data. Theoretical analysis typically only considers the worst case, making empirical study of average case performance increasingly important.

In this paper we propose a set of evaluation principles which we argue are essential for sound evaluation. Based on these principles we propose DPBENCH, a novel evaluation framework for standardized evaluation of privacy algorithms. We then apply our benchmark to evaluate algorithms for answering 1- and 2-dimensional range queries. The result is a thorough empirical study of 15 published algorithms on a total of 27 datasets that offers new insights into algorithm behavior—in particular the influence of dataset scale and shape—and a more complete characterization of the state of the art. Our methodology is able to resolve inconsistencies in prior empirical studies and place algorithm performance in context through comparison to simple baselines. Finally, we pose open research questions which we hope will guide future algorithm design.

## 1. INTRODUCTION

Privacy is a major obstacle to deriving important insights from collections of sensitive records donated by individuals. Differential privacy [5–8] has emerged as an important standard for protection of individuals’ sensitive information. Informally, differential privacy is a property of the analysis algorithm which guarantees that the output the analyst receives is statistically indistinguishable (governed by a privacy parameter  $\epsilon$ ) from the output the analyst would have received if any one individual had opted out of the collection. Its general acceptance by researchers has led to a flood of work across the database, data mining, theory, machine learning, programming languages, security, and statistics communities.

Most differentially private algorithms work by introducing noise into query answers. Finding algorithms that satisfy  $\epsilon$ -differential

privacy and introduce the least possible error for a given analysis task is a major ongoing challenge, in both research and practice. Standard techniques for satisfying differential privacy that are broadly-applicable (e.g. the Laplace and exponential mechanisms [8]) often offer sub-optimal error rates. Much recent work that deems differential privacy impractical for real world data (e.g., [13]) use only these standard techniques.

Many new differentially private algorithms have been proposed to address these limitations and reduce achievable error rates. Take the example task of privately answering 1- and 2-dimensional range queries on a dataset (the primary focus of this paper). Proposed techniques for this task include answering range queries using noisy hierarchies of equi-width histograms [4, 11, 16, 22], noisy counts on a coarsened domain [15, 21, 26, 27, 29], or by reconstructing perturbed wavelet [25] or Fourier [1] coefficients, or based on a synthetic dataset built using multiplicative weights updates [10]. These recent innovations reduce error at a fixed privacy level  $\epsilon$  by many orders of magnitude for certain datasets, and can have a large impact on the success of practical differentially private applications.

However, the current state-of-the-art poses a new challenge of algorithm selection. Consider a data owner (say from the US Census Bureau) who would like to use differentially private algorithms to release a 1- or 2- dimensional histogram over their data. She sees a wide variety of algorithms in the published literature, each demonstrating settings or contexts in which they have advantageous theoretical and empirical properties. Unlike in other fields (e.g. data mining), the data owner can not run all the algorithms on her dataset and choose the algorithm that incurs the least error – this violates differential privacy, as the choice of the algorithm would leak information about the input dataset. Hence, the data owner must make this choice using prior theoretical and empirical analyses of these algorithms, and faces the following problems:

1. *Gaps in Empirical Evaluations:* As algorithms become progressively more complex, their error rates are harder to analyze theoretically, underscoring the importance of good empirical evaluations. For a number of reasons, including lack of benchmark datasets, space constraints in publications, and the chronology of proposed algorithms, existing empirical evaluations do not comprehensively evaluate all known algorithms leaving gaps (and even inconsistencies; see Section 3.2) in our knowledge about algorithm performance.
2. *Understanding Data-Dependence:* A number of recent algorithms are *data dependent*; i.e., their error is sensitive to properties of the input. Thus, a data dependent algorithm  $A$  may have lower error than another algorithm  $B$  on one dataset, but the reverse may be true on another dataset. While a few prior empirical evaluations evaluate algorithms on diverse datasets, there

Permission to make digital or hard copies of all or part 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 bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](http://permissions.acm.org).

SIGMOD’16, June 26–July 01, 2016, San Francisco, CA, USA

© 2016 ACM. ISBN 978-1-4503-3531-7/16/06...\$15.00

DOI: <http://dx.doi.org/10.1145/2882903.2882931>

is little guidance for the data owner on how the error would extrapolate to a new dataset (e.g., one with a greater number of tuples) or a new experimental setting (e.g., a smaller  $\epsilon$ ).

3. *Choosing Values for Free Parameters:* Algorithms are often associated with free parameters (other than the privacy parameter  $\epsilon$ ), but their effect on error is incompletely quantified. Published research provides little guidance to the data owner on how to set these parameters, and in many cases the default values are sub-optimal.
4. *Unsound Utility Comparisons:* Even when empirical analyses are comprehensive, the results may not be useful to a practitioner. For instance, the error of a differentially private algorithm is a random variable. However, most empirical analyses only report the mean error and not the variability in error. Moreover, algorithm error is often not compared with that of simple baselines like the Laplace mechanism, which are the first algorithms a practitioner would attempt applying.

Given these problems, even as research progresses, the practitioner is lost and incapable of deploying the right algorithm. Moreover, researchers are likely to propose new algorithms that improve performance in narrow contexts. In this paper, we attempt to remedy the above problems with a rigorous framework for empirical evaluation, and we shed light on the algorithm selection problem for 1- and 2-dimensional range query answering. We make the following novel contributions:

1. We propose a set of principles that any differentially private evaluation must satisfy (Section 4). Based on these principles we develop DPBENCH, a novel methodology for evaluating differentially private algorithms (Section 5). Experiments designed using DPBENCH help tease out dependence of algorithm error on specific data characteristics like the size of the domain, the dataset’s number of tuples (or scale) and its empirical distribution (or shape). DPBENCH provides an algorithm to automatically tune free parameters of algorithms, and ensures fair comparisons between algorithms. Finally, by reporting both the mean and variation of error as well as comparing algorithms to baselines the results of DPBENCH experiments help the practitioner get a better handle on the algorithm selection problem.
2. Using DPBENCH, we present a comprehensive empirical study of 15 published algorithms for releasing 1- or 2-dimensional range queries on a total of 27 datasets (Section 6). We evaluate algorithm error on 7,920 experimental settings. Our study presents the following novel insights into algorithm error (Section 7):
  - (a) *Scale and Data Dependence:* The error incurred by recently proposed data dependent algorithms is heavily influenced by scale (number of tuples). At smaller scales, the best data-dependent algorithms can beat simpler data independent algorithms by an order of magnitude. However, at large scales, many data dependent algorithms start performing worse than data independent algorithms.
  - (b) *Baselines:* A majority of the recent algorithms do not even consistently beat the Laplace mechanism, especially at medium and larger scales in both the 1- and 2-D cases.
  - (c) *Understanding Inconsistencies:* Our results also help us resolve and explain inconsistencies from prior work. For instance, a technique that uses multiplicative weights [10] was shown to outperform a large class of data independent mechanisms (like [25]), but in a subsequent paper [15], the

reverse was shown to be true. Our results explain this apparent inconsistency by identifying that the former conclusion was drawn using a dataset with small scale while the latter conclusions were made on datasets with larger scales.

3. While the DPBENCH algorithm for tuning free parameters is straightforward, we show a 13 $\times$  improvement in error for certain algorithms like the multiplicative weights method [10] over the original implementations with default parameters.
4. We formalize two important theoretical properties, *scale-epsilon exchangeability* and *consistency*, that can help extrapolate results from an empirical study to other experimental setting not considered by the study. For an algorithm that is *scale-epsilon exchangeable*, increasing the scale of the input dataset and increasing epsilon have equivalent effects on the error. Thus, an algorithm’s error is roughly the same for all scale and  $\epsilon$  pairs with the same product. We prove that most algorithms we consider in this paper satisfy this property. An algorithm that is *consistent* has error that tends to 0 as the privacy parameter  $\epsilon$  tends to  $\infty$ . That is, inconsistent algorithms return answers that are biased even in the absence of privacy constraints. While all the data independent algorithms considered in the paper are consistent, we show that some data dependent algorithms are not consistent, and hence a practitioner must be wary about using such algorithms.

While our analysis is restricted to 1- and 2-dimensional range queries, we believe our results will be useful beyond this context as the evaluation principles will extend to evaluation of differentially private algorithms for other tasks. We conclude the paper with comparisons to published results, lessons for practitioners, and a discussion of open questions in Section 8.

## 2. PRELIMINARIES

In this section we review basic privacy definitions and introduce notation for databases and queries.

### 2.1 Differential Privacy

In this paper we are concerned with algorithms that run on a private database and publish their output. To ensure the released data does not violate privacy, we require the algorithms to obey the standard of differential privacy.

Let  $I$  be a database instance consisting of a single relation. Let  $nbrs(I)$  denote the set of databases differing from  $I$  in at most one record; i.e., if  $I' \in nbrs(I)$ , then  $|(I - I') \cup (I' - I)| = 1$ .

**DEFINITION 1 (DIFFERENTIAL PRIVACY [7]).** *A randomized algorithm  $\mathcal{A}$  is  $\epsilon$ -differentially private if for any instance  $I$ , any  $I' \in nbrs(I)$ , and any subset of outputs  $S \subseteq Range(\mathcal{A})$ ,*

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

For an individual whose data is represented by a record in  $I$ , differential privacy offers an appealing guarantee. It says that including this individual’s record cannot significantly affect the output: it can only make some outputs slightly more (or less) likely – where “slightly” is defined as at most a factor of  $e^\epsilon$ . If an adversary infers something about the individual based on the output, then the same inference would also be likely to occur *even if the individual’s data had been removed from the database* prior to running the algorithm.

Many of the algorithms considered in this paper are composed of multiple subroutines, each taking the private data as input. Provided each subroutine achieves differential privacy, the whole algorithm is differentially private. More precisely, the sequential execution of  $k$  algorithms  $\mathcal{A}_1, \dots, \mathcal{A}_k$ , each satisfying  $\epsilon_i$ -differential

privacy, results in an algorithm that is  $\epsilon$ -differentially private for  $\epsilon = \sum_i \epsilon_i$  [19]. Hence, we think of  $\epsilon$  as representing an algorithm’s *privacy budget* which can be allocated across its subroutines.

A commonly used subroutine is the Laplace mechanism, a general purpose algorithm for computing numerical functions on the private data. It achieves privacy by adding noise to the function’s output. We use  $\text{Laplace}(\sigma)$  to denote the Laplace probability distribution with mean 0 and scale  $\sigma$ .

**DEFINITION 2 (LAPLACE MECHANISM [7]).** *Let  $f(I)$  denote a function on  $I$  that outputs a vector in  $\mathbb{R}^d$ . The Laplace mechanism  $\mathcal{L}$  is defined as  $\mathcal{L}(I) = f(I) + \mathbf{z}$ , where  $\mathbf{z}$  is a  $d$ -length vector of random variables such that  $z_i \sim \text{Laplace}(\Delta f/\epsilon)$ .*

*The constant  $\Delta f$  is called the sensitivity of  $f$  and is the maximum difference in  $f$  between any two databases that differ only by a single record,  $\Delta f = \max_{I, I' \in \text{nbrs}(I)} \|f(I) - f(I')\|_1$ .*

The Laplace mechanism can be used to provide noisy counts of records satisfying arbitrary predicates. For example, suppose  $I$  contains medical records and  $f$  reports two counts: the number of male patients with heart disease and the number of female patients with heart disease. The sensitivity of  $f$  is 1: given any database instance  $I$ , adding one record to it (to produce neighboring instance  $I'$ ), could cause at most one of the two counts to increase by exactly 1. Thus, the Laplace mechanism would add random noise from  $\text{Laplace}(1/\epsilon)$  to each count and release the noisy counts.

## 2.2 Data Model and Task

The database  $I$  is an instance of a single-relation schema  $R(\mathbb{A})$ , with attributes  $\mathbb{A} = \{A_1, A_2, \dots, A_\ell\}$ . Each attribute is discrete, having an ordered domain (continuous attributes can be suitably discretized). We are interested in answering range queries over this data; range queries support a wide range of data analysis tasks including histograms, marginals, data cubes, etc.

We consider the following task. The analyst specifies a subset of target attributes, denoted  $\mathbb{B} \subseteq \mathbb{A}$ , and  $\mathbf{W}$ , a set of multi-dimensional range queries over  $\mathbb{B}$ . We call  $\mathbf{W}$  the *workload*. For example, suppose the database  $I$  contains records from the US Census describing various demographic characteristics of US citizens. The analyst might specify  $\mathbb{B} = \{\text{age}, \text{salary}\}$  and a set  $\mathbf{W}$  where each query is of the form,

```
select count(*) from R
where  $a_{low} \leq \text{age} \leq a_{high}$  and  $s_{low} \leq \text{salary} \leq s_{high}$ 
```

with different values for  $a_{low}$ ,  $a_{high}$ ,  $s_{low}$ ,  $s_{high}$ . We restrict our attention to the setting where the dimensionality,  $k = |\mathbb{B}|$ , is small (our experiments report on  $k \in \{1, 2\}$ ). All the differentially private algorithms considered in this paper attempt to answer the range queries in  $\mathbf{W}$  on the private database  $I$  while incurring as little error as possible.

In this paper, we will often represent the database as a multi-dimensional array  $\mathbf{x}$  of counts. For  $\mathbb{B} = \{B_1, \dots, B_k\}$ , let  $n_j$  denote the domain size of  $B_j$  for  $j \in [1, k]$ . Then  $\mathbf{x}$  has  $(n_1 \times n_2 \times \dots \times n_k)$  cells and the count in the  $(i_1, i_2, \dots, i_k)^{th}$  cell is

```
select count(*) from R
where  $B_1 = i_1$  and  $B_2 = i_2$  and  $\dots B_k = i_k$ 
```

To compute the answer to a query in  $\mathbf{W}$ , one can simply sum the corresponding entries in  $\mathbf{x}$ . Because they are range queries, the corresponding entries form a (hyper-)rectangle in  $\mathbf{x}$ .

**Example:** Suppose  $\mathbb{B}$  has the attributes age and salary (in tens of thousands) with domains  $[1, 100]$  and  $[1, 50]$  respectively. Then  $\mathbf{x}$  is a  $100 \times 50$  matrix. The  $(25, 10)^{th}$  entry is the number of tuples with age 25 and salary \$100,000.

Algorithm	Properties					Analysis	
	H	P	Dimension	Parameters	Side info	Consistent	Scale- $\epsilon$ Exch.
<i>Data-independent</i>							
IDENTITY [7]			Multi-D	–		yes	yes
PRIVELET [25]	X		Multi-D	–		yes	yes
H [11]	X		1D	$b = 2$		yes	yes
H <sub>b</sub> [22]	X		Multi-D	–		yes	yes
GREEDY H [15]	X		1D, 2D	$b = 2$		yes	yes
<i>Data-dependent</i>							
UNIFORM		~	Multi-D	–		no	yes
MWEM [10]			Multi-D	$T$	scale	no	yes
MWEM*			Multi-D	–		no	yes
AHP [29]		X	Multi-D	$\rho, \eta$		yes	yes
AHP*		X	Multi-D	–		yes	yes
DPCUBE [26]	~	X	Multi-D	$\rho = .5, n_p = 10$		yes	yes
DAWA [15]	X	X	1D, 2D	$\rho = .25, b = 2$		yes	yes
QUADTREE [4]	X	X	2D	$c=10$		no*	yes
UGRID [21]		X	2D	$c = 10$	scale	yes	yes
AGRID [21]	~	X	2D	$c = 10, c_2 = 5, \rho = .5$	scale	yes	yes
PHP [1]		X	1D	$\rho = .5$		no	yes
EFPA [1]			1D	–		yes	yes
SF [27]		X	1D	$\rho, k, F$	scale	yes*	no

Table 1: Algorithms evaluated in benchmark. Property column H indicates hierarchical algorithms and P indicates partitioning. Parameters without assignments are ones that remain free. Side information is discussed in Section 4.2. Analysis columns are discussed in Section 5.5 and Section 7.4. Algorithm variants MWEM\* and AHP\* are explained in Section 6.4.

We identify three key properties of  $\mathbf{x}$ , each of which significantly impacts the behavior of privacy algorithms. The first is the *domain size*,  $n$ , which is equivalently the number of cells in  $\mathbf{x}$  (i.e.,  $n = n_1 \times \dots \times n_k$ ). The second is the *scale* of the dataset, which is the total number of tuples, or the sum of the counts in  $\mathbf{x}$ , which we write as  $\|\mathbf{x}\|_1$ . Finally, the *shape* of a dataset is denoted as  $\mathbf{p}$  where  $\mathbf{p} = \mathbf{x} / \|\mathbf{x}\|_1 = [p_1, \dots, p_n]$  is a non-negative vector that sums to 1. The shape captures how the data is distributed over the domain and is independent of scale.

## 3. ALGORITHMS & PRIOR RESULTS

### 3.1 Overview of Algorithm Strategies

The algorithms evaluated in this paper are listed in Table 1. For each algorithm, the table identifies the dataset dimensionality it supports as well as other key properties (discussed further below). In addition, it identifies algorithm-specific parameters and the possible use of “side information” (discussed in Section 4). The table also summarizes our theoretical analysis, which is described in detail later (Sections 5.5 and 7.4). Descriptions of individual algorithms are provided in Appendix B.

In this section, we categorize algorithms as either data-independent or data-dependent, and further highlight some key strategies employed, such as the use of hierarchical aggregations and partitioning. In addition, we also illustrate how algorithm behavior is affected by properties of the input including dataset shape, scale, and domain size.

First, we describe a simple baseline strategy: release  $\mathbf{x}$  after adding independent random noise to each count in  $\mathbf{x}$ . To ensure differential privacy, the noise distribution is calibrated according to Definition 2, and each cell receives independent Laplace noise with a scale of  $1/\epsilon$ . The limitation with this simple strategy is that when answering range queries, the variance in the answer increases lin-

early with the number of cells that fall within the range. For large ranges, the error becomes intolerably high. Thus, the performance of this approach depends critically on the *domain size*.

**Hierarchical aggregation:** To mitigate the noise accumulation, several approaches not only obtain noisy counts for the individual cells in  $\mathbf{x}$ , but also obtain noisy estimates for the total count in hierarchically grouped subsets of cells. Because each record is now being counted multiple times, the noise must be proportionally increased. However, it has been shown in the 1D case that the increase is only logarithmic in the domain size and, further, any range query now requires summing only a logarithmic, rather than linear, number of noisy counts [11, 24]. For higher dimensions, the relative benefit of hierarchical aggregations diminishes [22]. Algorithms employing hierarchical aggregations include PRIVELET [25], H [11],  $H_b$  [22], GREEDY H [15], DAWA [15], AGRID [21], DPCUBE [26], QUADTREE [4], HYBRIDTREE [4].

For many of the algorithms mentioned above, the noise added does not depend on the data. Thus, the performance is the same on all datasets of a given domain size. Following [15], an algorithm whose error rate is the same for all possible datasets on a given domain is characterized as *data independent*. Table 1 indicates which algorithms are data-independent.

It can be shown that all of the data independent algorithms studied here are instances of the matrix mechanism [16, 18], a generic framework that computes linear combinations of cell counts, adds noise, and then reconstructs estimates for the individual cells through linear transformations. The algorithms differ in the choice of linear combination and thus experience different error rates on a given workload  $\mathbf{W}$ . (Computing the optimal linear combination is computationally infeasible.)

Thus, for data-independent algorithms, the only property of the input that affects performance is domain size. We next describe strategies of data-dependent algorithms, whose performance is affected by dataset shape and scale.

**Partitioning:** One class of data-dependent algorithm partitions the data, reducing the error due to noise by only computing noisy aggregations of cell groups. An example of such an approach is an equi-width histogram: the domain is partitioned into disjoint intervals (called “buckets”) of equal size and a noisy count for each bucket is obtained. To approximate the count for a cell within a bucket, the standard *assumption of uniformity* is employed. The success of such a strategy depends critically on the *shape* of the dataset: If the distribution of cell counts within a bucket are nearly uniform, the noise is effectively reduced because each cell only receives a fraction of the noise; on the other hand, a non-uniform distribution implies high error due to approximating each cell by the bucket average. Algorithms that partition include AGRID, HYBRIDTREE, QUADTREE, DPCUBE, DAWA, as well as PHP [1], AHP [29], SF [27], UGRID [21].

Equi-width partitions are used by QUADTREE, UGRID, and AGRID whereas other algorithms select the partition adaptively based on the characteristics of the data. This is non-trivial because one must prove that the adaptive component of the algorithm satisfies differential privacy (and thus does not adapt too specifically to the input). Other algorithms, such as MWEM and EFPA [1], adapt to the data as well, but use a different strategy than partitioning.

While one might expect that the performance of data-dependent algorithms is affected by dataset shape, a novel contribution of this paper is to show that it is also affected by *scale*. The intuition for this can be seen by considering equi-width histograms. Holding shape fixed, as scale increases, any deviations from a uniform distribution become magnified. Thus, the relative benefit of partitioning cells into buckets diminishes. While data adaptive approaches

could, in principle, adjust with increasing scale, our findings and theoretical analysis show that this is not always the case.

**Baselines:** IDENTITY is a data-independent algorithm described earlier: it adds noise to each cell count in  $\mathbf{x}$ , and is equivalent to applying the Laplace mechanism on the function that transforms  $I$  into  $\mathbf{x}$ . UNIFORM is the second baseline, which uses its privacy budget to estimate the number of tuples (scale) and then produces a data-dependent estimate of the dataset by assuming uniformity. It is equivalent to an equi-width histogram that contains a single bucket as wide as the entire domain.

With each baseline, the workload queries can be answered by summing the corresponding noisy cell counts.

## 3.2 Empirical Comparisons in Prior Work

We review prior studies with a focus on two things. First, we characterize what is known about the state of the art for both 1D and 2D settings and identify gaps and inconsistencies. Second, we look at the algorithmic strategies described in Section 3.1 and what is known about their effect on performance.

For the 1D setting, the study of Qardaji et al. [22] suggests that  $H_b$  generally is the best data-independent algorithm, achieving lower error than many competing techniques, including IDENTITY, H, PRIVELET, SF, and others [4, 17, 18, 28]. However, their study does not include any data-dependent algorithms other than SF. Thus, their study does not address whether  $H_b$  can outperform the data-dependent algorithms listed in Table 1, a gap in knowledge that has yet to be resolved.

The results of Hardt et al. [10] suggest that  $H_b$  and the other data-independent techniques studied in [22] would be outperformed by MWEM for both 1D and 2D range queries. In their study, MWEM almost always achieves an error rate that is less than a lower bound that applies to any instance of the matrix mechanism [16, 18], a generic class of differentially private algorithms that includes the top performers from the study of Qardaji et al. [22] and all of the data-independent techniques listed in Table 1.

Data-dependent and data-independent techniques are also compared in Li et al. [15] and their findings are inconsistent with the results of Hardt et al. Specifically, MWEM and other data-dependent techniques do not always outperform matrix mechanism techniques; for some “hard” datasets, the error of MWEM is 2-10 $\times$  higher than PRIVELET, one of the matrix mechanism instances included in their study. The authors investigate the difficulty of datasets, but do not identify what properties of the input affect the performance of data-dependent algorithms. In this paper, by explicitly controlling for scale and shape, we are able to resolve this gap and offer a much richer characterization of when data-dependent algorithms do well and explain the apparent contradiction between [10] and [15].

For the 2D setting, we are not aware of any work that offers a comprehensive evaluation of all available techniques. The 2D study of Qardaji et al. [22] only compares  $H_b$  against IDENTITY. A second work by Qardaji et al. [21] compare their proposed algorithms with four techniques from the literature, and so is not as comprehensive as our study here. Li et al. [15] which appeared later did not compare to [21]. Thus, another gap in the literature is a clear picture of the state of the art for 2D.

Finally, we highlight key findings regarding the algorithmic strategies discussed in Section 3.1. For hierarchical aggregation, Qardaji et al. [22] carefully consider the effect of domain size and show that hierarchies can achieve significantly lower error than “flat” approaches like IDENTITY but only when the domain size is sufficiently large. Further, the minimum domain size increases exponentially with dimensionality. It must be at least 45 for 1D; at least  $64^2$  for 2D, and at least  $121^3$  for 3D. For the data-dependent strat-

egy of partitioning, a number of works show that it leads to low error for some datasets [1, 4, 15, 21, 26, 27, 29]; however, we are not aware of any work that characterizes precisely when it improves upon data-independent techniques.

In summary, the existing literature has shown that data-dependent algorithms can sometimes do well and that domain size is an important factor in algorithm performance. However, it has not clarified whether the best data-dependent algorithms can consistently outperform data-independent algorithms, and also leaves open the question of what factors, beyond domain size, affect performance.

## 4. EVALUATION PRINCIPLES

In this section we present fundamental principles that our evaluation framework DPBENCH (see Section 5) is based on. These are principles that are often not followed in prior work, and when unheeded, can result in (a) gaps/incomplete understanding of the performance of differentially private algorithms, or (b) unfair comparison of algorithms. Our principles are categorized into three classes: *diversity of inputs*, *end-to-end private algorithms* and *sound evaluation of outputs*.

### 4.1 Diversity of Inputs

The first set of principles pertain to key inputs to the algorithm: the private data vector  $x$  and the differential privacy parameter  $\epsilon$ . The intent of the principles is to ensure that algorithms are evaluated across a diverse set of inputs in order to provide a data owner with a comprehensive picture of algorithm performance. Almost all work adheres to Principle 1, but we state it for completeness.

**PRINCIPLE 1 ( $\epsilon$  DIVERSITY).** *Evaluate algorithms under different  $\epsilon$  values.*

In addition, most empirical work compares algorithms on several datasets. We identify three critical characteristics of the dataset – *domain size*, *scale*, and *shape*, as described in Section 2.2 – each of which significantly impacts the behavior of privacy algorithms. A novelty of our evaluation approach is to study the sensitivity of algorithm error to each of these properties (while holding the other two constant).

**PRINCIPLE 2 (SCALE DIVERSITY).** *Evaluate algorithms using datasets of varying scales.*

**PRINCIPLE 3 (SHAPE DIVERSITY).** *Evaluate algorithms using datasets of varying shapes.*

**PRINCIPLE 4 (DOMAIN SIZE DIVERSITY).** *Evaluate algorithms using datasets (and workloads) defined over varying domain sizes.*

We cannot anticipate in advance what dataset the algorithm might be run on, and what settings of  $\epsilon$  a data owner may select. A sound evaluation should be able to predict algorithm performance on as many values of  $\epsilon$ , scale, shape and domain size as possible. This is especially important for data-dependent algorithms as their performance changes not only quantitatively but also *qualitatively* when changing these settings.

Our empirical comparisons of differentially private algorithms for range queries (Section 7), in both the one- and two-dimensional cases, show that the algorithms offering least error vary depending on the values of  $\epsilon$ , scale, shape, and domain size. Because current algorithms offer no clear winner, evaluation on diverse inputs remains very important.

**Remarks on Prior Work.** Empirical evaluations in prior work only compared algorithms on a single shape (e.g., [4, 26]), or a single domain size (e.g., [15]), or on multiple datasets without controlling for each of the input characteristics (e.g. [1, 10, 15, 27, 29]). These result in some of the gaps/inconsistencies in algorithm performance discussed in Section 3.2.

### 4.2 End-to-End Private Algorithms

The next three principles require that algorithms provide equivalent privacy guarantees to ensure fair comparisons. While all published algorithms include a proof of differential privacy, some actions taken during deployment or evaluation of the algorithm can undermine that guarantee in subtle ways.

**PRINCIPLE 5 (PRIVATE PRE- AND POST-PROCESSING).** *Any computation on the input dataset must be accounted for in the overall privacy guarantee of the algorithm being evaluated.*

Any computation that comes before or after the execution of the private algorithm must also be considered in the analysis of privacy. This includes pre-processing steps such as data cleaning as well as any post-processing (e.g., selecting the “best” output from among a set of outputs). If necessary, a portion of the user’s  $\epsilon$  privacy budget can be devoted to these tasks.

**PRINCIPLE 6 (NO FREE PARAMETERS).** *Include in every algorithm definition a data-independent or differentially private method for setting each required parameter.*

An important special case of preprocessing is parameter selection. Many published algorithms have additional parameters (beyond  $\epsilon$ ) whose setting can have a large impact on the algorithm’s behavior. Borrowing a notion from physics, a parameter is considered *free* if it can be adjusted to fit the data – in other words, the value of the parameter that optimizes algorithm performance is data-dependent. Free parameters are problematic if they are later set by tuning them to the input data. In a production environment, it risks violating the differential privacy guarantee. In a research environment, it may impede a fair evaluation of algorithm performance, as some algorithms may be more sensitive to tuning than others. A principled methodology of parameter tuning is proposed in Section 5.2 and evaluated in Section 7.3.

**PRINCIPLE 7 (KNOWLEDGE OF SIDE INFORMATION).** *Use public knowledge of side information about the input dataset (e.g., scale) consistently across algorithms.*

Some algorithms assume that certain properties of the input dataset (e.g., scale) are considered public and use this during the execution of the algorithm. Other algorithms do not use this information, or allocate some privacy budget to estimate it, leading to inconsistent comparisons. In general, if the availability of “side information” is not accounted for properly, it can weaken the privacy guarantee.

**Remarks on Prior Work.** Some work on differentially private machine learning pre-processes the input dataset using non-private data transformation algorithms [3, 9]. In other work, parameters were tuned on the same datasets on which the algorithm’s performance is later evaluated [4, 10, 24, 27, 29]. Moreover, a few algorithms [10, 21, 27] assume that the scale of the dataset is known.

### 4.3 Sound Evaluation of Outputs

The last four principles pertain to algorithm evaluation.

**PRINCIPLE 8 (MEASUREMENT OF VARIABILITY).** *Algorithm output should be evaluated by expected error as well as a measure of error variability.*

The error of a differentially private algorithm is a random variable. Most evaluations report its expectation or estimate it empirically. However, when the algorithm is deployed, a user will apply it to private data and receive a single output. Thus, it is also important to measure variability of error around its expectation. Note that this is different from reporting error bars representing standard error, which measures the uncertainty in the estimate of mean error and shrinks with increasing trials. Instead, we want to measure how much the error of any single output might deviate from the mean.

**PRINCIPLE 9 (MEASUREMENT OF BIAS).** *Algorithm output should be evaluated for bias in the answer.*

In addition to the expected error, a sound evaluation should also check whether the algorithm’s output is biased. For instance, the Laplace mechanism is unbiased; i.e., the expected value of the output of the Laplace mechanism is equal to the original input. An attractive property of unbiased algorithms like the Laplace mechanism is that they are consistent – as  $\epsilon$  tends to  $\infty$ , the error tends to 0. Recent data-dependent algorithms incur less error than the Laplace mechanism by introducing some bias in the output. Understanding the bias is important; we show in this paper (Section 7.4) that some algorithms incur a bias (and therefore incur error) even when  $\epsilon$  tends to infinity.

**PRINCIPLE 10 (REASONABLE PRIVACY AND UTILITY).** *Evaluate algorithms for input settings that result in reasonable privacy and utility.*

Under extreme settings of the input parameters, one algorithm may outperform another, but both algorithms may produce useless results. Similarly, it is not meaningful to make relative comparisons of algorithm performance when a reasonable privacy guarantee is not offered.

*Remarks on Prior Work.* Almost all prior empirical work on differential privacy (with the exception of those that prove theoretical bounds on the error in terms of  $(\alpha, \delta)$ -usefulness [2]) focus on mean error and not its variability. Algorithm bias is seldom analyzed. While most algorithms are explicitly compared to a baseline like IDENTITY ([4] is an exception), none of algorithms have been compared with a data-dependent baseline like UNIFORM.

## 5. DPBENCH EVALUATION FRAMEWORK

In this section, we describe DPBENCH, a framework for evaluating the accuracy of differentially private algorithms. DPBENCH’s goal is to formulate a set of standards for empirical evaluation that adhere to the principles from Section 4. The framework can be applied to a number of analysis tasks. Because different analysis tasks require the use of disparate algorithms, it is necessary to instantiate distinct benchmarks for each kind of task.<sup>1</sup> DPBENCH aims to provide a common framework for developing these benchmarks.

We define a benchmark as a 9-tuple  $\{\mathcal{T}, \mathcal{W}, \mathcal{D}, \mathcal{M}, \mathcal{L}, \mathcal{G}, \mathcal{R}, \mathcal{E}_M, \mathcal{E}_I\}$ .  $\mathcal{T}$  denotes the task that the benchmark is associated with.  $\mathcal{W}$  denotes the set of query workloads that are representative for the task  $\mathcal{T}$ . For instance, if  $\mathcal{T}$  is the task of “answering one-dimensional range queries”,  $\mathcal{W}$  could contain a workload of random range queries. Note that one could use more than one workload to evaluate algorithms for a task. Evaluating differentially private algorithms is typically done on publicly available datasets, which we refer to as a *source* data instance  $I$ .  $\mathcal{D}$  denotes a set of such source datasets.

<sup>1</sup>This is akin to the various TPC benchmarks for evaluating database performance.

$\mathcal{M}$  lists the set of algorithms that are compared by the benchmark.  $\mathcal{L}$  denotes *loss functions* that are used to measure the error (or distance) between  $\mathbf{y} = \mathbf{W}\mathbf{x}$ , the true answer to a workload, and  $\hat{\mathbf{y}}$ , the output of a differentially private algorithm in  $\mathcal{M}$ . An example of a loss function that we will consistently use in this paper is the  $L_2$  norm of the difference between  $\mathbf{y}$  and  $\hat{\mathbf{y}}$ ; i.e. error is  $\|\mathbf{W}\mathbf{x} - \hat{\mathbf{y}}\|_2$ .

We call  $\mathcal{W}, \mathcal{D}, \mathcal{M}$  and  $\mathcal{L}$  *task specific components* of a DPBENCH benchmark, since they vary depending on the task under consideration. We discuss them in detail in Section 6. On the other hand the remaining components  $\mathcal{G}, \mathcal{R}, \mathcal{E}_M, \mathcal{E}_I$  are task independent, and help benchmarks to adhere to the principles in Section 4.  $\mathcal{G}$  is a *data generator* that ensures diversity of inputs (Principles 1 to 4).  $\mathcal{R}$  defines a set of *algorithm repair functions* that ensure that the algorithms being evaluated satisfy end-to-end privacy (Principles 5 to 7). In particular, DPBENCH provides a technique to automatically train free parameters of an algorithm.  $\mathcal{E}_M$  and  $\mathcal{E}_I$  denote standards for measuring and interpreting empirical error. These help adhere to Principles 8 to 10 by explicitly defining standards for measuring variation and bias, declaring winners among algorithms, and defining baselines for interpreting error. Next, we describe these in detail, and on the way define two novel theoretical properties *scale-epsilon exchangeability* and *consistency*.

### 5.1 Data Generator $\mathcal{G}$

Let  $I \in \mathcal{D}$  be a source dataset with true domain  $D_I = (A_1 \times \dots \times A_k)$  and true scale  $s_I$ . The data generator  $\mathcal{G}$  takes as input a scale  $m$ , and a new domain  $D$ , which may be constructed by (i) considering a subset of attributes  $\mathbb{B}$ , and (ii) coarsening the domain of attributes in  $\mathbb{B}$ . Note that  $m$  and  $D$  are possibly different from the true scale  $s_I$  and domain  $D_I$ .  $\mathcal{G}$  outputs a new data vector  $\mathbf{x}$  with the new scale  $m$  and new domain  $D$  as follows. First,  $\mathcal{G}$  computes a histogram  $\mathbf{x}'$  of counts on the new domain based on  $I$ . Then,  $\mathcal{G}$  normalizes  $\mathbf{x}'$  by its true scale to isolate the *shape*  $\mathbf{p}$  of the source data on the new domain  $D$ .  $\mathcal{G}$  generates  $\mathbf{x}$  by sampling  $m$  times (with replacement) from  $\mathbf{p}$ . If we set the desired scale to the true scale  $s_I$  we will generate a data vector  $\mathbf{x}$  that is approximately the same as the original (modulo sampling differences).

The output vector  $\mathbf{x}$  of  $\mathcal{G}$  is used as input to the privacy algorithms. Diversity of scale (Principle 2) is achieved by varying  $m$ . Diversity of domain size (Principle 4) is achieved by varying  $D$ . Shape is primarily determined by  $I$  but also impacted by the domain definition,  $D$ , so diversity of shape (Principle 3) requires a variety of source datasets and coordinated domain selection.

The data generator  $\mathcal{G}$  outputs datasets that help evaluate algorithms on each of the data characteristics shape, scale and domain size while keeping the other two properties constant. Without such a data generator, the effects of shape and scale on algorithm error can’t be distinguished, thus prior work was unable to correctly identify features of datasets that impacted error. For a fixed shape, increasing scale offers a stronger “signal” about the underlying properties of the data. In addition, this sampling strategy always results in datasets with integral counts (simply multiplying the distribution by some scale factor may not).

### 5.2 Algorithm Repair Functions $\mathcal{R}$

While automatically verifying whether an algorithm performs pre- or post-processing that violates differential privacy is out of the scope of this benchmark, we discuss two *repair functions* to help adhere to the free parameters and side information principles (Principles 6 and 7, respectively).

*Learning Free Parameter Settings  $R_{param}$ .* We present a first cut solution to handling free parameters. Let  $\mathcal{K}_\theta$  denote a private algorithm  $\mathcal{K}$  instantiated with a vector of free parameters  $\theta$ .

We use a separate set of datasets to tune these parameters; these datasets will *not* be used in the evaluation. Given a set of training datasets  $\mathcal{D}_{train}$ , we apply data generator  $\mathcal{G}$  and learn a function  $R_{param} : (\epsilon, \|\mathbf{x}\|_1, n) \rightarrow \theta$  that given  $\epsilon$ , scale, and the domain size, outputs parameters  $\theta$  that result in the lowest error for the algorithm. Given this function, the benchmark extends the algorithm by adaptively selecting parameter settings based on scale and epsilon. If the parameter setting depends on scale, a part of the privacy budget is spent estimating scale, and this introduces a new free parameter, namely the budget spent for estimating scale. The best setting for this parameter can also be learned in a similar manner.

*Side Information  $R_{side}$ .* Algorithms which use non-private side information can typically be corrected by devoting a portion of the privacy budget to learning the required side information, then using the noisy value in place of the side information. This process is difficult to automate but may be possible with program analysis in some cases. This has the side-effect of introducing a new parameter which determines the fraction of the privacy budget to devote to this component of the algorithm, which in turn can be set using our learning algorithm (described above).

### 5.3 Standards for Measuring Error $\mathcal{E}_M$

*Error.* DPBENCH uses *scaled average per-query error* to quantify an algorithm’s error on a workload.

**DEFINITION 3 (SCALED AVERAGE PER-QUERY ERROR).** *Let  $\mathbf{W}$  be a workload of  $q$  queries,  $\mathbf{x}$  a data vector and  $s = \|\mathbf{x}\|_1$  its scale. Let  $\hat{\mathbf{y}} = \mathcal{K}(\mathbf{x}, \mathbf{W}, \epsilon)$  denote the noisy output of algorithm  $\mathcal{K}$ . Given a loss function  $L$ , we define scale average per-query error as  $\frac{1}{s \cdot q} L(\hat{\mathbf{y}}, \mathbf{W}\mathbf{x})$ .*

By reporting scaled error, we avoid considering a fixed absolute error rate to be equivalent on a small scale dataset and a large scale dataset. For example, for a given workload query, an absolute error of 100 on a dataset of scale 1000 has very different implications than an absolute error of 100 for a dataset with scale 100,000. In our scaled terms, these absolute errors would be clearly distinguished as 0.1 and 0.001 scaled error. Accordingly, scaled error can be interpreted in terms of population percentages. Using scaled error also helps us define the scale-epsilon exchangeability property in Section 5.5.

Considering per-query error allows us to compare the error on different workloads of potentially different sizes. For instance, when examining the effect of domain size  $n$  on the accuracy of algorithms answering the identity workload, the number of queries  $q$  equals  $n$  and hence would vary as  $n$  varies.

*Measuring Error.* The error measure (Definition 3) is a random variable. We can estimate properties such as their mean and variance through repeated executions of the algorithm. In addition to comparing algorithms using mean error, DPBENCH also compares algorithms based on the 95 percentile of the error. This takes into account the variability in the error (adhering to Principle 8) and might be an appropriate measure for a “risk averse” analyst who prefers an algorithm with reliable performance over an algorithm that has lower mean performance but is more volatile. Means and 95 percentile error values are computed on multiple independent repetitions of the algorithm over multiple samples  $\mathbf{x}$  drawn from the data generator to ensure high confidence estimates.

DPBENCH also identifies algorithms that are competitive for state-of-the-art performance for each setting of scale, shape and domain size. An algorithm is *competitive* if it either (i) achieves the lowest error, or (ii) the difference between its error and the lowest

error is not statistically significant. Significance is assessed using an unpaired t-test with a Bonferroni corrected  $\alpha = 0.05/(n_{algs} - 1)$ , for running  $(n_{algs} - 1)$  tests in parallel.  $n_{algs}$  denotes the number of algorithms being compared. Competitive algorithms can be chosen both based on mean error (a “risk neutral” analyst) and 95 percentile error (a “risk averse” analyst).

### 5.4 Standards for Interpreting Error $\mathcal{E}_I$

When drawing conclusions from experimental results, Principle 10 should be respected. One way to assess reasonable utility is by comparing with appropriate baselines.

We use IDENTITY and UNIFORM (described in Section 3.1) as upper-bound baselines. Since IDENTITY is a straightforward application of the Laplace mechanism, we expect a more sophisticated algorithm to provide a substantial benefit over the error achievable with IDENTITY. Similarly, UNIFORM learns very little about  $\mathbf{x}$ , only its scale. An algorithm that offers error rates comparable or worse than UNIFORM is unlikely to provide useful information in practical settings. Note that there might be a few settings where these baselines can’t be beaten (e.g., when shape of  $\mathbf{x}$  is indeed uniform). However, an algorithm should be able to beat these baselines in a majority of settings.

### 5.5 Scale Epsilon Exchangeability

We describe next a novel property of most algorithms due to which increasing  $\epsilon$  and scale both have an equivalent effect on algorithm error. We say an algorithm is *scale-epsilon exchangeable* if increasing scale by a multiplicative factor  $c$  has a precisely equivalent effect, on the scaled error, as increasing epsilon by a factor of  $c$ . Exchangeability has an intuitive meaning: to get better accuracy from an exchangeable algorithm, a user can either acquire more data or, equivalently, find a way to increase their privacy budget. Moreover, for such algorithms, diversity of scale (Principle 2) implies diversity of  $\epsilon$  (Principle 1).

**DEFINITION 4 (SCALE-EPSILON EXCHANGEABILITY).** *Let  $\mathbf{p}$  be a shape and  $\mathbf{W}$  a workload. If  $\mathbf{x}_1 = m_1\mathbf{p}$  and  $\mathbf{x}_2 = m_2\mathbf{p}$ , then algorithm  $\mathcal{K}$  is scale-epsilon exchangeable if  $error(\mathcal{K}(\mathbf{x}_1, \mathbf{W}, \epsilon_1)) = error(\mathcal{K}(\mathbf{x}_2, \mathbf{W}, \epsilon_2))$  whenever  $\epsilon_1 m_1 = \epsilon_2 m_2$ .*

We prove in Appendix C that all but one of the algorithms we evaluate in this paper are scale-epsilon exchangeable. The exception is SF [27], which is not exchangeable but empirically behaves so. An important side-effect of this insight is that in empirical evaluations one can equivalently vary scale rather than  $\epsilon$  if the algorithm is proven to satisfy scale- $\epsilon$  exchangeability.

### 5.6 Consistency

As  $\epsilon$  increases, the privacy guarantee weakens. A desirable property of a differential privacy algorithm is that its error decreases with increasing  $\epsilon$  and ultimately disappears as  $\epsilon \rightarrow \infty$ . We call this property consistency. Algorithms that are not consistent are inherently biased.

**DEFINITION 5 (CONSISTENCY).** *An algorithm satisfies consistency if the error for answering any workload tends to zero as  $\epsilon$  tends to infinity.*

## 6. EXPERIMENTAL SETUP

In this section we use our benchmark framework to define concrete benchmarks for 1- and 2-D range queries. For each benchmark, we precisely describe the task-specific components of the benchmark  $(\mathcal{D}, \mathcal{W}, \mathcal{M}, \mathcal{L})$ , namely the datasets, workloads, algorithms and loss functions with the goal of providing a reproducible standard of evaluation. Experimental results appear in Section 7.

Dataset name	Original Scale	% Zero Counts	Previous works
<i>1D datasets</i>			
ADULT	32,558	97.80%	[10, 15]
HEPPh	347,414	21.17%	[15]
INCOME	20,787,122	44.97%	[15]
MEDCOST	9,415	74.80%	[15]
TRACE	25,714	96.61%	[1, 11, 27, 29]
PATENT	27,948,226	6.20%	[15]
SEARCH	335,889	51.03%	[1, 11, 27, 29]
BIDS-FJ	1,901,799	0%	new
BIDS-FM	2,126,344	0%	new
BIDS-ALL	7,655,502	0%	new
MD-SAL	135,727	83.12%	new
MD-SAL-FA	100,534	83.17%	new
LC-REQ-F1	3,737,472	61.57%	new
LC-REQ-F2	198,045	67.69%	new
LC-REQ-ALL	3,999,425	60.15%	new
LC-DTIR-F1	3,336,740	0%	new
LC-DTIR-F2	189,827	11.91%	new
LC-DTIR-ALL	3,589,119	0%	new
<i>2D datasets</i>			
BJ-CABS-S	4268780	78.17%	[12]
BJ-CABS-E	4268780	76.83%	[12]
GOWALLA	6442863	88.92%	[21]
ADULT-2D	32561	99.30%	[10]
SF-CABS-S	464040	95.04%	[20]
SF-CABS-E	464040	97.31%	[20]
MD-SAL-2D	70526	97.89%	new
LC-2D	550559	92.66%	new
STROKE	19435	79.02%	new

Table 2: Overview of datasets.

## 6.1 Datasets

Table 2 is an overview of the datasets we consider. 11 of the datasets have been used to evaluate private algorithms in prior work. We have introduced 14 new datasets to increase shape diversity. Datasets are described in Appendix A. The table reports the original scale of each dataset. We use the data generator  $\mathcal{G}$  described before to generate datasets with scales of  $\{10^3, 10^4, 10^5, 10^6, 10^7, 10^8\}$ .

The maximum domain size is 4096 for 1D datasets and  $256 \times 256$  for 2D datasets. The table also reports the fraction of cells in  $\mathbf{x}$  that have a count of zero at this domain size. By grouping adjacent buckets, we derive versions of each dataset with smaller domain sizes. For 1D, the domain sizes are  $\{256, 512, 1024, 2048\}$ ; for 2D, they are  $\{32 \times 32, 64 \times 64, 128 \times 128, 256 \times 256\}$ .

For each scale and domain size, we randomly sample 5 data vectors from our data generator and for each data vector, we run the algorithms 10 times.

## 6.2 Workloads & Loss Functions

We evaluate our algorithms on different workloads of range queries. For 1D, we primarily use the *Prefix* workload, which consists of  $n$  range queries  $[1, i]$  for each  $i \in [1, n]$ . The *Prefix* workload has the desirable property that any range query can be derived by combining the answers to exactly two queries from *Prefix*. For 2D, we use 2000 random range queries as an approximation of the set of all range queries.

As mentioned in Section 5, we use  $L_2$  as the loss function.

## 6.3 Algorithms

The algorithms compared are listed in Table 1. The dimension column indicates what dimensionalities the algorithm can support; algorithms labeled as Multi-D are included in both experiments. Complete descriptions of algorithms appear in Appendix B.

## 6.4 Resolving End-to-End Privacy Violations

**Inconsistent side information:** Recall that Principle 7 prevents the inappropriate use of private side information by an algorithm. SF,

MWEM, UGRID, and AGRID assume the true scale of the dataset is known. To gauge any potential advantage gained from side information, we evaluated algorithm variants where a portion of the privacy budget, denoted  $\rho_{total}$ , is used to noisily estimate the scale. To set  $\rho_{total}$ , we evaluated the algorithms on synthetic data using varying values of  $\rho_{total}$ . In results not shown, we find that setting  $\rho_{total} = 0.05$  achieves reasonable performance. For the most part, the effect is modestly increased error (presumably due to the reduced privacy budget available to the algorithm). However, the error rate of MWEM increases significantly at small scales (suggesting it is benefiting from side information). In Section 7, all results report performance of the original unmodified algorithms. While this gives a slight advantage to algorithms that use side information, it also faithfully represents the original algorithm design.

**Illegal parameter setting** Table 1 shows all the parameters used for each algorithm. Parameters with assignments have been set according to fixed values provided by the authors of the algorithm. Those without assignments are free parameters that were set in prior work in violation of Principle 6.

For MWEM, the number of rounds  $T$  is a free variable that has a major impact on MWEM’s error. According to a pre-print version of [10], the best performing value of  $T$  is used for each task considered. For the one-dimensional range query task considered,  $T$  is set to 10. Similarly, for AHP, two parameters are left free:  $\eta$  and  $\rho$  which were tuned on the input data.

To adhere to Principle 6, we use the learning algorithm for setting free parameters (Section 5.2) to set free parameters for MWEM and AHP. In our experiments, the extended versions of the algorithms are denoted MWEM\* and AHP\*. In both cases, we train on shape distributions synthetically generated from power law and normal distributions.

For MWEM\*, we determine experimentally the optimal  $T \in [1, 200]$  for a range of  $\epsilon$ -scale products. As a result,  $T$  varies from 2 to 100 over the range of scales we consider. This improves the performance of MWEM (versus a static setting of  $T$ ) and does not violate our principles for private parameter setting. The success of this method is an example of data-independent parameter setting.

SF requires three parameters:  $\rho, k, F$ . Parameter  $F$  is free only in the sense that it is a function of scale, which is side information (as discussed above). For  $k$ , the authors propose a recommendation of  $k = \lceil \frac{\rho}{10} \rceil$  after evaluating various  $k$  on input datasets. Their evaluation, therefore, did not adhere to Principle 6. However, because our evaluation uses different datasets, we can adopt their recommendation without violating Principle 6 – in effect, their experiment serves as a “training phase” for ours. Finally,  $\rho$  is a function of  $k$  and  $F$ , and thus is no longer free once those are fixed.

## 6.5 Implementation Details

We use implementations from the authors for DAWA, GREEDY H, H, PHP, EFPA, and SF. We implemented MWEM,  $H_b$ , PRIVELET, AHP, DPCUBE, AGRID, UGRID and QUADTREE ourselves in Python. All experiments are conducted on Linux machines running CentOS 2.6.32 (64-bit) with 16 Intel(R) Xeon(R) CPU E5-2643 0 @ 3.30GHz with 16G or 24G of RAM.

## 7. EXPERIMENTAL FINDINGS

We present our findings for the 1D and 2D settings. For the 1D case, we evaluated 14 algorithms on 18 different datasets, each at 6 different scales and 4 different domain sizes. For 2D, we evaluated 14 algorithms on 9 different datasets, each at 6 scales and 4 domain sizes. In total we evaluated 7,920 different experimental configura-

rations. The total computational time was  $\approx 22$  days of single core computation. Given space constraints, we present a subset of the experiments that illustrate key findings.

In results not shown, we compare the error of data-independent algorithms ( $H_b$ , GREEDY H, H, PRIVELET, IDENTITY) on various range query workloads. The error of these algorithms is, of course, independent of the input data distribution, and the absolute error is independent of scale. (Or, in scaled-error terms, it diminishes predictably with increasing scale for all algorithms). The error for this class of techniques is well-understood because it is more amenable to analysis than data-dependent techniques. In order to simplify the presentation of subsequent results, we *restrict our attention to two data-independent algorithms*: IDENTITY, which is a useful baseline, and  $H_b$ , which is the best performing data-independent technique for 1D and 2D range query workloads with larger ranges.

## 7.1 Impact of Input Properties on Performance

As advocated in Principles 1 to 4, we now systematically examine the key properties of algorithm input that can affect performance: scale, domain size, shape, and  $\epsilon$ . To aid in readability of figures, we report mean error without error bars but discuss variability over trials in detail in Section 7.4.

**Scale:** Figs. 1a and 1b compare algorithm performance at a fixed epsilon ( $\epsilon = 0.1$ ) and increasing scales. Because of the scale-epsilon exchangeability property (Definition 4), which almost all algorithms satisfy,<sup>2</sup> the figures look identical if we fix scale and show increasing  $\epsilon$ . For example, the first panel of Fig. 1a reports error for scale=1000 and  $\epsilon = .1$  but the plotted error rates and relationships are identical to those for scale=10,000 and  $\epsilon = .01$ , scale=100,000 and  $\epsilon = .001$ , etc. Figs. 1a and 1b also capture variation across datasets (each black dot is the error of an algorithm on a particular dataset at the given scale; the white diamond is the algorithm’s error averaged over all datasets).

**FINDING 1 (BENEFITS OF DATA-DEPENDENCE).** *Data-dependence can offer significant improvements in error, especially at smaller scales or lower epsilon values.*

The figures show that at small scales, almost all data-dependent algorithms outperform the best data-independent algorithm,  $H_b$ , on at least one dataset (black dot), sometimes by an order of magnitude. Further, the best data-dependent algorithms beat the best data-independent algorithms on average across all datasets (white diamonds), often by a significant margin. (For 1D, at the smallest scale, it is as much as a factor of 2.47; for 2D, it is up to a factor of 3.10.)

**FINDING 2 (PENALTY OF DATA DEPENDENCE).** *Some data-dependent algorithms break down at larger scales (or higher  $\epsilon$ ).*

Despite the good performance of data-dependent algorithms at small scales, at moderate or larger scales, many data-dependent algorithms have error significantly larger than IDENTITY, the data-independent baseline, which indicates quite disappointing performance. Even for the best data-dependent methods, the comparative advantage of data-dependence decreases and, for the most part, eventually disappears as scale increases. At the largest scales in both 1D and 2D, almost all data-dependent algorithms are beaten by  $H_b$ . At these scales, the only algorithms to beat  $H_b$  are DAWA and AHP\*. In the 1D case, this only happens on a couple of datasets; in the 2D case, it happens on about half the datasets.

**Shape:** In Figs. 1a and 1b, the effect of shape is evident in the spread of black dots for a given algorithm.

<sup>2</sup>SF is not exchangeable, but it empirically behaves so.

**FINDING 3 (VARIATION OF ERROR WITH DATA SHAPE).** *Algorithm error varies significantly with dataset shape, even for a fixed scale, and algorithms differ on the dataset shapes on which they perform well.*

Even when we control for scale, each data-dependent algorithm displays significant variation of error across datasets. In the 1D case (Fig. 1a) at the smallest scale, EFPA’s error varies by a factor of 24.88 from lowest error to highest. Across all scales, algorithms like MWEM, MWEM\*, PHP, EFPA, and AHP\* show significantly different error rates across shape. SF and DAWA offer the lowest variation across dataset shapes. In the 2D case (Fig. 1b), we also see significant variation at all scales. (Interestingly, DAWA exhibits high variation in 2D.)

While Figs. 1a and 1b show variation, one cannot compare algorithms across shapes. To do that, we use Figs. 2a and 2b, where scale is fixed and shape is varied. (For readability, we show only a subset of algorithms as explained in the caption for Fig. 2.) These figures show that the comparative performance of algorithms varies across shape. Fig. 2a shows that four different algorithms achieve the lowest error on some shape. In addition, a dataset that is “easy” for one algorithm appears “hard” for another: witness EFPA is able to exploit the shape of BIDS-ALL yet suffers on TRACE; the opposite is true for MWEM. In Fig. 2b, the shapes where DAWA performs poorly, AGRID does quite well. These results suggest that data-dependent algorithms are exploiting different properties of the dataset.

The unpredictable performance across data shape is a challenge in deployment scenarios because privacy algorithms cannot be selected by looking at the input data and because data-dependent algorithms rarely come with public error bounds. We discuss these issues further in Section 8.

**Domain Size:** Fig. 2c shows the effect of domain size for the 2D case. For two datasets (ADULT-2D and BJ-CABS-E) at two different scales ( $10^4$  and  $10^6$ ), it reports error for domain sizes varying from  $32 \times 32$  to  $256 \times 256$ .

**FINDING 4 (DIFFERENT DOMAIN SIZE EFFECT).** *Domain size affects data-dependent algorithms differently than data-independent algorithms.*

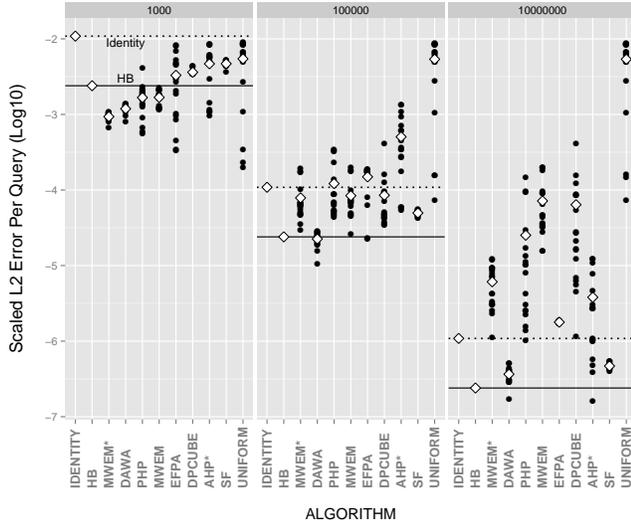
As discussed in [22], the error of data-independent algorithms IDENTITY and  $H_b$  is expected to increase with domain size, and at sufficiently large domains,  $H_b$  should achieve lower error than IDENTITY. Those effects are confirmed in Fig. 2c. However, the data-dependent algorithms exhibit different behavior. AGRID’s error is nearly flat across domain size. This makes sense because the grid size is selected independent of domain size. DAWA’s error is flat for some datasets but for others, it increases with domain size. DAWA selects a partition based on the data shape: for some shapes, the partition selected may grow increasingly fine-grained with domain size.

## 7.2 Assessment of State of the Art

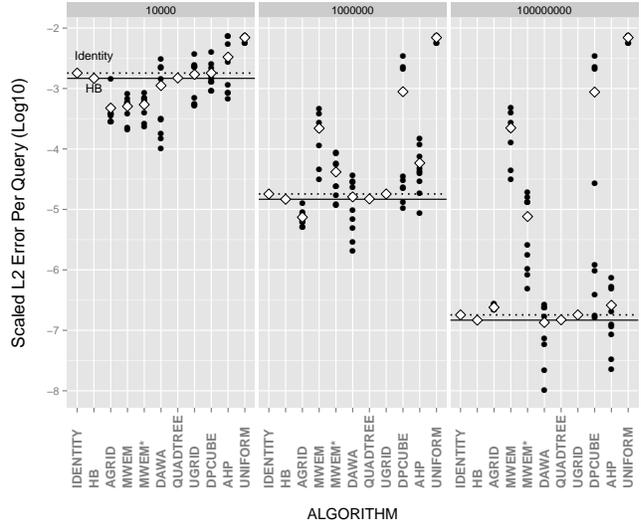
Tables 3a and 3b show the number of datasets for which an algorithm is *competitive* for a given scale. We determine which algorithms are competitive by accounting for the variability in the mean error using t-tests (as discussed in Section 5.4).

**FINDING 5 (COMPETITIVE ALGORITHMS).** *No single algorithm offers uniformly low error. At small scales, data-dependent algorithms dominate; at large scales data-independent algorithms dominate.*

There is no algorithm that dominates all others by offering uniformly lower error. For 1D, 8 algorithms are competitive for at



(a) 1D case: domain=4096, workload=Prefix



(b) 2D case: domain=128x128, workload=2000 random range queries

Figure 1: These figures show how average error rates of algorithms vary across the datasets described in Table 2, for both the 1D case (a) and 2D case (b). Each black dot (●) corresponds to average error for one dataset; white diamonds (◊) correspond to the mean error over all datasets. Horizontal lines are used to emphasize the error rates of the best and worst data-independent algorithms.

	Scale		
	$10^3$	$10^5$	$10^7$
DAWA	8	17	3
$H_b$		14	16
MWEM*	15		
EFPA	4	2	
PHP	2		
MWEM	3		
UNIFORM	3		
AHP*			1

(a) 1D case for domain size 4096

	Scale		
	$10^4$	$10^6$	$10^8$
DAWA	5	3	5
AGRID	5	6	
$H_b$			4
QUADTREE			4
AHP			1

(b) 2D case for domain size  $128 \times 128$ .

Table 3: Tables show the number of datasets on which algorithms are *competitive*, based on statistical significance of mean error.

least one combination of scale and shape; for 2D, 5 algorithms are competitive. At small scales for both 1D and 2D, the competitive algorithms are all data-dependent. At the largest scales, the only truly data-dependent algorithms are DAWA and AHP\* (in the 2D case, QUADTREE is effectively data-independent at this domain size because the leaves of the tree are individual cells).

While no algorithm universally dominates across settings, DAWA offers the best overall performance in the following sense. We compare the error experienced by a user who selects a single algorithm to run on all datasets and scales to the error experienced by a user with access to an oracle allowing them to select the optimal algorithm (which would vary among the algorithms mentioned above on a case-by-case basis). We compute the ratio of errors per dataset/scale and then compute the average ratio, using a geometric mean. We call this measure “regret.” DAWA has a regret of 1.32 on 1D and 1.73 on 2D. No single algorithm achieves lower regret (the next best on 1D is  $H_b$  with regret of 1.51 and on 2D it is AGRID with regret of 1.90).

### 7.3 Tuning Free Parameters

FINDING 6 (IMPROPER TUNING SKEWS EVALUATION). *For some algorithms tuning free parameters to fit the data can reduce error substantially, in some cases by a factor larger than 7.5.*

To demonstrate the sensitivity of performance to parameter settings, we ran AHP, DAWA, and MWEM under several different scenarios (i.e., 1D datasets of varying scale and shape) and found settings of their respective parameters that were optimal for some scenario (i.e., achieved lowest error on some 1D dataset at some scale). Then for a particular 1D dataset, MEDCOST, at a scale of  $10^5$ , we measure the lowest error and the highest error achieved, considering *only those parameter settings that were optimal for some scenario*. All three algorithms are significantly impacted by the choice of parameters. Errors can be 2.5x (for DAWA) and around 7.5x (for MWEM and AHP) larger using sub-optimal parameters (even though those same parameters were in fact optimal for other reasonable inputs).

We proposed a method (Section 6.4) for setting free parameters in a manner consistent with Principle 6 and for removing dependence on side information (Principle 7). We applied this procedure to MWEM and AHP—violators of Principle 6—to yield MWEM\* and AHP\* respectively. We report our results for the 1D case.

FINDING 7 (IMPROVED MWEM). *Training on synthetic data to decide free parameters (described in Section 6.4) leads to significant performance improvements for MWEM, especially at large scales.*

Ratio of error (MWEM/MWEM*), average all datasets						
Scale	$10^3$	$10^4$	$10^5$	$10^6$	$10^7$	$10^8$
Error ratio	1.799	.951	1.063	5.166	12.000	27.875

The reason for this improved performance is that a stronger signal (here higher scale; equivalently higher epsilon) allows MWEM to benefit from a higher number of rounds because a larger number measurements can be taken with reasonable accuracy. The training procedure, even with a single synthetic dataset, can help to find the right relationship between the scale-epsilon product and  $T$ . Since our method for setting  $T$  does not depend on the input data, this result shows that  $T$  can be set profitably in a data-independent way.

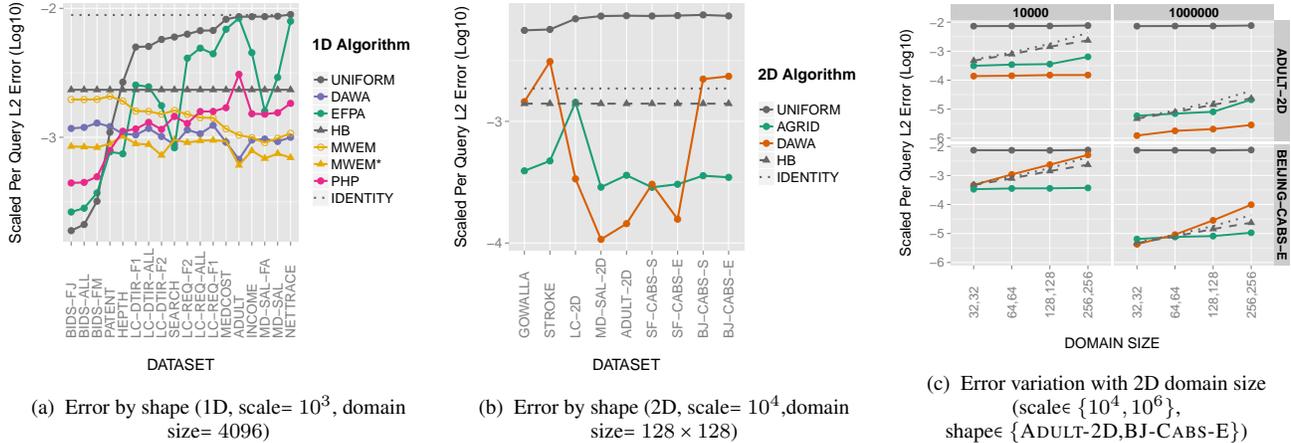


Figure 2: Error variation with data shape (1D and 2D cases) and domain size (2D case, for two shapes and two scales). The only algorithms shown are baselines,  $H_b$ , and those data-dependent algorithms that were competitive (on some dataset) for the scales shown.

This result also shows that we can cope with the removal of side information of scale without a significant impact on error.

For AHP, we applied our training method to determine parameters  $\eta$  and  $\rho$ . We compare the performance of AHP\* against AHP using one of the several parameters settings used in [29]. At smaller scales, the error rates of both algorithms were similar. At larger scales, the results were mixed: on some datasets AHP\* outperformed the fixed setting but on others vice versa. On average, over all datasets, AHP\* was slightly better, with a ratio of improvement of 1.03 at scale  $10^7$ . This shows that parameter tuning for AHP is more sensitive to shape and that it may be worth using a portion of the privacy budget to derive parameters.

## 7.4 Measurement of Variability and Bias

### FINDING 8 (RISK AVERSE ALGORITHM EVALUATION).

*Algorithms differ in the variation in error. The algorithm with lowest mean error may not be preferred by a risk averse user.*

In a real deployment scenario, a user will receive the output of a single run of the algorithm. Thus, she has no opportunity to reliably achieve mean error rates if error rates vary considerably around the mean. We are not aware of any prior work that has considered this issue in empirical algorithm evaluation.

While a risk-neutral user may evaluate algorithms with respect to mean error, a *risk averse* user should evaluate algorithms with respect to error rates she is very likely to achieve. Therefore, for the risk averse user, we consider the 95<sup>th</sup> percentile error rate (i.e. the error rate  $X$  for which 95% of trials have error less than  $X$ ).

In most cases algorithms that are the best according to mean error are also the best according to 95<sup>th</sup> percentile error. In 1D, we see that DAWA has high variability in error and hence in ten scenarios (i.e., 10 scale-shape pairs at domain size 4096), we see DAWA being the best under mean error but not under 95% error. We see three scenarios (PATENT at scale  $10^3$ , MD-SAL at scale  $10^5$  and TRACE at scale  $10^7$ , all at domain size 4096) where an algorithm that was not competitive under mean error has the least 95% error. That algorithm was either UNIFORM or  $H_b$ , both of which have low variability in error. In 2D we see very few scenarios where an algorithm has the least mean error but not the least 95% error.

**FINDING 9 (BIAS AND CONSISTENCY).** *The high error of MWEM, MWEM\*, PHP, and UNIFORM at large scales is due to bias. These algorithms are proven to be inconsistent.*

As scale increases, one expects error to decrease. However, Fig. 1 shows that the rate of decrease varies by algorithm. In fact, for some algorithms – PHP (1D only), MWEM, MWEM\*, and UNIFORM – error remains quite high even at large scales. The cause is due to inherent bias introduced by these algorithms. In empirical results (not shown) we decompose error into bias and variance terms and find that at scale increases, the error of these algorithms becomes dominated by bias.

We complement our empirical findings with theoretical analysis. We show these algorithms are, in fact, inconsistent – implying that the bias does not vanish even as scale (or  $\epsilon$ ) goes to  $\infty$ . Table 1 reports a complete analysis of consistency for all algorithms; proofs appear in Appendix C.

## 7.5 Findings: Reasonable Utility

The previous section compared published techniques in a relative sense: it examined which algorithm performs the best in a given experimental setting. In this section, we evaluate the algorithms in an absolute sense: do the published algorithms offer reasonable utility? This is not an easy question to answer, as the notion of “reasonable” may be specific to the particular task that an analyst wishes to perform on the dataset. We address this by identifying situations when the error is unreasonably high for any application by comparing algorithms against a few simple baselines.

**FINDING 10 (COMPARISON TO BASELINES).** *For both 1D and 2D, many algorithms are beaten by the IDENTITY baseline at large scales. For 1D, the UNIFORM baseline beats all algorithms on some datasets at small scales.*

The experiments include two baseline methods (defined in Section 3.1) that can help us assess whether algorithms are providing meaningful improvements. The first baseline, UNIFORM learns virtually nothing about the input data (only its scale) and exploits a simplistic assumption of uniformity. Therefore, we argue that when sophisticated algorithms are delivering error rates comparable to UNIFORM, even on datasets that are decidedly non-uniform, they are not providing reasonable utility. Fig. 1 shows that most algorithms outperform UNIFORM at larger scales in 1D and at all scales in 2D. However, for the 1D case at scale 1000, UNIFORM achieves lowest error on some datasets. By examining Fig. 2a, one can see on which datasets UNIFORM beats competing algorithms. This figure shows that on some datasets, many algorithms do not

offer much improvement over UNIFORM, suggesting that results for this scale (at least at this  $\epsilon$ ) are unlikely to be useful.

The second baseline is IDENTITY. A sophisticated algorithm that does not consistently and significantly improve over IDENTITY does not justify its complexity. This standard is highly dependent on scale, since, in terms of scaled error, IDENTITY improves with scale. For 1D, if we examine Fig. 1a and compare mean error across all datasets (the white diamonds), this standard rules out PHP, EFPA, AHP\* at moderate scales ( $10^5$ ) and EFPA, AHP\*, MWEM, MWEM\*, PHP, DPCUBE at large scales ( $10^7$ ). For 2D (Fig. 1b), this standard rules out MWEM, MWEM\*, DPCUBE, AHP at moderate scales ( $10^6$ ) and AGRID, MWEM, MWEM\*, DPCUBE, AHP at large scales ( $10^8$ ).

## 8. DISCUSSION AND TAKEAWAYS

In this section, we synthesize our findings to (a) explain gaps or resolve inconsistencies from prior work, (b) present lessons for practitioners and (c) pose open research questions.

*Explaining Results from Prior Work.* We revisit the gaps and inconsistencies identified in Section 3.2 in light of our findings.

- We systematically compared data-dependent algorithms against the state of the art data-independent ones and learn there is no clear winner. Both scale and shape significantly affect algorithm error. Moreover, shape affects each data dependent algorithm differently.
- We can resolve the apparent inconsistency between [10] and [15] regarding the comparative performance of MWEM against data-independent matrix mechanism techniques by accounting for the effect of scale as well as the scale-epsilon exchangeability property. MWEM outperforms data-independent techniques (including instances of the matrix mechanism) either when the scale is small or when  $\epsilon$  is small. Hardt et al. evaluate on small datasets and low  $\epsilon$  whereas the datasets of Li et al. where MWEM struggled turn out to be datasets with large scale (over 335,000). We demonstrated this by using the same datasets as [10, 15] and controlling for scale.
- Our results are consistent with the findings of Qardaji et al. [22] regarding domain size and data-independent hierarchical algorithms. Interestingly, however, we show that data-dependent algorithms have a different relationship with domain size.
- We conduct a comprehensive evaluation of algorithms for answering 2D range queries, which is missing from the literature.

*Lessons for practitioners.* Our results identify guidelines for practitioners seeking the best utility for their task and facing the daunting task of selecting and configuring an appropriate algorithm.

The first consideration for a practitioner should be the overall strength of the "signal" available to them. This is determined by both the  $\epsilon$  budget and the scale of the dataset. (We have shown that these two factors are exactly exchangeable in their impact on scaled error.) In a "high signal" regime (high scale, high  $\epsilon$ ), it is unlikely that any of the more complex, data-dependent algorithms will beat the simpler, easier-to-deploy, data independent methods such as IDENTITY and  $H_b$ . This greatly simplifies algorithm selection and deployment because error bounds are easy to derive, performance doesn't depend on the input dataset, and there are few parameters to set for these algorithms. In "low signal" regimes (low scale, low  $\epsilon$ ), deploying a data-dependent algorithm should be seriously considered, but with an awareness of the limitations: depending on the properties of the input data, error can vary considerably and error bounds are not provided by the data-dependent algorithms. For answering 1D range queries, DAWA is a competitive choice on most datasets in the low and medium signal regimes. For 2D range queries, AGRID consistently beats the data indepen-

dent techniques, but DAWA can significantly outperform AGRID and the data independent techniques on very sparse datasets.

*Open research Problems.* Our experimental findings raise a number of research challenges we believe are important for advancing differentially private algorithms.

- *Understanding Data Dependence:* We have shown that data-dependent algorithms do not appear to be exploiting the same features of shape. The research community appears to know very little about the features of the input data that permit low error. Are there general algorithmic techniques that can be effective across diverse datasets, or should algorithm designers seek a set of specialized approaches that are effective for specific data properties?
- *Algorithm Selection:* While our evaluation of 1D and 2D algorithms identifies a class of state-of-the-art algorithms, it still does not fully solve the problem faced by the practitioner of selecting the algorithm that would result in the least error given a new dataset. We believe further research into analytical and empirical methods for algorithm selection would greatly aid the adoption of differential privacy algorithms in real world systems.
- *Error Bounds:* A related problem is that data-dependent algorithms typically do not provide public error bounds (unlike, e.g., the Laplace mechanism). Hence, users cannot predict the error they will witness without knowing the dataset, presenting a major challenge for algorithm deployment. Developing publishable error bounds is particularly important for this class of algorithms.
- *Parameter Tuning:* We showed that parameter tuning can result in significant gains in utility. Our training procedure is a first step towards setting parameters, but already distinguishes between a case where simple tuning can greatly improve performance over a fixed value (MWEM) and a case where parameter setting is more challenging because of data-dependence (AHP).
- *Expected Error versus Variability:* Existing empirical evaluations have focused on mean error while ignoring the variation of error over trials. We show that risk-averse users favoring low variability over low expected error may choose different algorithms than risk-seeking users. Current algorithms do not offer users the ability to trade off expected error and variability of achieved error, which could be an important feature in practice.

## 9. CONCLUSION

We have presented DPBENCH a novel and principled framework for evaluating differential privacy algorithms. We use DPBENCH to evaluate algorithms for answering 1- and 2-D range queries and as a result (a) resolved gaps/inconsistencies in prior work, (b) identified state-of-the-art algorithms that achieve the least error for their datasets, and (c) posed open research questions.

We are eager to extend our investigation in a number of ways. We have focused here primarily on evaluating the utility of algorithms, assuming that standard settings of  $\epsilon$  offer sufficient privacy. We would like to provide meaningful guidelines for setting privacy parameters along with utility standards. We hope to expand our investigation to broader tasks (beyond 1- and 2-D range queries), and, as noted in Table 1, some of the algorithms considered support broader classes of workloads and may offer advantages not seen in our present experiments.

**Acknowledgments** We appreciate the comments of each of the anonymous reviewers. This material is based upon work supported by the National Science Foundation under Grant Nos. 1012748, 1253327, 1408982, 1443014, 1409125, and 1409143.

## 10. REFERENCES

- [1] G. Ács, C. Castelluccia, and R. Chen. Differentially private histogram publishing through lossy compression. In *ICDM*, pages 1–10, 2012.
- [2] A. Blum, K. Ligett, and A. Roth. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)*, 60(2):12, 2013.
- [3] K. Chaudhuri and S. A. Vinterbo. A stability-based validation procedure for differentially private machine learning. In *Advances in Neural Information Processing Systems*, pages 2652–2660, 2013.
- [4] G. Cormode, M. Procopiuc, E. Shen, D. Srivastava, and T. Yu. Differentially private spatial decompositions. In *ICDE*, pages 20–31, 2012.
- [5] C. Dwork. Differential privacy: A survey of results. In *TAMC*, 2008.
- [6] C. Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95, 2011.
- [7] C. Dwork, F. M. K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.
- [8] C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Found. and Trends in Theoretical Computer Science, 2014.
- [9] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *USENIX Security*, 2014.
- [10] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In *NIPS*, 2012.
- [11] M. Hay, V. Rastogi, G. Miklau, and D. Suciuc. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1-2):1021–1032, 2010.
- [12] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava. Dpt: differentially private trajectory synthesis using hierarchical reference systems. *Proceedings of the VLDB Endowment*, 8(11):1154–1165, 2015.
- [13] X. Hu, M. Yuan, J. Yao, Y. Deng, L. Chen, Q. Yang, H. Guan, and J. Zeng. Differential privacy in telco big data platform. *Proc. VLDB Endow.*, 8(12):1692–1703, Aug. 2015.
- [14] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *VLDB*, pages 275–286, 1998.
- [15] C. Li, M. Hay, and G. Miklau. A data- and workload-aware algorithm for range queries under differential privacy. *PVLDB*, 2014.
- [16] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, pages 123–134, 2010.
- [17] C. Li and G. Miklau. An adaptive mechanism for accurate query answering under differential privacy. *PVLDB*, 5(6):514–525, 2012.
- [18] C. Li, G. Miklau, M. Hay, A. McGregor, and V. Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB Journal*, pages 1–25, 2015.
- [19] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30, 2009.
- [20] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from <http://crawdad.org/epfl/mobility/20090224>, Feb. 2009.
- [21] W. Qardaji, W. Yang, and N. Li. Differentially private grids for geospatial data. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 757–768. IEEE, 2013.
- [22] W. Qardaji, W. Yang, and N. Li. Understanding hierarchical methods for differentially private histograms. *PVLDB*, 6(14), 2013.
- [23] P. A. Sandercock, M. Niewada, A. Czlonkowska, et al. The international stroke trial database. *Trials*, 12(1):1–7, 2011.
- [24] X. Xiao, G. Bender, M. Hay, and J. Gehrke. ireduct: Differential privacy with reduced relative errors. In *SIGMOD*, 2011.
- [25] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, pages 225–236, 2010.
- [26] Y. Xiao, L. Xiong, L. Fan, S. Goryczka, and H. Li. DPCube: Differentially private histogram release through multidimensional partitioning. *Transactions of Data Privacy*, 7(3), 2014.
- [27] J. Xu, Z. Zhang, X. Xiao, Y. Yang, G. Yu, and M. Winslett. Differentially private histogram publication. *The VLDB Journal*, pages 1–26, 2013.
- [28] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao. Low-rank mechanism: Optimizing batch queries under differential privacy. *PVLDB*, 5(11):1136–1147, 2012.
- [29] X. Zhang, R. Chen, J. Xu, X. Meng, and Y. Xie. Towards accurate histogram publication under differential privacy. In *ICDM*, 2014.

## APPENDIX

### A. DATASET DESCRIPTIONS

Table 2 provides an overview of all datasets, 1D and 2D, considered in the paper.

**1D Datasets.** The first seven were described in the papers in which they originally appeared [1,10,11,15,27,29]. The eleven new datasets were derived from three original data sources. A single primary attribute was selected for an initial shape distribution (denote by suffix -ALL). In addition, filters on a secondary attribute were applied to the data, resulting in alternative 1D shape distributions, still on the primary attribute.

The BIDS datasets are derived from a Kaggle competition whose goal is to identify online auctions that are placed by robots.<sup>3</sup> The histogram attribute is the IP address of each individual bid. BIDS-FJ and BIDS-FM result from applying distinct filtering conditions on attribute “merchandise”: BIDS-FJ is a histogram on IP address but counting only individuals where “merchandise=jewelry”, BIDS-FM is a histogram on IP address but counting only individuals where “merchandise=mobile”.

The MD-SAL datasets are based on the Maryland salary database of state employees in 2012.<sup>4</sup> The histogram attribute is “YTD-gross-compensation”. MD-SAL-FA is filtered on condition “pay-type=Annually”.

The LC datasets are derived from data published by the “Lending Club” online credit market.<sup>5</sup> It describes loan applications that were rejected. The LC-REQ datasets are based on attribute “Amount Requested” while the LC-DTIR datasets are based on attribute “Debt-To-Income Ratio”. In both cases, additional shapes are generated by applying filters on the “Employment” attribute. LC-REQ-F1 and LC-DTIR-F1 only count records with attribute “Employment” in range [0, 5], LC-REQ-F2 and LC-DTIR-F2 only count records with attribute “Employment” in range (5, 10].

**2D Datasets.** The BJ-CABS datasets and GOWALLA have been used and described in previous papers [12, 21]. The SF-CABS datasets are derived from well-known mobility traces data from San Francisco taxis [20]. These location-based data are represented as latitude longitude pairs. To get more diverse data shape, we divide the two cab trace data into four by using only the start point and end point in a single dataset. BJ-CABS-S and SF-CABS-S contain only the start location of a cab trip, while BJ-CABS-E and SF-CABS-E record only the end locations.

ADULT-2D, MD-SAL-2D and LC-2D are derived from same sources as a subset of data we used in the 1D experiments. ADULT-2D is derived from source of ADULT. This source is also used in [10] with attributes “age” and “hours”, but we use the attributes “capital-gain” and “capital-loss” in order to test on a larger domain. MD-SAL-2D is based on the Maryland salary database using the attributes “Annual Salary” and “Overtime earnings”. LC-2D is derived from the accepted loan data from Lending Club with attributes “Funded Amount” and “Annual Income”.

STROKE is a new dataset derived from the International Stroke Trial database [23], which is one of the largest randomized trial

<sup>3</sup><https://www.kaggle.com/c/facebook-recruiting-iv-human-or-bot/data>

<sup>4</sup><http://data.baltimoresun.com/salaries/state/cy2012/>

<sup>5</sup><https://www.lendingclub.com/info/download-data.action>

conducted in acute stroke for public use. We used the attributes “Age” and “Systolic blood pressure”.

## B. ALGORITHM DESCRIPTIONS

Section 3.1 and Table 1 provide an overview of the algorithms studied; here we describe each algorithm individually. We begin with data-independent approaches, whose expected error rate is the same for all datasets, then describe data-dependent approaches. Baseline algorithms IDENTITY and UNIFORM are already described in Section 3.1.

*Data-Independent Algorithms.* Each of the data-independent approaches we consider can be described as an instance of the matrix mechanism [16, 18] (although a number were developed independently). The central idea is to select a set of linear queries (called the strategy), estimate them privately using the Laplace mechanism, and then use the noisy results to reconstruct answers to the workload queries. The strategy queries, which can be conveniently represented as a matrix, should have lower sensitivity than the workload and should allow for effective reconstruction of the workload.

If the discrete Haar wavelet matrix is selected as the strategy, the result is the PRIVELET method [25] which is based on the insight that any range query can be reconstructed by just a few of the wavelet queries and that the sensitivity of the Haar wavelet grows with  $\prod_{i=1}^k \log_2 n_i$  where  $k$  is the dimensionality (number of attributes) and  $n_i$  is the domain size of the  $i^{\text{th}}$  attribute.

Several approaches have been developed that use a strategy consisting of hierarchically structured range queries. Conceptually the queries can be arranged in a tree. The leaves of the tree consist of individual queries  $x_i$  (for  $i = 1, \dots, n$ ). Each internal node computes the sum of its children; at the root, the sum is therefore equal to the number of records in the database,  $\|\mathbf{x}\|_1$ . The approaches differ in terms of the branching factor and the privacy budget allocation. The H method [11] has a branching factor of  $b$  and uniform budget allocation. The  $H_b$  method [22] uses the domain size to determine the branching factor: specifically, it finds the  $b$  that minimizes the average variance of answering all range queries through summations of the noisy counts in the tree. It also uses a uniform budget allocation. GREEDY H is a subroutine of the DAWA [15] algorithm but can also be used as a stand-alone algorithm. It has a branching factor  $b$  and employs a greedy algorithm to tune the budget allocation to the workload. When used in DAWA, GREEDY H is applied to the partition that DAWA computes; by itself, it is applied directly to  $\mathbf{x}$ .

All of the above techniques reduce error by using a set of strategy queries which are a good match for the workload of interest. H,  $H_b$ , and PRIVELET were all initially designed to answer the set of *all* range queries, but the strategy each approach uses is static – it is not adapted to the particular queries in the workload. GREEDY H is the only technique that modifies the strategy (by choosing weights for hierarchical queries) in response to the input workload. In the table, it is marked as “workload-aware.” While H and  $H_b$  naturally extend to the multi-dimensional setting (quadtrees, octrees, etc.), GREEDY H extends to 2D by applying a Hilbert transform to obtain a 1D representation.

*Data-Dependent Partitioning Algorithms.* Many of the data-dependent approaches use a partitioning step to approximate the input data vector and we discuss them as a group. These algorithms partition the domain into a collection of disjoint buckets whose union span the domain. The partition must be selected using a differentially private algorithm, since it is highly dependent on the input data, and this constitutes the first step for each algo-

rithm in this group. Once a partition is selected, in a second step, they obtain noisy counts only for the buckets. A noisy data vector can be derived from the noisy bucket counts by assuming the data is uniform within each bucket, and this vector is then used to answer queries. Partitioning trades off between two sources of error: the error due to adding noise to the bucket count, which diminishes with increased bucket size, and the error from assuming uniformity within a bucket, which increases when non-uniform regions are lumped together. When the partition step can effectively find regions of the data that are close to uniform, these algorithms can perform well.

By the sequential composition property of differential privacy, any  $\epsilon_1$ -differentially private algorithm for partition selection can be combined with any  $\epsilon_2$ -differentially private algorithm for count estimation and achieve  $\epsilon$ -differential privacy provided that  $\epsilon_1 + \epsilon_2 \leq \epsilon$ . All these algorithms therefore share a parameter that determines how to allocate the privacy budget across these two steps. We will use  $\rho$  to denote the proportion of  $\epsilon$  used to identify the partition. Thus,  $\epsilon_1 = \epsilon\rho$  and  $\epsilon_2 = \epsilon(1 - \rho)$ .

Unlike the first step, the approaches for estimating the bucket counts are typically data-independent. In fact, PHP, AHP, SF, and UGRID use the Laplace mechanism to obtain noisy counts.

The distinguishing features of each algorithm in this group are:

PHP [1] finds a partition by recursively bisecting the domain into subintervals. The midpoint is found using the exponential mechanism with a cost function based on the expected absolute error from the resulting partition. PHP is limited to 1D.

AHP [29] uses the Laplace mechanism to obtain a noisy count for each  $x_i$  for  $i = 1, \dots, n$  and sets noisy counts below a threshold to zero. The counts are then sorted and clustered to form a partition. As was done by [29], the greedy clustering algorithm is used in the experiments. The threshold is determined by parameter  $\eta$ . AHP extends to the multi-dimensional setting.

DAWA [17] uses dynamic programming to compute the least cost partition in a manner similar in spirit to V-optimal histograms [14]. The cost function is the same as the one used in the exponential mechanism of PHP [1]. To ensure differential privacy, noisy costs are used in place of actual costs. Once the partition is found, a hierarchical set of strategy queries are derived using GREEDY H. To operate on 2D data, DAWA applies a Hilbert transformation.

SF [27] aims to find a partition that minimizes the expected sum of squared error. While the aforementioned algorithms choose the size of the partition privately, in SF, the number of buckets is specified as a parameter  $k$ . Xu et al. [27] recommend setting  $k$  as a function of the domain size and we follow those guidelines in our experimental evaluation (see Section 6.5). Given the specified number of buckets, the bucket boundaries are selected privately using the exponential mechanism. Xu et al. [27] propose two variants of the algorithm, one based on the mean and another based on the median; our experiments evaluate the former. The parameters of SF include  $k$ , the number of buckets;  $\rho$ , the ratio for privacy budget allocation; and  $F$ , an upper bound on the count of a bucket (which is necessary for the cost function used in the exponential mechanism to select bucket boundaries). SF is limited to 1D.

DPCUBE [26], which is multi-dimensional, selects the partition by first applying the Laplace mechanism to obtain noisy counts for each cell in the domain, and then running a standard kd-tree on the noisy counts. Once the partition is selected it obtains fresh noisy counts for the partitions and uses inference to average the two sets of counts.

The remaining partitioning algorithms are designed specifically for 2D.

QUADTREE [4] generates a quadtree with fixed height and then

noisily computes the counts of each node and does post-processing to maintain consistency. The height of the tree is a parameter. This algorithm becomes data-dependent if the tree is not high enough (i.e., the leaves contains aggregations of individual cells). Because the partition structure is fixed, this algorithm does not use any privacy budget to select it (i.e.,  $\rho = 0$ ).

HYBRIDTREE [4] is the combination of a kd-tree and the aforementioned quadtree. It uses a differentially private algorithm to build a kd-tree in a top down manner (DPCUBE goes bottom up). This tree extends a few levels and then a fixed quadtree structured is used for the remaining levels until a pre-specified height is reached.

UGRID [21] builds an equi-width partition where the width is chosen in a data-dependent way, based on the scale of the dataset. AGRID [21] builds a two-level hierarchy of partitions. The top level partition produces equi-width buckets, similar to UGRID. Then within each bucket of the top-level partition, a second partition is chosen in a data-adaptive way based on a noisy count of the number of records in the bucket.

*Other Data-Dependent Algorithms.* MWEM [10] is a workload aware algorithm that supports arbitrary workloads of linear queries (not just range queries). It starts with an estimate of the dataset that is completely uniform (based on assumed knowledge of the scale), and updates the estimate iteratively. Each iteration privately selects the workload query with highest error and then updates the estimate using a multiplicative weights update step. The number of iterations is an important parameter whose setting has a significant impact on performance. We propose a method for setting  $T$ , described in Section 6.4, that adheres to our evaluation principles.

EFPA [1] is based on applying the discrete Fourier transform (DFT) to a 1D data vector  $\mathbf{x}$ . It retains the top  $k$  Fourier coefficients, adds noise to them using the Laplace mechanism, and then inverts the transformation. By choosing  $k < n$ , the magnitude of the noise is lowered at the potential cost of introducing approximation error. The value of  $k$  is chosen in a data-dependent way using the exponential mechanism with a cost function that is based on expected squared error. The privacy budget is divided evenly selecting  $k$  and measuring the  $k$  coefficients.

## C. THEORETICAL ANALYSIS

In this section, we theoretically analyze the properties of scale-epsilon exchangeability (Definition 4) and consistency (Definition 5) for the algorithms considered in the paper.

**THEOREM 1.** *Any instance of the Matrix Mechanism satisfies consistency and scale-epsilon exchangeability.*

**PROOF.** Given a  $n \times k$  workload matrix  $\mathbf{W}$ , let  $\mathbf{S}$  be a  $p \times k$  strategy matrix, and  $\mathbf{S}^+$  denote its Moore-Penrose pseudoinverse, such that  $\mathbf{W}\mathbf{S}^+ = \mathbf{W}$ . The matrix mechanism is given by the following:

$$\mathcal{M}_{\mathbf{S}}(\mathbf{W}, \mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{W}\mathbf{S}^+ \text{Lap}(\Delta_{\mathbf{S}}/\epsilon)^p \quad (1)$$

where,  $\text{Lap}(\lambda)^p$  denotes  $p$  independent random variables drawn from the Laplace distribution with scale  $\lambda$ , and  $\Delta_{\mathbf{S}}$  is the sensitivity of the strategy workload.

The expected scaled per query error of this mechanism on a database  $\mathbf{x}$  is given by:

$$\frac{E\|\mathbf{W}\mathbf{S}^+ \text{Lap}(\Delta_{\mathbf{S}}/\epsilon)^p\|_2}{\|\mathbf{x}\|_1 \cdot k} = \frac{\|\mathbf{W}\mathbf{S}^+ (\frac{\Delta_{\mathbf{S}}}{\epsilon} \mathbf{1}_{p \times 1})\|_2}{\|\mathbf{x}\|_1 \cdot k} \propto \frac{1}{\epsilon \times \|\mathbf{x}\|_1} \quad (2)$$

where  $\mathbf{1}_{p \times 1}$  is a column vector of  $p$  ones. This completes the proof.

Matrix mechanism algorithms are consistent because the noise is additive and at high values of  $\epsilon$  or high scales, the scaled error tends to 0.  $\square$

**COROLLARY 1.** *IDENTITY, PRIVELET, H, H<sub>b</sub>, GREEDY H ensure both consistency and scale-epsilon exchangeability.*

**PROOF.** These techniques are instances of Matrix Mechanism and the proof follows from Theorem 1.  $\square$

**LEMMA 1.** *When  $\epsilon$  goes to infinity, Exponential Mechanism picks one of the items with the highest score with probability 1.*

**PROOF.** Suppose  $T$  is the set of all items,  $T^*$  contains all the items with the highest score  $\alpha$ . Since Exponential Mechanism picks any item  $t$  with the probability proportional to  $e^{\epsilon \text{score}(t)}$ , the probability of picking any item in  $T^*$  is

$$\begin{aligned} P(T^*) &= \frac{\sum_{t \in T^*} e^{\epsilon \alpha}}{\sum_{t \in T} e^{\epsilon \text{score}(t)}} = \frac{\sum_{t \in T^*} e^{\epsilon \alpha}}{\sum_{t \in T^*} e^{\epsilon \alpha} + \sum_{t \in T - T^*} e^{\epsilon \text{score}(t)}} \\ &= \frac{|T^*|}{|T^*| + \sum_{t \in T - T^*} e^{\epsilon(\text{score}(t) - \alpha)}} = 1 \text{ (as } \epsilon \rightarrow \infty) \end{aligned}$$

$\square$

**THEOREM 2.** *EFPA ensures consistency.*

**PROOF.** EFPA works in two steps: (a) pick  $k < n$  coefficients, and (b) add noise to the chosen coefficients. The number of coefficients  $k$  is chosen by Exponential Mechanism with the score function of expected noise added to the  $k$  chosen coefficients ( $\frac{2k}{\epsilon}$ ) plus the error due to dropping  $n - k$  Fourier coefficients. When  $\epsilon$  goes to infinity, the noise term goes to zero. Thus  $k = n$  (where all the coefficients are chosen) receives the highest score. By Lemma 1, this will be chosen as  $\epsilon$  tends to infinity. Therefore, the scaled error from EFPA will tend to zero when  $\epsilon$  tends to infinity, which satisfies consistency.  $\square$

**THEOREM 3.** *AHP, DAWA and DPCUBE are consistent.*

**PROOF.** All AHP, DAWA and DPCUBE noisily partition the domain and then estimate partition counts. When  $\epsilon$  tends to infinity, these algorithms will use a partitioning with zero bias. Also, partition counts are estimated using Laplace Mechanism whose scaled error goes to zero as  $\epsilon$  or scale tends to infinity. Thus AHP, DAWA and DPCUBE ensure consistency.  $\square$

**THEOREM 4.** *AGRID and UGRID ensure consistency.*

**PROOF.** When  $\epsilon$  goes to infinity, both AGRID and UGRID are equal to IDENTITY having grids with size 1. Thus AGRID and UGRID are consistent.  $\square$

**THEOREM 5.** *On sufficiently large domains, QUADTREE and HYBRIDTREE do not ensure consistency.*

**PROOF.** Both QUADTREE and HYBRIDTREE set the maximum height of the tree. If the input dataset has large domain size, the leaf node in the tree contains more than one cell in the domain introducing bias. Therefore, QUADTREE and HYBRIDTREE are not consistent.  $\square$

**THEOREM 6.** *PHP does not ensure consistency.*

**PROOF.** PHP recursively bisects the domain but sets the maximum iterations to be  $\log_2 n$  ( $n$  is the domain size). After  $\log_2 n$  iterations, it is possible that there exist clusters with bias greater than zero. **Example:** Let the scale  $m = 2^n - 1$  and  $X = \{x_1, \dots, x_n\}$  be the dataset with  $x_i = 2^{n-i}$ . In iteration  $j$  ( $\epsilon = \infty$ ), PHP will divide the domain into  $\{x_j\}$  and  $\{x_{j+1}, \dots, x_n\}$ . After  $\log_2 n$  iterations, the last partition will contain domain elements with different counts. Thus the bias of the partition does not go to zero even when  $\epsilon$  tends to infinity.  $\square$

The consistency of SF depends on the implementation. Our experiments use the implementation provided by the authors that includes the modification described in Sec. 6.2 of [27].

**THEOREM 7.** *StructureFirst does not satisfy consistency. If the modification described in Sec. 6.2 of [27] is applied, then it does satisfy consistency.*

**PROOF.** StructureFirst fixes the number of clusters first (it sets the number  $k$  to be  $\frac{n}{10}$  empirically,  $n$  is the domain size), which will introduce bias if the real number of clusters are greater than  $k$ . **Example:** Let  $X = \{x_1, \dots, x_n\}$  be the dataset with  $x_i = i$ . If we set the number of clusters  $k < n$  (suppose  $k = \frac{n}{10}$ ), there will be one cluster  $c$  containing more than one domain point. Since all the  $x_i$  are different, the bias of  $c$  does not go to zero even if  $\epsilon$  tends to infinity. The modification proposed in [27] builds a hierarchical histogram (i.e., H) within each bin. This makes the algorithm consistent for the same reason that H is consistent.  $\square$

**THEOREM 8.** *MWEM does not ensure consistency.*

**PROOF.** MWEM fixes the number of iterations (it sets  $T = 10$  empirically), which will introduce bias if  $T$  queries are not enough to update the estimate. **Example:** Let  $X = \{x_1, \dots, x_n\}$  be the dataset with  $x_i = i$  and workload  $W$  contains all the single count queries. If the number of iterations  $T$  is less than  $n$  (suppose  $T = 10 < n$ ), we can only pick  $T$  single count queries and correct  $T$  domain point when  $\epsilon$  goes to infinity. The bias generated from other domain points will not go to zero even if  $\epsilon$  tends to infinity.  $\square$

**LEMMA 2.** *Laplace Mechanism is scale-epsilon exchangeable. Exponential Mechanism is scale-epsilon exchangeable when the score is a linear function of scale.*

**PROOF.** Laplace Mechanism ensures scale-epsilon exchangeability since the scaled expected  $L_2$  error for Laplace Mechanism is  $\frac{\Delta}{\epsilon \|\mathbf{x}\|_1}$  ( $\Delta$  is the fixed sensitivity given workloads).

Exponential Mechanism picks any item  $t$  with the probability proportional to  $e^{\text{score}(t)}$ . When the score is a linear function of scale, Exponential Mechanism is scale-epsilon exchangeable.  $\square$

**THEOREM 9.** *PHP, MWEM, EFPA are scale-epsilon exchangeable.*

**PROOF.** All these three methods use Exponential Mechanism to choose clustering or parameter. Since their score functions are all linear functions of scale, based on Lemma 2, they will choose the same clustering or parameter with the same probability when  $\epsilon * \text{scale}$  is fixed. Once they use the same clustering or parameter, the scaled error is the sum of the bias and the variance. The variance comes from Laplace Mechanism which is scale-epsilon exchangeable by Lemma 2. It is also easy to check those bias remain the same when fixing  $\epsilon * \text{scale}$ . Thus PHP, MWEM, EFPA ensure scale-epsilon exchangeability.  $\square$

**THEOREM 10.** *StructureFirst does not satisfy scale-epsilon exchangeability.*

**PROOF.** The score function used in StructureFirst is a linear function of the square of scale. Based on Lemma 2, even if  $\epsilon * \text{scale}$  is fixed, the probability of picking the same clustering will be different for different  $\epsilon$  settings. Once the clustering is different, the bias will be different. **Example:** Suppose we use  $\epsilon = 1$  on  $D_1 =$

$\{1, 2\}$  and  $\epsilon = 0.5$  on  $D_2 = \{2, 4\}$ . For both  $D_1$  and  $D_2$ , there are only two clusterings  $C_1 = \{(x_1, x_2)\}$ ,  $C_2 = \{(x_1), (x_2)\}$ . For  $D_1$ ,  $P_1(C_1) = \frac{e^{\text{score}(C_1)}}{e^{\text{score}(C_1)} + e^{\text{score}(C_2)}} = \frac{e^{1.5}}{e^{1.5} + e^2}$ . For  $D_2$ ,  $P_2(C_1) = \frac{e^3}{e^3 + e^4}$ . We can see  $P_1(C_1) \neq P_2(C_2)$ .  $\square$

**THEOREM 11.** *DAWA ensures scale-epsilon exchangeability.*

**PROOF.** Let  $D_1 = m_1 \cdot \mathbf{p}$  and  $D_2 = m_2 \cdot \mathbf{p}$  be two datasets with the same shape  $\mathbf{p}$ , but different scales  $m_1$  and  $m_2$ . Suppose we run DAWA on  $D_1$  with privacy budget  $\epsilon_1$  and  $D_2$  with privacy budget  $\epsilon_2$  such that  $m_1 \epsilon_1 = m_2 \epsilon_2$ . We show that the scaled expected error of the synthetic histogram output by DAWA ( $\hat{\mathbf{x}}$ ) will be the same in both cases. The proof is in two steps:

1. In both cases, the probability that DAWA picks any partitioning  $\mathcal{P}$  is identical.
2. Therefore, the scaled expected error of the noisy histogram  $\hat{\mathbf{x}}$  is the same in both cases.

To prove the first statement, consider all partitionings considered by DAWA. Every partitioning  $\mathcal{P}$  is associated with a score

$$\sum_{C \in \mathcal{P}} \left( \sum_{v \in C} |\mathbf{x}[v] - \frac{\mathbf{x}[C]}{|C|}| + \frac{1}{\epsilon} \right) \quad (3)$$

where  $\mathbf{x}[v]$  is the count of domain value  $v$ , and  $\mathbf{x}[C]$  is the total count of values in  $C$ . In fact, we can rewrite Eq 3 as

$$\|\mathbf{x}\|_1 \sum_{C \in \mathcal{P}} \left( \sum_{v \in C} |\mathbf{p}[v] - \frac{\mathbf{p}[C]}{|C|}| + \frac{1}{\|\mathbf{x}\|_1 \cdot \epsilon} \right) \quad (4)$$

DAWA uses the exponential mechanism to pick the best partitioning. Thus,  $\mathcal{P}$  will be picked with a probability proportional to:

$$\exp \left( \frac{\epsilon \cdot \|\mathbf{x}\|_1}{2\Delta} \sum_{C \in \mathcal{P}} \left( \sum_{v \in C} |\mathbf{p}[v] - \frac{\mathbf{p}[C]}{|C|}| + \frac{1}{\|\mathbf{x}\|_1 \cdot \epsilon} \right) \right) \quad (5)$$

Applying DAWA with budget  $\epsilon_1$  on  $D_1 = m_1 \mathbf{p}$  and with budget  $\epsilon_2$  on  $D_2 = m_2 \mathbf{p}$  will result in exactly the same probability distribution over partitionings.

Furthermore, the second step of DAWA adds noise to all the counts  $\mathbf{x}[C]$ , for  $C \in \mathcal{P}$ . If the partitioning in both datasets  $D_1$  and  $D_2$  is the same, then the resulting noisy histograms will have the same scaled expected error due to the scale epsilon exchangeability of the Laplace mechanism. The proof is complete since we showed that the partitionings have the same probability distribution in  $D_1$  (with  $\epsilon_1$ ) and  $D_2$  (with  $\epsilon_2$ ).  $\square$

**THEOREM 12.** *AHP ensures scale-epsilon exchangeability.*

**PROOF.** The proof is very similar to that of Theorem 11.  $\square$

**THEOREM 13.** *AGRID, UGRID, QUADTREE and HYBRIDTREE are scale-epsilon exchangeable.*

**PROOF.** The partitioning or the tree generation of these four algorithms are independent with  $m \cdot \epsilon$ , where  $m$  is the scale of the input dataset and  $\epsilon$  is the privacy budget. After the partitioning or tree generation, each algorithm applies Laplace Mechanism to answer node queries. Laplace Mechanism is scale-epsilon exchangeable. Thus, AGRID, UGRID, QUADTREE and HYBRIDTREE ensure scale-epsilon exchangeability.  $\square$