Differentially Private Data Cubes: Optimizing Noise Sources and Consistency

Bolin Ding[1] Marianne Winslett[2,1] Jiawei Han[1] Zhenhui Li[1]
[1]Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
[2]Advanced Digital Sciences Center, Singapore
{bding3, winslett, hanj, zli28}@uiuc.edu

ABSTRACT

Data cubes play an essential role in data analysis and decision support. In a data cube, data from a *fact table* is aggregated on subsets of the table's dimensions, forming a collection of smaller tables called *cuboids*. When the fact table includes sensitive data such as salary or diagnosis, publishing even a subset of its cuboids may compromise individuals' privacy. In this paper, we address this problem using *differential privacy* (DP), which provides provable privacy guarantees for individuals by adding noise to query answers. We choose an initial subset of cuboids to compute directly from the fact table, injecting DP noise as usual; and then compute the remaining cuboids from the initial set. Given a fixed privacy guarantee, we show that it is NP-hard to choose the initial set of cuboids so that the maximal noise over all published cuboids is minimized, or so that the number of cuboids with noise below a given threshold (*precise cuboids*) is maximized. We provide an efficient procedure with running time polynomial in the number of cuboids to select the initial set of cuboids, such that the maximal noise in all published cuboids will be within a factor $(\ln|\mathcal{L}|+1)^2$ of the optimal, where $|\mathcal{L}|$ is the number of cuboids to be published, or the number of precise cuboids will be within a factor $(1-1/e)$ of the optimal. We also show how to enforce consistency in the published cuboids while simultaneously improving their utility (reducing error). In an empirical evaluation on real and synthetic data, we report the amounts of error of different publishing algorithms, and show that our approaches outperform baselines significantly.

Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Data mining*; H.2.7 [**DATABASE MANAGEMENT**]: Database Administration—*Security, integrity, and protection*

General Terms

Algorithms, Security, Theory

Keywords

OLAP, data cube, differential privacy, private data analysis

1. INTRODUCTION

Data cubes play an essential role in multidimensional data analysis and fast OLAP operations. Often the underlying data is sensitive, and publishing all or part of the cube may endanger the privacy of individuals. For example, privacy concerns prevent Singapore's Ministry of Health (MOH) from performing wide-scale monitoring for adverse drug reactions among Singapore's three main ethnic groups, none of which are typically included in pharmaceutical companies' drug trials. Privacy concerns also limit MOH's published health summary tables to extremely coarse categories, reducing their utility for policy planning. Institutional Review Board (IRB) approval is now required for access to most high-level summary tables from studies funded by the US National Institutes of Health, making it very hard to leverage results from past studies to plan new studies. In these and many other scenarios, society can greatly benefit from the publication of detailed, high-utility data cubes that also preserve individuals' privacy.

The data cube of a fact table consists of *cells* and *cuboids*. A cell aggregates the rows in the fact table that match on certain dimensions. The fact table in Figure 1(a) has three dimensions, to be aggregated with count measure. As in Figures 1(b)-1(d), a cuboid can be viewed as the projection of a fact table on a subset of dimensions, producing a set of cells with associated aggregate measures.

With background knowledge, an adversary can infer sensitive information about an individual from a published cube [16, 21, 35]. For example, in the data cube in Figure 1, if we know that Alice is aged 31-40 and is in the table, the count in cuboid $\{\mathsf{Age}, \mathsf{Salary}\}$ tells us her salary is 50-200k. If we know Bob is in the table and is aged 21-30, we learn there is a 75% chance that his salary is 10-50k and a 25% chance it is 50-200k. If we also know that Carl, aged 21-30 and with salary 50-200k, is in the table, then the values of count in cuboid $\{\mathsf{Age}, \mathsf{Salary}\}$ tell us Bob's salary is 10-50k.

Even publishing large actual aggregate counts is still not safe, if an adversary has enough background knowledge. For example, suppose there are 100 individuals in a fact table (Sex, Age, Salary), and we publish two cells $(*, *,$ 10-50k) and $(*, *,$ 50-200k), both with count equal to 50. Suppose the adversary knows everyone's salary except Bob's: if 49 people have salary 10-50k and 50 have 50-200k, then s/he can infer that Bob's salary is 10-50k.

We apply the notion of ϵ-*differential privacy* [10] (DP or ϵ-DP for short in the rest of this paper) in data cube publishing. Compared to previous techniques for privacy-preserving data publishing (see [2, 15] for surveys), DP makes very conservative assumptions about the adversary's background knowledge. The privacy guarantee it provides is independent of any background knowledge the adversary may have about the database. In particular, a publishing algorithm satisfying DP protects the privacy of any individual row in the database even if the adversary knows every other row [10].

Sex	Age	Salary
F	21-30	10-50k
F	21-30	10-50k
F	31-40	50-200k
F	41-50	500k+
M	21-30	10-50k
M	21-30	50-200k
M	31-40	50-200k
M	60+	500k+

(a) Fact Table T

Sex	Age	Salary	c
$*$	$*$	0-10k	0
$*$	$*$	10-50k	3
$*$	$*$	50-200k	3
$*$	$*$	...	...

(b) Cuboid {Salary}

Sex	Age	Salary	c
F	21-30	0-10k	0
F	21-30	10-50k	2
...	...	...	...

(c) Cuboid {Sex, Age, Salary}

Sex	Age	Salary	c
$*$	21-30	0-10k	0
$*$	21-30	10-50k	3
$*$	21-30	50-200k	1
$*$	21-30	200-500k	0
$*$	21-30	500k+	0
$*$	31-40	0-10k	0
$*$	31-40	10-50k	0
$*$	31-40	50-200k	2
$*$	31-40	200-500k	0
$*$	31-40	500k+	0
$*$	...	...	...

(d) Cuboid {Age, Salary}

Figure 1: Fact Table and count Data Cube

Informally, DP guarantees that the presence/absence or specific value of any particular individual's record has a negligible impact on the likelihood that a particular result is returned to a query. Thus an adversary cannot make meaningful inferences about any one individual's record values, or even whether the record was present.

One way to achieve DP is to add random noise to query results [10]. The noise is carefully calibrated to the query's *sensitivity*, which measures the total change of the query output under a small change of the input. As the variance of the noise increases, the privacy guarantee becomes stronger, but the utility of the result drops.

To publish an ϵ-DP data cube over d dimensions, there are two baselines. (i) We can compute the count measure for each cuboid from the fact table, and then add noise to each cell. As changing one row in the table affects 2^d cells, according to [12], each cell needs Laplace noise $\mathrm{Lap}(2^d/\epsilon)$, which destroys the utility of the cube unless d is small. The same idea is applied in [4] to publish a set of marginals of a contingency table. (ii) If we only compute count for the *base cuboid* ({Sex, Age, Salary} in Figure 1(c)) from the fact table, $\mathrm{Lap}(1/\epsilon)$ suffices. We compute the other cuboids from the noisy base cuboid to ensure DP. However, noise in high-level cuboids, such as {Salary}, will be magnified significantly, and utility will be low. This idea can be applied to universal histograms, but noise accumulation also makes them ineffective there [19].

Another possible way is to treat each cell in a cuboid as a query, and apply methods in [23] to answer a workload of count queries while ensuring DP. But this approach is not practical in our context, as its running time/space is at least quadratic in the number of cells.

If we roll up measures across DP cuboids, the sums may not match the totals recorded in other cuboids. According to a Microsoft user study [1], users are likely to accept these kinds of small inconsistencies if they trust the original data and understand why the inconsistencies are present. However, if the users do not trust the original data, they may interpret the inconsistencies as evidence of bad data. So it is desirable to enforce a requirement for correct roll-up totals across dimensions in the DP cuboids to be published. Consistency also boosts accuracy of DP data publishing in some cases, e.g., in answering one-dimensional range queries [19].

Contributions. We study how to *publish all or part of a data cube for a given fact table, while ensuring ϵ-differential privacy and limiting the variance of the noise added to the cube.* We propose a general noise control framework in which a subset $\mathcal{L}_{\mathsf{pre}}$ of cuboids is computed from the fact table, plus random noise. The remaining cuboids are computed directly from those in $\mathcal{L}_{\mathsf{pre}}$, which is the "source of noise". When $\mathcal{L}_{\mathsf{pre}}$ is larger, each of its members requires more noise, but the cuboids computed from $\mathcal{L}_{\mathsf{pre}}$ accumulate less noise. So a clever selection of $\mathcal{L}_{\mathsf{pre}}$ can reduce the overall noise.

We define two publishing scenarios that fit the needs of the Ministry of Health. In the first scenario, MOH identifies a set of cuboids which must be released, and the goal is to minimize the max noise in the cuboids. In the second scenario, MOH has a large body of cross-tabulations that can be useful for urban planners and the medical community. A weighting function indicates the importance of releasing each cuboid. The question is, which of these cuboids can be released in a DP manner, while respecting a given noise variance bound for each cell (called *precise cuboids*)–the goal is to maximize the sum of the weights of the released precise cuboids.

We formalize these two optimization problems for the selection of $\mathcal{L}_{\mathsf{pre}}$, prove that they are NP-hard, and give efficient algorithms with provable approximation guarantees. For the first problem, the max noise variance in all published cuboids will be within a factor $(\ln|\mathcal{L}|+1)^2$ of optimal, where $|\mathcal{L}|$ is the number of cuboids to be published; and for the second, the number/weight of precise cuboids will be within a factor $(1-1/e)$ of optimal.

We also show how to enforce consistency over a DP data cube by computing a *consistent measure* from the *noisy measure* released by a DP algorithm. We minimize the L^p distance between the consistent measure and the noisy measure. The revised data cube is still DP, as we do not revisit the fact table when enforcing consistency. The consistency-enforcing techniques in [4] are similar to our L^∞ version, but our L^1 version yields a much better theoretical bound on error than the L^∞ version. We show that in the L^2 version, the consistent measure can be computed efficiently, and provide better utility than the original inconsistent noisy measure.

Organization. Section 2 provides background for data cubes and DP, plus our noise-control goals. Section 3 gives our noise-control publishing framework and formalizes the optimization problems for noise control. Section 4 gives approximation algorithms for these problems. Section 5 shows how to enforce consistency across cuboids. Experimental results are reported in Section 6, followed by discussion and extension of our techniques in Section 7, and related work in Section 8. Proofs of all theorems are in the Appendix.

2. BACKGROUND AND PROBLEM

Data Cubes

Consider a *fact table* T with d *nominal dimensions* $\mathcal{A}=\{A_1, A_2, \dots, A_d\}$. We also use A_i to denote the set of all possible values for the i^{th} dimension, and $|A_i|$ to denote the number of distinct values (i.e., *cardinality*). For a row $r \in T$, $r[i]$ is r's value for A_i.

The *data cube* of T consists of *cells* and *cuboids*. A cell a takes the form $(a[1], a[2], \dots, a[d])$, where $a[i] \in (A_i \cup \{*\})$ denotes the i^{th} dimension's value for this cell. A cell is associated with certain aggregate measure of interest. In this paper, we first focus on the count *measure* $\mathsf{c}(a)$ and discuss how to handle other measures in Section 7.3. $\mathsf{c}(a)$ is the number of rows r in T that are aggregated in cell a (with the same values on non-$*$ dimensions of a): $\mathsf{c}(a) = |\{r \in T \mid \forall 1 \le i \le d,\ r[i] = a[i] \vee a[i] = *\}|$.

Cell a is an *m-dim cell* if exactly m of $a[1], a[2], \dots, a[d]$ are *not* $*$. An *m-dim cuboid C* is specified by m dimensions $[C] = \{A_{i_1}, A_{i_2}, \dots, A_{i_m}\}$. The cuboid C consists of all m-dim cells a such that $\forall 1 \le k \le m$, $a[i_k] \in A_{i_k}$, and C can be interpreted as the projection of T on the set of dimensions $[C]$. The d-dim cuboid is called the *base cuboid* and the cells in it are *base cells*.

For two cuboids C_1 and C_2, if $[C_1] \subseteq [C_2]$ (denoted as $C_1 \preceq C_2$), then (measures of cells in) C_1 can be computed from C_2. C_1 is said to be an *ancestor* of C_2, and C_2 is a *descendant* of C_1. Let $\mathcal{L}_{\mathsf{all}}$ denote the set of all cuboids. Clearly, $(\mathcal{L}_{\mathsf{all}}, \preceq)$ forms a *lattice*.

EXAMPLE 2.1. *Fact table T in Table 1 has three dimensions:* Sex = {M, F}*;* Age = {0-10, 11-20, 21-30, 31-40, 41-50, 51-60, 60+}*; and* Salary = {0-10k, 10-50k, 50-200k, 200-500k, 500k+}*. Figure 2(a) shows the lattice of cuboids of T. As* {Salary} $\subseteq$ {Age, Salary}*, cuboid* {Salary} *can be computed from cuboid* {Age, Salary}*. For example, to compute cell* $(*, *,$ 10-50k$)$*, we aggregate cells* $(*.$ 0-10, 10-50k$)$, …, $(*.$ 60+, 10-50k$)$.

ϵ-*Differential Privacy*

Differential privacy (*DP* for short in the rest of this paper) is based on the concept of *neighbors*. Two tables T_1 and T_2 are neighbors if they differ by at most one row, i.e., $|(T_1 - T_2) \cup (T_2 - T_1)| = 1$.[1] Let $\mathrm{nbrs}(T)$ be the neighbors of T, and $\mathbb{TB}$ be the set of all possible table instances with dimensions $A_1, \ldots, A_d$. Let $F : \mathbb{TB} \to \mathbb{R}^n$ be a function that produces a vector of length n from a table instance. In our context, F computes the set of cuboids we select.

Definition 1. (*Differential Privacy* [10]) A randomized algorithm $\mathcal{K}$ is ϵ-differentially private if for any two neighboring tables T_1 and T_2 and any subset S of the output of $\mathcal{K}$,

$$\Pr[\mathcal{K}(T_1) \in S] \le \exp(\epsilon) \times \Pr[\mathcal{K}(T_2) \in S],$$

where the probability is taken over the randomness of $\mathcal{K}$.

Consider an individual's concern about the privacy of her/his record r. When the publisher specifies a small ϵ, Definition 1 ensures that the inclusion or exclusion of r in the fact table makes a negligible difference in the chances of $\mathcal{K}$ returning any particular answer.

Definition 2. (*Sensitivity* [10]) The L_1 sensitivity of F is:

$$S(F) = \max_{\forall T_1, T_2 \in \mathbb{TB} \,:\, T_1 \in \mathrm{nbrs}(T_2)} ||F(T_1) - F(T_2)||_1,$$

where $||x - y||_1 = \sum_{1 \le i \le n} |x_i - y_i|$ is the L^1 distance between two n-dimensional vectors $x = \langle x_1, \ldots, x_n \rangle$ and $y = \langle y_1, \ldots, y_n \rangle$.

Let $\mathrm{Lap}(\lambda)$ denote a sample Y taken from a zero-mean Laplace distribution with probability density function $f(x) = \frac{1}{2\lambda} e^{-|x|/\lambda}$. Here, $\mathrm{E}[Y] = 0$ and variance $\mathrm{Var}[Y] = 2\lambda^2$. We write $\langle \mathrm{Lap}(\lambda) \rangle^n$ to denote a vector of n independent random samples $\mathrm{Lap}(\lambda)$.

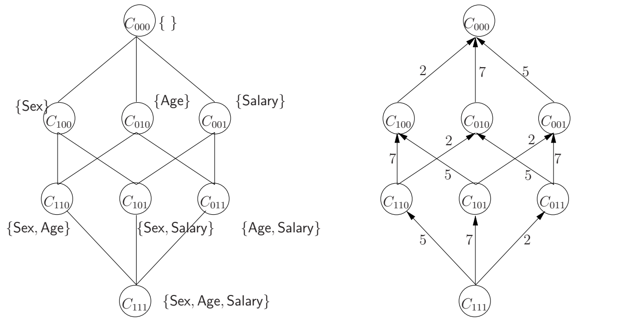
THEOREM 1. ([12]) *Let F be a query sequence of length n. The randomized algorithm that takes as input database T and output $\tilde{F}(T) = F(T) + \langle \mathrm{Lap}(S(F)/\epsilon) \rangle^n$ is ϵ-differentially private.*

Problem Description

Given a d-dimensional fact table T, we aim to publish a subset $\mathcal{L}$ of all T's cuboids $\mathcal{L}_{\mathsf{all}}$ with measure $\tilde{\mathsf{c}}(\cdot)$, a noisy version of the count measure $\mathsf{c}(\cdot)$, using an algorithm $\mathcal{K}$ that ensures ϵ-differential privacy. In particular, for any cell a in a cuboid in $\mathcal{L}$, we want to publish a noisy count measure $\tilde{\mathsf{c}}(a)$ using the DP algorithm $\mathcal{K}$.

Measuring Noise. We *measure the utility of an algorithm by the variance of the noisy measure it publishes.* As we apply Laplace mechanism in Theorem 1, we will show noisy measure $\tilde{\mathsf{c}}(\cdot)$ published by our algorithms is unbiased, i.e., the expectation $\mathrm{E}[\tilde{\mathsf{c}}(a)] = \mathsf{c}(a)$. So for one cell, the variance $\mathrm{Var}[\tilde{\mathsf{c}}(a)]$ is equal to the *expected squared error*, i.e., $\mathrm{Var}[\tilde{\mathsf{c}}(a)] = \mathrm{E}\left[(\tilde{\mathsf{c}}(a) - \mathsf{c}(a))^2\right]$, and we use it to measure the noise/error in $\tilde{\mathsf{c}}(a)$. Similarly, for any set P of cells, we use $\mathrm{Var}\left[\sum_{a \in P} \tilde{\mathsf{c}}(a)\right]$ to measure the noise/error.

[1]Some DP papers use a slightly different definition: T_1 and T_2 are neighbors if they have the same cardinality and T_1 can be obtained from T_2 by replacing one row (ϵ-indistinguishable [12]). Our algorithms also work with this definition, if we double the noise.



(a) Lattice Structure of Cuboids (b) Variance Magnification

Figure 2: Lattice of cuboids, Variance magnification of noise

Noise-Control Objectives. We aim to control the noise in the published cuboids in one of the following two ways:

(i) (*Bounding max noise*) Minimize the maximal variance over all cells, $\max_a \mathrm{Var}[\tilde{\mathsf{c}}(a)]$, in the published cuboids.

(ii) (*Publishing as many as possible*) Given a threshold θ_0, a cuboid C is said to be *precise* if $\mathrm{Var}[\tilde{\mathsf{c}}(a)] \le \theta_0$ for all cells a in C. Maximize the number of precise cuboids in all published ones.

Enforcing Consistency. When the consistency is required, we show how to take noisy measure $\tilde{\mathsf{c}}(\cdot)$ as input and alter it into measure $\hat{\mathsf{c}}(\cdot)$ to fit *consistency constraints* that the cuboids should sum up correctly. Our consistency-enforcing algorithm takes only $\tilde{\mathsf{c}}(\cdot)$ as input (not touching the fact table T), and thus from the composition property of differential privacy [28], it is also ϵ-DP. The output consistent measure $\hat{\mathsf{c}}(\cdot)$ should be as close to $\tilde{\mathsf{c}}(\cdot)$ and $\mathsf{c}(\cdot)$ as possible. The consistency constraints will be formulated in Section 5.

3. NOISE CONTROL FRAMEWORK

We first present two straw man approaches, $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$, followed by our noise-optimizing approach $\mathcal{K}_{\mathsf{part}}$.

Straw Man Release Algorithms $\mathcal{K}_{\mathsf{all}}$ *and* $\mathcal{K}_{\mathsf{base}}$

One option, $\mathcal{K}_{\mathsf{all}}$, is to compute the exact measure $\mathsf{c}(a)$ for each cell a in each cuboid in $\mathcal{L}$, and add noise to each cell independently, including empty cells. Formally, let $F_{\mathsf{all}}(T)$ be the vector containing measure $\mathsf{c}(a)$ for each cell a of each cuboid in $\mathcal{L}$. F_{all} has sensitivity $|\mathcal{L}|$ (Definition 2), since a row in T contributes 1 to exactly one cell in each cuboid in $\mathcal{L}$. $\mathcal{K}_{\mathsf{all}}$ adds noise drawn from $\mathrm{Lap}(|\mathcal{L}|/\epsilon)$ to each cell–$\tilde{\mathsf{c}}(a) = \mathsf{c}(a) + \mathrm{Lap}(|\mathcal{L}|/\epsilon)$–and publishes the resulting vector. The noise variance in $\mathcal{K}_{\mathsf{all}}$ is high. With 8 dimensions, $|\mathcal{L}|$ can reach 2^8, making $\mathrm{Var}[\tilde{\mathsf{c}}(a)] = 2 \cdot 2^{16}/\epsilon^2$. We will discuss the relationship between $\mathcal{K}_{\mathsf{all}}$ and related work [4, 36] in Section 8.

THEOREM 2. $\mathcal{K}_{\mathsf{all}}$ *is ϵ-differentially private. For any cell a to be published,* $\mathrm{E}[\tilde{\mathsf{c}}(a)] = \mathsf{c}(a)$ *and* $\mathrm{Var}[\tilde{\mathsf{c}}(a)] = 2|\mathcal{L}|^2/\epsilon^2$.

Another option, $\mathcal{K}_{\mathsf{base}}$, computes only the cells in the d-dim cuboid, i.e., the base cuboid, directly from T, and adds noise into them. $\mathcal{K}_{\mathsf{base}}$ computes the measures of the remaining cells to be released by aggregating the noisy measures of the base cells. The vector $F_{\mathsf{base}}(T)$ has each entry as the measure $\mathsf{c}(a)$ of a cell in the base cuboid, and thus its sensitivity is 1; therefore, publishing $\tilde{\mathsf{c}}(a) = \mathsf{c}(a) + \mathrm{Lap}(1/\epsilon)$ for each base cell preserves ϵ-differential privacy. When $\mathcal{K}_{\mathsf{base}}$ computes higher-level cuboids from $F_{\mathsf{base}}(T)$, we do not touch T, so ϵ-DP is preserved. However, the noise variance grows as we aggregate more base cells for cells in higher levels.

THEOREM 3. $\mathcal{K}_{\mathsf{base}}$ *is ϵ-differentially private. For any published cell a,* $\mathrm{E}[\tilde{\mathsf{c}}(a)] = \mathsf{c}(a)$. *If a is in a cuboid with dimensions $[C]$, then $\tilde{\mathsf{c}}(a)$ has noise variance* $\mathrm{Var}[\tilde{\mathsf{c}}(a)] = 2\prod_{A_j \notin [C]} |A_j|/\epsilon^2$.

EXAMPLE 3.1. *For the fact table T in Figure 1(a), Figure 2(a) shows the lattice of cuboids under the relationship $\preceq$. Three dimensions* Sex, Age, *and* Salary *have cardinality 2, 7, and 5, respectively. Each cuboid is labeled as $C_{x_1x_2x_3}$, where $x_i = 1$ iff the i^{th} dimension is in this cuboid. For example, C_{011} is the cuboid* $\{\text{Age}, \text{Salary}\}$. *The label on an edge from C to C' in Figure 2(b) is the* variance magnification *ratio when cells in cuboid C are aggregated to compute the cells in C'. For example, if C_{011} is computed from C_{111}, the noise variance doubles, since the dimension* Sex *has 2 distinct values–each cell in C_{011} is the aggregation of 2 cells in C_{111}. If C_{001} is computed from C_{111}, the noise variance is magnified $2 \times 7 = 14$ times, since 14 cells in C_{111} form one in C_{001}.*

Suppose we want to publish all the cuboids in Figure 2(a). Using $\mathcal{K}_{\mathsf{all}}$, we add Laplace noise $\text{Lap}(8/\epsilon)$ to each cell, giving noise variance $2 \times 64/\epsilon^2 = 128/\epsilon^2$ for one cell. Using $\mathcal{K}_{\mathsf{base}}$, we add Laplace noise $\text{Lap}(1/\epsilon)$ to each cell in C_{111} and then aggregate its cells. Each cell in C_{100} is built from 7×5 cells in C_{111}, with noise variance $35 \times 2/\epsilon^2 = 70/\epsilon^2$. A cell in C_{000} is built from $7 \times 5 \times 2$ cells in C_{111}, with noise variance $70 \times 2/\epsilon^2 = 140/\epsilon^2$.

A better approach is to compute cuboids $C_{111}, C_{110}, C_{101}$, and C_{100} from T, adding Laplace noise $\text{Lap}(4/\epsilon)$ to each (the sensitivity is 4). We compute the other cuboids from these. Then the noise variance in C_{100} is $32/\epsilon^2$ and in C_{000} is $2 \times 32/\epsilon^2 = 64/\epsilon^2$ (aggregating 2 cells of C_{100}).

As shown in Example 3.1, the more cuboids we compute directly from the table T, the higher their sensitivity is, and so the more noise they require, but the less noise is accumulated when other cuboids are computed from them. $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$ represent the extremes of computing as many or as few cuboids as possible directly from T. There must be a strategy between $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$ that gives better bounds for the noise variance of released cuboids.

A Better Algorithm $\mathcal{K}_{\mathsf{part}}$

To publish a set $\mathcal{L} \subseteq \mathcal{L}_{\mathsf{all}}$ of cuboids, $\mathcal{K}_{\mathsf{part}}$ chooses which cuboids, denoted as $\mathcal{L}_{\mathsf{pre}}$, to compute directly from the fact table T, in a manner that reduces the overall noise. $\mathcal{L}_{\mathsf{post}} = \mathcal{L} - \mathcal{L}_{\mathsf{pre}}$ includes all the other cuboids in $\mathcal{L}$. We do *not* require $\mathcal{L}_{\mathsf{pre}} \subseteq \mathcal{L}$.

1. (*Noise Sources*) For each cell a of cuboids in $\mathcal{L}_{\mathsf{pre}}$, $\mathcal{K}_{\mathsf{part}}$ computes $\mathsf{c}(a)$ from T and releases $\tilde{\mathsf{c}}(a) = \mathsf{c}(a) + \text{Lap}(s/\epsilon)$, where the sensitivity $s = |\mathcal{L}_{\mathsf{pre}}|$. Note that $\mathcal{L}_{\mathsf{pre}}$ is selected by our algorithms in Section 4.1 s.t. all cuboids in $\mathcal{L}$ can be computed from $\mathcal{L}_{\mathsf{pre}}$.

2. (*Aggregation*) For each cuboid $C \in \mathcal{L}_{\mathsf{post}}$, $\mathcal{K}_{\mathsf{part}}$ selects a descendant cuboid C^* from $\mathcal{L}_{\mathsf{pre}}$ s.t. $C^* \succeq C$, and computes $\tilde{\mathsf{c}}(a)$ for each cell $a \in C$ by aggregating the noisy measure of cells in C^*. We discuss how to pick C^* as follows.

The measure $\tilde{\mathsf{c}}(a)$ output by $\mathcal{K}_{\mathsf{part}}$ is an unbiased estimator of $\mathsf{c}(a)$ for every cell a, i.e., $\text{E}[\tilde{\mathsf{c}}(a)] = \mathsf{c}(a)$. For a cell a in cuboid $C' \in \mathcal{L}_{\mathsf{pre}}$, $\text{Var}[\tilde{\mathsf{c}}(a)] = 2s^2/\epsilon^2$ ($s = |\mathcal{L}_{\mathsf{pre}}|$). Suppose cell a in $C \in \mathcal{L}_{\mathsf{post}}$ is computed from $C' \in \mathcal{L}_{\mathsf{pre}}$ by aggregating on dimensions $[C'] - [C] = \{A_{k_1}, \ldots, A_{k_q}\}$, the *variance magnification* is defined as $\text{mag}(C, C') = \prod_{1 \le i \le q} |A_{k_i}|$. So the noise variance is

$$\text{Var}[\tilde{\mathsf{c}}(a)] = \text{mag}(C, C') \cdot 2s^2/\epsilon^2. \tag{1}$$

Let $\text{mag}(C, C) = 1$. If C cannot be computed from C' ($C \not\preceq C'$), let $\text{mag}(C, C') = \infty$. We should compute the cells in $C \in \mathcal{L}_{\mathsf{post}}$ from the cuboid $C^* \in \mathcal{L}_{\mathsf{pre}}$ for which $\text{mag}(C, C^*)$ is minimal, i.e., $\text{mag}(C, C^*) = \min_{C' \in \mathcal{L}_{\mathsf{pre}}} \text{mag}(C, C')$. Let

$$\text{noise}(C, \mathcal{L}_{\mathsf{pre}}) = \min_{C' \in \mathcal{L}_{\mathsf{pre}}} \text{mag}(C, C') \cdot 2s^2/\epsilon^2 \tag{2}$$

be the smallest possible noise variance when computing C from a single cuboid in $\mathcal{L}_{\mathsf{pre}}$. For $C' \in \mathcal{L}_{\mathsf{pre}}$, $\text{noise}(C', \mathcal{L}_{\mathsf{pre}}) = 2s^2/\epsilon^2$.

THEOREM 4. *Algorithm $\mathcal{K}_{\mathsf{part}}$ is ϵ-differentially private. For any cuboid cell a to be released,* $\text{E}[\tilde{\mathsf{c}}(a)] = \mathsf{c}(a)$.

EXAMPLE 3.2. *Consider the data cube in Example 2.1 and release algorithm $\mathcal{K}_{\mathsf{part}}$ with $\mathcal{L}_{\mathsf{pre}} = \{C_{111}, C_{110}, C_{101}, C_{100}\}$. C_{000} can be computed from C_{100} with noise variance $2 \times 32/\epsilon^2$, since $\text{mag}(C_{000}, C_{100}) = 2$; or from C_{110} with noise variance $14 \times 32/\epsilon^2 = 448/\epsilon^2$, since $\text{mag}(C_{000}, C_{110}) = 2 \times 7 = 14$. So $\mathcal{K}_{\mathsf{part}}$ computes C_{000} from C_{100}, and $\text{noise}(C_{000}, \mathcal{L}_{\mathsf{pre}}) = 64/\epsilon^2$.*

$\mathcal{K}_{\mathsf{part}}$ shares some similarities to the *matrix mechanism* in [23]. $\mathcal{K}_{\mathsf{part}}$ chooses cuboids $\mathcal{L}_{\mathsf{pre}}$ to compute $\mathcal{L}$, while [23] chooses a set of queries to answer a given workload of count queries. If we adapt the matrix mechanism for our problem by treating a cell as a count query, we need to manipulate matrices of sizes equal to the number of cells (e.g., matrices with $10^6 \times 10^6$ entries for a moderate data cube with 10^6 cells). So [23] is not applicable in our context.

Efficient Implementation of $\mathcal{K}_{\mathsf{part}}$. Suppose $\mathcal{L}_{\mathsf{pre}}$ is given. A naive way to compute DP cuboids in $\mathcal{L}$ is to first inject noise into each cuboid in $\mathcal{L}_{\mathsf{pre}}$. Then for each $C \in \mathcal{L}$, query its descendant $C^* \in \mathcal{L}_{\mathsf{pre}}$ ($C \preceq C^*$), and aggregate cells in C^* to compute cells in C.

We can compute DP cuboids in $\mathcal{L}$ more efficiently with fewer cell queries. Suppose $C \preceq C'' \preceq C^*$, noise is injected into $C^* \in \mathcal{L}_{\mathsf{pre}}$ for ensuring DP, and C'' is computed from C^*. Then computing C from C^* is equivalent to computing it from C'', with DP ensured. So after noise is injected into all cuboids in $\mathcal{L}_{\mathsf{pre}}$, DP cuboids in $\mathcal{L}$ can be computed in a level-by-level way, i.e., computing i-dim cuboids from $(i+1)$-dim cuboids. The total running time is $O(Nd^2)$, where $N = \Pi_j(|A_j| + 1)$ is the total number of cells.

Problems of Optimizing Noise Source: Choosing $\mathcal{L}_{\mathsf{pre}}$

We formalize two versions of the problem of choosing $\mathcal{L}_{\mathsf{pre}}$ with different goals, given table T and set $\mathcal{L}$ of cuboids to be published.

Problem 1. (BOUND MAX VARIANCE) Choose a set $\mathcal{L}_{\mathsf{pre}}$ of cuboids s.t. all cuboids in $\mathcal{L}$ can be computed from $\mathcal{L}_{\mathsf{pre}}$ and the max noise $\text{noise}(\mathcal{L}_{\mathsf{pre}}) = \max_{C \in \mathcal{L}} \text{noise}(C, \mathcal{L}_{\mathsf{pre}})$ is minimized.

Problem 2. (PUBLISH MOST) Given threshold θ_0 and cuboid weighting $w(\cdot)$, choose $\mathcal{L}_{\mathsf{pre}}$ s.t. $\sum_{C \in \mathcal{L}:\ \text{noise}(C, \mathcal{L}_{\mathsf{pre}}) \le \theta_0} w(C)$ is maximized. In other words, maximize the weight of the cuboids computed from $\mathcal{L}_{\mathsf{pre}}$ with noise variance no more than θ_0.

THEOREM 5. BOUND MAX VARIANCE *and* PUBLISH MOST *are both NP-hard, where $|\mathcal{L}|$ is the input size.*

The proof uses a *non-trivial* reduction from the VERTEX COVER problem in degree-3 graphs. Details are in the appendix.

Note that $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$ are special cases of $\mathcal{K}_{\mathsf{part}}$ by letting $\mathcal{L}_{\mathsf{pre}} = \mathcal{L}$ and $\mathcal{L}_{\mathsf{pre}} = \{\text{the base cuboid}\}$, respectively. We will introduce two algorithms that choose $\mathcal{L}_{\mathsf{pre}}$ carefully so that $\mathcal{K}_{\mathsf{part}}$ is better than $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$ w.r.t. objectives in Problems 1 and 2.

4. OPTIMIZING THE SOURCE OF NOISE

A brute force approach for Problems 1 and 2–enumerating all possible choices of $\mathcal{L}_{\mathsf{pre}}$ (all possible cuboid subsets)–takes $O(2^{2^d})$ time, which is *not* practical even for $d = 5$. Due to the hardness result in Theorem 5, we introduce approximation algorithms for these two problems, with running time polynomial in 2^d.

4.1 Bounding the Maximum Variance

We now present a $(\ln(|\mathcal{L}|)+1)^2$-approximation algorithm for the BOUND MAX VARIANCE problem, with running time polynomial in $|\mathcal{L}|$ and 2^d. First, suppose we have an efficient algorithm FEASIBLE for subproblem FEASIBILITY($\mathcal{L}, \theta, s$): for a fixed θ and s, is there a set of s cuboids $\mathcal{L}_{\sf pre}$ s.t. for all $C \in \mathcal{L}$, $\mathrm{noise}(C, \mathcal{L}_{\sf pre}) \le \theta$? Let FEASIBLE($\mathcal{L}, \theta, s$) return $\mathcal{L}_{\sf pre}$ if a solution exists, and "NO" otherwise. Then to solve Problem 1, we can find the minimum θ such that for some s, FEASIBLE($\mathcal{L}, \theta, s$) returns a feasible solution. This θ can be found using binary search instead of guessing all possible values. Algorithm 1 provides the details.

```
1: $\theta_L \leftarrow 0$, $\theta_R \leftarrow 2|\mathcal{L}|^2/\epsilon^2$ (or $\theta_R \leftarrow 2 \cdot 4^d/\epsilon^2$ if $|\mathcal{L}| = 2^d$);
2: while $|\theta_L - \theta_R| > 1/\epsilon^2$ do
3:     $\theta \leftarrow (\theta_L + \theta_R)/2$;
4:     if FEASIBLE($\mathcal{L}, \theta, s$) = NO for all $s = 1, \ldots, |\mathcal{L}|$ then $\theta_L \leftarrow \theta$;
       else $\theta_R \leftarrow \theta$;
5: return $\theta^* = \theta_R$ and the solution found by FEASIBLE($\mathcal{L}, \theta^*, s$).
FEASIBLE($\mathcal{L}, \theta, s$)
6: Compute coverage $\mathrm{cov}(C)$ for each cuboid based on $\theta$ and $s$;
7: $\mathcal{R} \leftarrow \varnothing$, $\mathcal{COV} \leftarrow \varnothing$;
8: repeat the following two steps $s$ times
9:     Select a cuboid $C'$ such that $|\mathrm{cov}(C') - \mathcal{COV}|$ is maximized
10:    $\mathcal{R} \leftarrow \mathcal{R} \cup \{C'\}$, $\mathcal{COV} \leftarrow \mathcal{COV} \cup \mathrm{cov}(C')$;
11: if $\mathcal{COV} = \mathcal{L}$ then return $\mathcal{R}$ as $\mathcal{L}_{\sf pre}$;
12: else return NO.
```

Algorithm 1: Algorithm for BOUND MAX VARIANCE problem

Theorem 5 says that BOUND MAX VARIANCE is NP-hard, so there cannot be an efficient exact algorithm for the subproblem FEASIBILITY. We provide a greedy algorithm (analogous to that for SET COVER) that achieves the promised approximation guarantee $(\ln |\mathcal{L}| + 1)^2$ for the BOUND MAX VARIANCE problem.

Using (2), we rewrite FEASIBILITY's noise constraint as:

$$\mathrm{noise}(C, \mathcal{L}_{\sf pre}) \le \theta \Leftrightarrow \min_{C' \in \mathcal{L}_{\sf pre}} \mathrm{mag}(C, C') \le \tfrac{\theta\epsilon^2}{2s^2}. \quad (3)$$

For fixed θ, ϵ, and s, cuboid C' *covers* cuboid C if $C \preceq C'$ and $\mathrm{mag}(C, C') \le \frac{\theta\epsilon^2}{2s^2}$. Define C''s *coverage* to be:

$$\mathrm{cov}(C', \theta, \epsilon, s) = \{C \in \mathcal{L} \mid C \preceq C',\ \mathrm{mag}(C, C') \le \tfrac{\theta\epsilon^2}{2s^2}\}. \quad (4)$$

For simplicity, we write $\mathrm{cov}(C')$ for $\mathrm{cov}(C', \theta, \epsilon, s)$ when θ, ϵ, and s are fixed. The following lemma is from (3) and (4).

LEMMA 1. *$\mathrm{noise}(C, \mathcal{L}_{\sf pre}) \le \theta$ if and only if there exists $C' \in \mathcal{L}_{\sf pre}$ such that $C \in \mathrm{cov}(C', \theta, \epsilon, |\mathcal{L}_{\sf pre}|)$.*

By Lemma 1, FEASIBILITY($\mathcal{L}, \theta, s$) reduces to finding s cuboids that cover all cuboids in $\mathcal{L}$. To solve this problem, we employ the greedy algorithm for the SET COVER problem: in each of s iterations, we add to $\mathcal{L}_{\sf pre}$ the cuboid that covers the maximum number of not-yet-covered members of $\mathcal{L}$. The greedy algorithm can find at most $(\ln |\mathcal{L}| + 1)s^*$ cuboids to cover all cuboids in $\mathcal{L}$ if the minimum covering set has size at least s^*. This algorithm, denoted as FEASIBLE($\mathcal{L}, \theta, s$), appears in lines 6-12 of Algorithm 1.

THEOREM 6. *Algorithm 1 finds an $(\ln |\mathcal{L}| + 1)^2$-approximation to Problem 1 in $O(\min\{3^d,\ 2^d|\mathcal{L}|\}|\mathcal{L}| \log |\mathcal{L}|)$ time. Moreover, using the solution $\mathcal{L}_{\sf pre}$ produced by Algorithm 1, $\mathcal{K}_{\sf part}$ is at least as good as $\mathcal{K}_{\sf all}$ and $\mathcal{K}_{\sf base}$, in terms of the objective in Problem 1.*

Algorithm 1's running time is polynomial in 2^d and $|\mathcal{L}|$, and it provides a logarithmic approximation. Note that parameter d is not very large in most applications, but $|\mathcal{L}|$ can be as large as 2^d.

EXAMPLE 4.1. *Suppose we want to publish all cuboids in Figure 2(a) ($\mathcal{L} = \mathcal{L}_{\sf all}$). When FEASIBLE($\cdot$) is called with $\theta = 80/\epsilon^2$ and $s = 2$, from* (4)*, C' covers C iff $C \preceq C'$ and* $\mathrm{mag}(C, C') \le 10$. $\mathrm{mag}(C, C')$ *can be computed from Figure 2(b) by multiplying the edge weights on the path from C' to C. In the first iteration of lines 8-10, Algorithm 1 chooses C_{111} for* $\mathrm{cov}(C_{111}) = \{C_{111}, C_{110}, C_{010}, C_{101}, C_{011}\}$ *covers the most cuboids in $\mathcal{L}$, and puts C_{111} into $\mathcal{R}$. In the second iteration, Algorithm 1 chooses C_{101} because* $\mathrm{cov}(C_{101}) = \{C_{101}, C_{100}, C_{001}, C_{000}\}$ *which covers three (the most) cuboids not covered by C_{111} yet. C_{111} and C_{101} cover all the cuboids, so* FEASIBLE($\cdot$) *returns $\mathcal{L}_{\sf pre} = \{C_{111}, C_{101}\}$.*

The binary search in Algorithm 1 determines that $64/\epsilon^2$ is the smallest value of θ for which FEASIBLE *finds a feasible $\mathcal{L}_{\sf pre}$ for some s. In that iteration, for $s = 4$,* FEASIBLE *returns $\mathcal{L}_{\sf pre} = \{C_{111}, C_{110}, C_{101}, C_{100}\}$. There, we have* $\mathrm{cov}(C_{111}) = \{C_{111}, C_{011}\}$, $\mathrm{cov}(C_{110}) = \{C_{110}, C_{010}\}$, $\mathrm{cov}(C_{101}) = \{C_{101}, C_{001}\}$, *and* $\mathrm{cov}(C_{100}) = \{C_{100}, C_{000}\}$.

4.2 Publishing as Many as Possible

We now present a $(1 - 1/e)$-approximation algorithm for the PUBLISH MOST problem, with running time polynomial in $|\mathcal{L}|$ and 2^d. Given threshold θ_0, assuming the optimal solution has s cuboids, from Lemma 1, PUBLISH MOST is equivalent to finding a set of s cuboids $\mathcal{L}_{\sf pre}$ s.t. the weighted sum of the cuboids in $\mathcal{L}$ that they cover (with $\theta = \theta_0$ and the fixed s) is as high as possible. We will consider values up to $|\mathcal{L}|$ for s, and apply the greedy algorithm for the MAXIMUM COVERAGE problem for each choice of s.

As outlined in Algorithm 2, initially $\mathcal{R}$ and $\mathcal{COV}$ are empty. In each iteration, we find the cuboid for which the total weight of the not-previously-covered cuboids it covers in $\mathcal{L}$ is maximal. We put this cuboid into $\mathcal{R}$ and put the newly covered cuboids into $\mathcal{COV}$. After repeating this s times, $|\mathcal{R}| = s$ and $\mathcal{COV}$ is the set of cuboids with noise variance no more than θ_0 if computed from $\mathcal{R}$. At the end, pick the best $\mathcal{R}$ over all choices of s as $\mathcal{L}_{\sf pre}$. If some cuboids in $\mathcal{L}$ cannot be computed from $\mathcal{L}_{\sf pre}$ but we desire to publish them, we can simply add one more cuboid, the base cuboid, into $\mathcal{L}_{\sf pre}$.

```
1: for $s = 1, 2, \ldots, |\mathcal{L}|$ do $\mathcal{R}_s \leftarrow$ GREEDYCOVER($\mathcal{L}, \theta_0, s$);
2: return the best $\mathcal{R}_s$ as $\mathcal{L}_{\sf pre}$;
GREEDYCOVER($\mathcal{L}, \theta_0, s$)
3: Compute coverage $\mathrm{cov}(C)$ for each cuboid based on $\theta_0$ and $s$;
4: $\mathcal{R} \leftarrow \varnothing$, $\mathcal{COV} \leftarrow \varnothing$;
5: repeat the following steps $s$ times
6:     Select a cuboid $C'$ such that $|\mathrm{cov}(C') - \mathcal{COV}|$ is maximized
              (or $\sum_{C \in \mathrm{cov}(C') - \mathcal{COV}} w(C)$ is maximized)
7:     $\mathcal{R} \leftarrow \mathcal{R} \cup \{C'\}$, $\mathcal{COV} \leftarrow \mathcal{COV} \cup \mathrm{cov}(C')$;
8: return $\mathcal{R}$.
```

Algorithm 2: Algorithm for PUBLISH MOST

THEOREM 7. *Algorithm 2 finds a $(1 - 1/e)$-approximation to Problem 2 in $O(\min\{3^d,\ 2^d|\mathcal{L}|\}d|\mathcal{L}|)$ time. Moreover, using the solution $\mathcal{L}_{\sf pre}$ produced by Algorithm 2, $\mathcal{K}_{\sf part}$ is as least as good as $\mathcal{K}_{\sf all}$ and $\mathcal{K}_{\sf base}$, in terms of the objective in Problem 2.*

EXAMPLE 4.2. *Consider the effect of Algorithm 2 on Example 4.1, with $\theta_0 = 40/\epsilon^2$ and all cuboids to be published with equal weights. For $s = 2$, when we call* GREEDYCOVER($\mathcal{L}, 40/\epsilon^2, 2$), *$\{C_{111}, C_{101}\}$ is returned as $\mathcal{R}$ in line 8. Since no other value of s covers more cuboids, $\{C_{111}, C_{101}\}$ is finally returned as $\mathcal{L}_{\sf pre}$.*

5. ENFORCING CONSISTENCY

Suppose $\mathcal{L}_{\sf pre} = \{C_1, C_2\}$, where $[C_1] = \{A_1, A_2, A_3\}$ and $[C_2] = \{A_1, A_2, A_4\}$. Consider cuboid $[C] = \{A_1, A_2\}$ to be published. C can be computed from either C_1 or C_2. Since we add noise to C_1 and C_2 independently, the two computations of C will probably yield different results. If we revise $\mathcal{K}_{\sf part}$ by letting C be the weighted (based on variance) average of the two results, then C will not be an exact roll-up of either C_1 or C_2, though it will be unbiased. Such inconsistency occurs in data cubes released by both $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{part}}$, as long as $\mathcal{L}_{\mathsf{pre}}$ contains more than one cuboid.

In this section, we introduce systematic approaches to enforce consistency across released cuboids. The basic idea is as follows. Consider the noisy measures $\tilde{\mathsf{c}}(\cdot)$ of cells in each cuboid $C \in \mathcal{L}_{\mathsf{pre}}$ released by $\mathcal{K}_{\mathsf{part}}$, where $\mathcal{L}_{\mathsf{pre}}$ is selected by either Algorithm 1 or Algorithm 2. Recall that, in $\mathcal{K}_{\mathsf{all}}$, $\mathcal{L}_{\mathsf{pre}}$ is the set of all cuboids to be published. We construct *consistent measure* $\hat{\mathsf{c}}(\cdot)$ from $\tilde{\mathsf{c}}(\cdot)$ so that the consistency constraints are enforced and the distance between $\hat{\mathsf{c}}(\cdot)$ and $\tilde{\mathsf{c}}(\cdot)$ is minimized. Since the algorithm takes only $\tilde{\mathsf{c}}(\cdot)$ as the input and does not touch the real count measure $\mathsf{c}(\cdot)$ or the fact table T, ϵ-DP is automatically preserved.

Consistency Constraints and Distance Measures

Clearly, there is no inconsistency if we compute measures of all cells from the base cells (i.e., d-dim cells). For a cell a, let $\mathrm{Base}(a)$ be the set of all base cells, each of which aggregates a disjoint subset of rows that are contained in a. Formally, $a' \in \mathrm{Base}(a)$ if and only if for any dimension A_i, $a[i] \neq *$ implies $a[i] = a'[i]$. So we enforce *consistency constraints* for measure $\hat{\mathsf{c}}(\cdot)$ as follows.

$$\sum_{a' \in \mathrm{Base}(a)} \hat{\mathsf{c}}(a') = \hat{\mathsf{c}}(a), \ \ \forall \text{ cells } a. \tag{5}$$

We seek the choice of $\hat{\mathsf{c}}(\cdot)$ that satisfies consistency constraints in (5) and is as close as possible to $\tilde{\mathsf{c}}(\cdot)$, but can be computed without reexamining $\mathsf{c}(\cdot)$. We use L^p *distance* ($p \geq 1$) to measure how close $\hat{\mathsf{c}}(\cdot)$ and $\tilde{\mathsf{c}}(\cdot)$ are. Let $\mathcal{E}_{\mathsf{pre}}$ be the set of cells in cuboids belonging to $\mathcal{L}_{\mathsf{pre}}$. Treat $\hat{\mathsf{c}}(\cdot)$ and $\tilde{\mathsf{c}}(\cdot)$ as vectors in $\mathbb{R}^{\mathcal{E}_{\mathsf{pre}}}$, and the L^p distance between is $||\hat{\mathsf{c}}(\cdot) - \tilde{\mathsf{c}}(\cdot)||_p = \sum_{a \in \mathcal{E}_{\mathsf{pre}}} \left(|\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)|^p\right)^{1/p}$.

Finding $\hat{\mathsf{c}}(\cdot)$ to minimize $||\hat{\mathsf{c}}(\cdot) - \tilde{\mathsf{c}}(\cdot)||_p$ subject to (5) can be viewed as a *least-norm problem*. From classic results from convex optimization [7], it can be solved efficiently, at least in theory. We will prove that the utility of optimal solutions for L^1, L^2, and L^∞ distances satisfies certain statistical guarantees. More importantly, classic algorithms do not work in our context because the number of variables involved in (5) is equal to the number of cells, which is huge. We provide a practical and theoretically sound algorithm that minimizes the L^2 distance in time linear in the number of cells.

5.1 Minimizing L^∞ and L^1 Distance

We first consider minimizing the L^∞ distance, which is essentially minimizing the maximal difference between $\tilde{\mathsf{c}}(a)$ and $\hat{\mathsf{c}}(a)$ for any cell in $\mathcal{E}_{\mathsf{pre}}$, i.e., $||\hat{\mathsf{c}}(\cdot) - \tilde{\mathsf{c}}(\cdot)||_\infty = \max_{a \in \mathcal{E}_{\mathsf{pre}}} |\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)|$. Equivalently, we solve for $\hat{\mathsf{c}}(\cdot)$ in the following linear program.

$$\begin{aligned} &\text{minimize } z \\ &\text{s.t.} \quad |\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)| \leq z, \ \ \forall \text{ cells } a \in \mathcal{E}_{\mathsf{pre}}; \\ &\qquad \sum_{a' \in \mathrm{Base}(a)} \hat{\mathsf{c}}(a') = \hat{\mathsf{c}}(a), \ \ \forall \text{ cells } a \in \mathcal{E}_{\mathsf{pre}}. \end{aligned} \tag{6}$$

[4] considers a similar consistency enforcing scheme, but injects noise into all cuboids instead of the carefully-selected subset of cuboids $\mathcal{L}_{\mathsf{pre}}$. We can bound the error of $\hat{\mathsf{c}}(\cdot)$ as follows.

THEOREM 8. (Generalized Theorem 8 in [4]) *For $\hat{\mathsf{c}}(\cdot)$ obtained by solving (6), with probability at least $1 - \delta$, where $\delta = \frac{|\mathcal{E}_{\mathsf{pre}}|}{e^{\eta/2}}$,*

$$\sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\hat{\mathsf{c}}(a) - \mathsf{c}(a)| \leq \frac{|\mathcal{E}_{\mathsf{pre}}||\mathcal{L}_{\mathsf{pre}}|}{\epsilon} 2 \log \frac{|\mathcal{E}_{\mathsf{pre}}|}{\delta} = \frac{|\mathcal{E}_{\mathsf{pre}}||\mathcal{L}_{\mathsf{pre}}|}{\epsilon} \eta.$$

A linear program can also be used to minimize the L^1 distance. As $||\hat{\mathsf{c}}(\cdot) - \tilde{\mathsf{c}}(\cdot)||_1 = \sum_a |\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)|$, by introducing an auxiliary variable z_a for each cell a with constraint $-z_a \leq \hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a) \leq z_a$, minimizing $||\hat{\mathsf{c}}(\cdot) - \tilde{\mathsf{c}}(\cdot)||_1$ while enforcing consistency in $\hat{\mathsf{c}}(\cdot)$ is equivalent to the following linear program.

$$\begin{aligned} &\text{minimize } \sum_{a \in \mathcal{E}_{\mathsf{pre}}} z_a \\ &\text{s.t.} \quad |\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)| \leq z_a, \ \ \forall \text{ cell } a \in \mathcal{E}_{\mathsf{pre}}; \\ &\qquad \sum_{a' \in \mathrm{Base}(a)} \hat{\mathsf{c}}(a') = \hat{\mathsf{c}}(a), \ \ \forall \text{ cell } a \in \mathcal{E}_{\mathsf{pre}}. \end{aligned} \tag{7}$$

THEOREM 9. *For $\hat{\mathsf{c}}(\cdot)$ obtained by solving (7), if $\epsilon \leq 1$, with probability at least $1 - \delta$, where $\delta = \left(\frac{\eta}{2e^{\eta/2-1}}\right)^{|\mathcal{E}_{\mathsf{pre}}|}$ ($\eta > 2$),*

$$\sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\hat{\mathsf{c}}(a) - \mathsf{c}(a)| \leq \frac{|\mathcal{E}_{\mathsf{pre}}||\mathcal{L}_{\mathsf{pre}}|}{\epsilon} \eta.$$

For a data cube where the cardinality of every dimension is two, Theorem 8 in [4] yields a bound similar to that of our Theorem 8, by discarding the integrality constraint (i.e., counts are integers).

Theorem 9 mirrors Theorem 8, but replaces the upper tail $\delta = \frac{|\mathcal{E}_{\mathsf{pre}}|}{e^{\eta}}$ with $\left(\frac{\eta}{2e^{\eta/2-1}}\right)^{|\mathcal{E}_{\mathsf{pre}}|}$. As $|\mathcal{E}_{\mathsf{pre}}|$ is usually large, linear program (7) and Theorem 9 give a much better bound on the average error in $\hat{\mathsf{c}}(\cdot)$ than linear program (6) and Theorem 8.

To enforce the integrality constraint for $\hat{\mathsf{c}}(\cdot)$ obtained from linear program (6) or (7), we can simply round $\hat{\mathsf{c}}(a)$ for each cell to the nearest integer, which will replace the error bound $\frac{|\mathcal{E}_{\mathsf{pre}}||\mathcal{L}_{\mathsf{pre}}|}{\epsilon}\eta$ with $\frac{|\mathcal{E}_{\mathsf{pre}}||\mathcal{L}_{\mathsf{pre}}|}{\epsilon}\eta + |\mathcal{E}_{\mathsf{pre}}|$ in both Theorems 8 and 9.

5.2 Minimizing L^2 Distance

Now let's focus on minimizing the L^2 distance, i.e., the sum of squared differences between $\tilde{\mathsf{c}}(a)$ and $\hat{\mathsf{c}}(a)$ for all cells. This problem has a very efficient solution with good statistical guarantees.

$$\begin{aligned} &\text{minimize } \sum_{a \in \mathcal{E}_{\mathsf{pre}}} (\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a))^2 \\ &\text{s.t.} \ \sum_{a' \in \mathrm{Base}(a)} \hat{\mathsf{c}}(a') = \hat{\mathsf{c}}(a), \ \ \forall \text{ cell } a \in \mathcal{L}_{\mathsf{pre}}. \end{aligned} \tag{8}$$

Program (8) can be viewed as a *least L^2-norm problem*. Its unique solution $\hat{\mathsf{c}}(\cdot)$, called *the least square solution*, can be found using linear algebra [7]. The classical method needs to compute multiplication/inversion of $M \times M$-matrices, where $M = \Pi_j |A_j|$ is the total number of *base cells*. Since M is typically larger than 10^6, the classical method is inefficient in our context. Fortunately, we can derive the optimal solution $\hat{\mathsf{c}}(\cdot)$ much more efficiently by utilizing the structure of data cubes, as follows.

We define $\mathrm{Ancs}(a')$ to be the set of *ancestor cells* of cell a': $a \in \mathrm{Ancs}(a')$ if and only if a is in an ancestor of the cuboid containing a' and has the same value as a' on all non-$*$ dimensions; formally, $a \in \mathrm{Ancs}(a')$ if and only if for any dimension A_i, $a[i] \neq *$ implies $a[i] = a'[i]$. If a' is a base cell, $a \in \mathrm{Ancs}(a') \Leftrightarrow a' \in \mathrm{Base}(a)$.

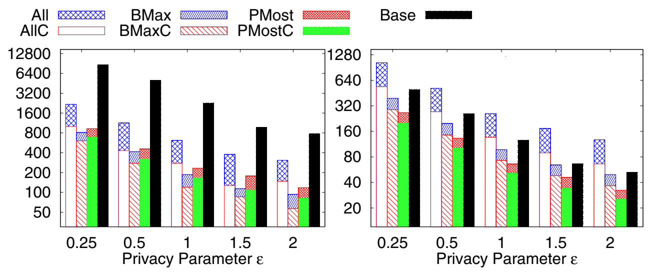
For a cell a and a cuboid C, let $a[C]$ be a's values on dimensions of $[C]$. Suppose $[C] = \{A_{i_1}, \ldots, A_{i_k}\}$, $a[C] = \langle a[i_1], \ldots, a[i_k] \rangle$.

For two cuboids C_1 and C_2, let $C_1 \vee C_2$ be the cuboid with dimensions $[C_1] \cup [C_2]$ and $C_1 \wedge C_2$ with dimensions $[C_1] \cap [C_2]$.

We provide the following two-stage algorithm to compute the consistent measure $\hat{\mathsf{c}}(\cdot)$ that is the optimal solution to program (8):

1. Bottom-up stage: We first compute $\mathrm{obs}(a'')$ for every cell a'' in a cuboid C'' that can be computed from $\mathcal{L}_{\mathsf{pre}}$. Let

$$\mathrm{obs}(a'') = \sum_{a' \in \mathrm{Base}(a'')} \ \sum_{a \in \mathcal{E}_{\mathsf{pre}} \cap \mathrm{Ancs}(a')} \tilde{\mathsf{c}}(a). \tag{9}$$



(a) Max Cuboid Error (b) Average Cuboid Error

Figure 3: Varying privacy parameter ϵ

A cuboid $C \in \mathcal{L}_{\text{pre}}$ is *maximal in* $\mathcal{L}_{\text{pre}}$ if there is no $C' \in \mathcal{L}_{\text{pre}}$ s.t. $C \prec C'$. In all maximal cuboids C'' in $\mathcal{L}_{\text{pre}}$, $\text{obs}(a'')$ can be computed as in Formula (9). In all the other cuboids that can be computed from $\mathcal{L}_{\text{pre}}$, $\text{obs}(a'')$ can be computed recursively as follows: suppose $a'' \in C''$ and C'' can be computed from $\mathcal{L}_{\text{pre}}$, there must be some cuboid C with dimensionality $\dim(C'')+1$ s.t. either $C \in \mathcal{L}_{\text{pre}}$ or C can be computed from $\mathcal{L}_{\text{pre}}$; then,

$$\text{obs}(a'') = \sum_{\substack{a:\ a\in C\\ a[C'']=a''[C'']}} \text{obs}(a). \tag{10}$$

2. Top-down stage: After $\text{obs}(a'')$ is computed for every possible cell a'', consider another quantity $\text{est}(a'')$ for each cell $a'' \in C''$:

$$\text{est}(a'') = \sum_{C\in\mathcal{L}_{\text{pre}}} \sum_{\substack{b:\ b\in(C\wedge C''),\\ b[C\wedge C'']=a''[C\wedge C'']}} \deg(C\vee C'')\hat{\mathsf{c}}(b). \tag{11}$$

where $\deg(C) = \prod_{A_i\in\mathcal{A}-[C]} |A_i|$ and $\mathcal{A}$ is the set of all dimensions and $|A_i|$ is the cardinality of dimension A_i. Suppose $\text{obs}(\cdot)$'s are computed in the first stage as constants and $\hat{\mathsf{c}}(\cdot)$'s are variables. Solving the equations $\text{est}(a'') = \text{obs}(a'')$, we can compute $\hat{\mathsf{c}}(\cdot)$'s in a top-down manner (from ancestors to descendants) as follows:

$$\text{ratio}(C'') = \sum_{\substack{C:\ C\in\mathcal{L}_{\text{pre}},\\ C''\preceq C}} \deg(C\vee C''), \tag{12}$$

$$\text{aux}(a'') = \sum_{\substack{C:\ C\in\mathcal{L}_{\text{pre}},\\ C''\not\preceq C}} \sum_{\substack{b:\ b\in(C\wedge C''),\\ b[C\wedge C'']=a''[C\wedge C'']}} \deg(C\vee C'')\hat{\mathsf{c}}(b), \tag{13}$$

$$\hat{\mathsf{c}}(a'') = \frac{1}{\text{ratio}(C'')}\left(\text{obs}(a'') - \text{aux}(a'')\right). \tag{14}$$

The above approach can be also applied to $\mathcal{K}_{\text{all}}$ because $\mathcal{K}_{\text{all}}$ is a special case of $\mathcal{K}_{\text{part}}$ obtained by setting $\mathcal{L}_{\text{pre}} = \mathcal{L}$.

Note that this approach generalizes the consistency-enforcing scheme in [19] from a tree-like structure (hierarchy of intervals) to a lattice. The method in [19] cannot be directly applied here.

Optimality. We can prove $\hat{\mathsf{c}}(\cdot)$ obtained above is not only consistent but also an unbiased estimator of $\mathsf{c}(\cdot)$. Also, $\hat{\mathsf{c}}(\cdot)$ is optimal in the sense that no other linear unbiased estimator of $\mathsf{c}(\cdot)$ obtained from $\tilde{\mathsf{c}}(\cdot)$ has smaller variance, i.e., smaller expected squared error.

THEOREM 10. *(i) The above algorithm correctly computes a value for $\hat{\mathsf{c}}(\cdot)$ that is consistent and solves the L^2 minimization problem* (8)*. (ii) The above algorithm requires $O(N(d^2+d|\mathcal{L}_{\text{pre}}|))$ time to compute $\hat{\mathsf{c}}(\cdot)$ for all N cells. (iii) For any cell of cuboids in $\mathcal{L}_{\text{pre}}$, $\hat{\mathsf{c}}(a)$ is an unbiased estimator of $\mathsf{c}(a)$, i.e., $\mathrm{E}[\hat{\mathsf{c}}(a)] = \mathsf{c}(a)$. (iv) For any cell a, $\hat{\mathsf{c}}(a)$ has the smallest variance among any linear unbiased estimator (e.g., $\tilde{\mathsf{c}}(a)$) of $\mathsf{c}(a)$ obtained from $\tilde{\mathsf{c}}(\cdot)$. (v) For any set P of cells, $\sum_{a\in P}\hat{\mathsf{c}}(a)$ has the smallest variance among any linear unbiased estimator of $\sum_{a\in P}\mathsf{c}(a)$ obtained from $\tilde{\mathsf{c}}(\cdot)$.*

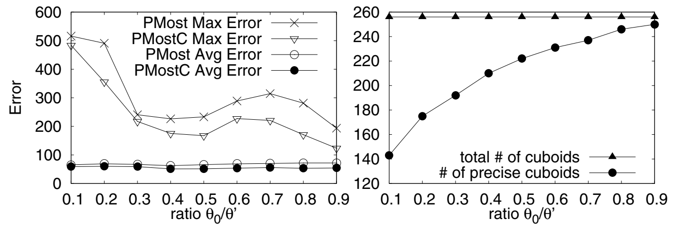


Figure 4: Varying θ_0 in Algorithm 2 (for PMost and PMostC)

6. EXPERIMENTS

We evaluate and compare seven proposed techniques on both real and synthetic datasets. The first two techniques are $\mathcal{K}_{\text{all}}$ and $\mathcal{K}_{\text{base}}$, which were defined at the beginning of Section 3. We denote them as All and Base, respectively, in this section. Another two are based on our generalized framework $\mathcal{K}_{\text{part}}$. One of these has the objective of bounding the max noise variance (Problem 1) and is denoted by BMax ($\mathcal{K}_{\text{part}}$ + Algorithm 1). The other one has the objective of maximizing the number of cuboids with noise variance no more than a given variance threshold θ_0 (Problem 2) and is denoted by PMost ($\mathcal{K}_{\text{part}}$ + Algorithm 2). The last three techniques involve applying the methods in Section 5.2 to enforce consistency in the data cubes published by All ($\mathcal{L}_{\text{pre}} = \mathcal{L}$), BMax, and PMost. The resulting three techniques are denoted AllC, BMaxC, and PMostC, respectively. The LP-based techniques in Section 5.1 are not practical for large tables. All seven algorithms are coded in C++ and evaluated on an 8GB 64-bit 2.40GHz PC.

Real dataset: We use the Adult dataset from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/) with census information. It contains 32,561 rows with 8 categorical dimensions: workclass (cardinality 9), education (16), marital-status (7), occupation (15), relationship (6), race (5), sex (2), and salary (2).

Synthetic dataset: To generate the fact table, we first fix the number of dimensions and the dimension cardinalities. Then we generate each row independently, with each of its column values drawn uniformly and independently from the domain of its dimension.

Error measurement: We compute the error as the absolute difference between the real count measure computed directly from the fact table and the noisy measure released by one of the seven ϵ-DP algorithms. The *cuboid error* is the average error for all cells in this cuboid. We report both the *max cuboid error* and the *average cuboid error* among all published cuboids.

We release $\mathcal{L}$ = all cuboids and evaluate all algorithms on the same set of published cuboids. This is the hardest test for all algorithms, as less noise will be needed if $\mathcal{L}$ omits some cuboids.

Exp-I: Varying privacy parameter ϵ. Using the Adult dataset, we vary privacy parameter ϵ from 0.25 to 2. Smaller ϵ implies more privacy and thus more noise. Recall Algorithm 2 (for PMost) takes a variance threshold θ_0 in the input; here, we set $\theta_0 = 0.5\theta'$, where θ' is the variance bound found by Algorithm 1 (for BMax).

Figure 3 uses a logarithmic scale to show the max/average error in the released cuboids. As the inconsistent version of an approach always has at least as much error as the consistent version, the two versions (All/AllC, BMax/BMaxC, and PMost/PMostC) are stacked together on a single bar in each histogram in Section 6.

BMax and PMost always incur much less error than the two baselines Base and All. BMax is better than PMost for bounding the max error, while PMost is better than BMax for bounding the average error. Base performs the worst in terms of the max error, because only the base cuboid is computed from the table, and the noise is magnified significantly when cuboids at higher levels are computed by aggregating cells in the base cuboid. All is the worst

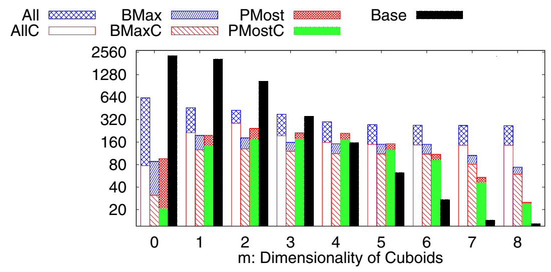


Figure 5: Max cuboid error in different cuboids as dimensionality varies, when all cuboids must be released

in terms of the average error, for the large amount of noise initially injected in each cuboid. Error decreases as ϵ increases.

As suggested by Theorem 10 (iv)-(v), our consistency enforcing techniques tend to reduce error, since the variance of the consistent measure $\hat{c}(\cdot)$ is no larger than that of the inconsistent $\tilde{c}(\cdot)$. So AllC/BMaxC/PMostC reduces error by 30%-50%, compared to All/BMax/PMost, while providing the same privacy guarantee.

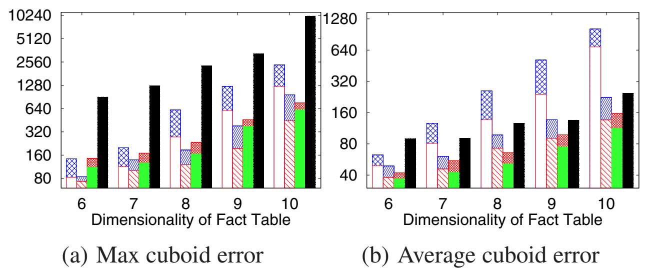
Exp-II: Varying variance threshold θ_0. Note that performance of PMost and PMostC depends on the input threshold θ_0 in Problem 2, as θ_0 determines $\mathcal{L}_{\text{pre}}$. Fix $\epsilon = 1$. Given the variance bound θ' found by Algorithm 1, we vary θ_0 from $0.1\theta'$ to $0.9\theta'$. Figure 4 shows how θ_0 affects Algorithm 2 on the Adult dataset. As θ_0 increases, we have more precise cuboids, i.e., the released cuboids with noise variance no more than θ_0. When $\theta_0 = \theta'$, all $2^8 = 256$ cuboids are precise, and PMost/PMostC is equivalent to BMax/BMaxC, as the same $\mathcal{L}_{\text{pre}}$ is used. But the max/average error of PMost and PMostC does not increase monotonically with θ_0: both $\theta_0 = \theta'$ and $\theta_0 = 0.5\theta'$ are local optimums for minimizing errors. In the remaining experiments, we set $\theta_0 = 0.5\theta'$.

Exp-III: Noise in different cuboids. Figure 5 shows the max error in cuboids of different dimensionality on Adult, for $\epsilon = 1$. Recall in an m-dim cuboid, a cell has non-$*$ values on exactly m dimensions. As m decreases, a cell in an m-dim cuboid aggregates more base cells. So Base aggregates more noise from base cells as m drops, and performs the worst for $m < 3$. BMaxC is the best when $0 < m < 5$, and its max error changes little for $0 < m < 7$. The consistency-enforcing techniques, AllC/BMaxC/PMostC, are very effective for small m, reducing error by up to 70%.

Exp-IV: Varying number of dimensions. We create two more dimensions, each of which has cardinality 4, on the Adult dataset, and generate their values randomly for each row. We consider the fact table with the first 6 to all the 10 dimensions. Fix $\epsilon = 1$. We report the max/average error of the seven approaches in Figure 6. As the number of dimensions increases, BMaxC and PMostC are always the best two, and the error in both All and Base is much larger and increases faster than the error in others.

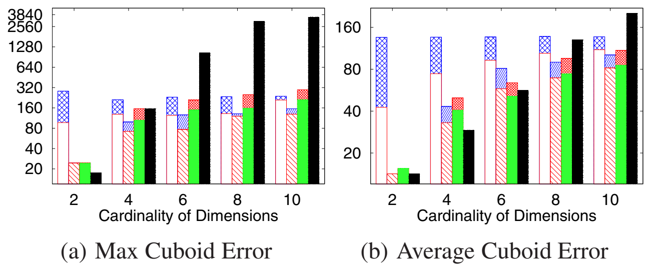
Exp-V: Varying cardinality of dimensions. We generate synthetic tables with 7 dimensions, each of which has the same cardinality. We vary the cardinality from 2 to 10, and generate a table with 10^8 rows randomly for each. We report the error of each approach in Figure 7, for $\epsilon = 1$. The performance of Base deteriorates quickly as the cardinality increases, because more cells in the base cuboid need to be aggregated. The performance of All does not change much, for its performance is mainly determined by the number of dimensions, which determines the total number of cuboids. Again, BMaxC and PMostC perform best in most cases.

Exp-VI: Efficiency. Consider a data cube with 5-10 dimensions on the Adult dataset (with two more synthetic dimensions as in Exp-IV). The overall DP data cube publishing time is reported in Figure 8, which can be decomposed as follows. Recall Algorithms 1-2



(a) Max cuboid error (b) Average cuboid error

Figure 6: Varying dimensionality of fact table



(a) Max Cuboid Error (b) Average Cuboid Error

Figure 7: Varying cardinality of dimensions (7 dimensions)

select $\mathcal{L}_{\text{pre}}$ for BMax/BMaxC and PMost/PMostC, respectively. The running time of Algorithms 1-2 is reported in Figure 9(a). For different choices of $\mathcal{L}_{\text{pre}}$ on the 8-dim table, the time needed for noise injection ($\mathcal{K}_{\text{part}}$) and consistency enforcement (the method in Section 5.2) in all cuboids is reported in Figure 9(b).

From Figure 9(a), although the running time of Algorithms 1-2 is polynomial in 2^d, they are not the bottleneck. Their running time is always a very small portion of the overall DP data cube publishing.

From Figure 9(b), it is shown that the consistency-enforcement time increases linearly with $|\mathcal{L}_{\text{pre}}|$, as predicted by Theorem 10 (ii). The time for noise injection decreases as $|\mathcal{L}_{\text{pre}}|$ increases. This is because when more cuboids are initially injected with noise, less aggregation of noise occurs later on.

From Figure 8, using consistency enforcement, AllC is especially expensive, because $|\mathcal{L}_{\text{pre}}| = 2^d$ in AllC. BMaxC and PMostC, although using consistency enforcement, usually only need 3%-10% time of AllC, for they use smaller $\mathcal{L}_{\text{pre}}$'s. For all approaches, the publishing time increases exponentially with the dimensionality, mainly because the total number of cells increases exponentially.

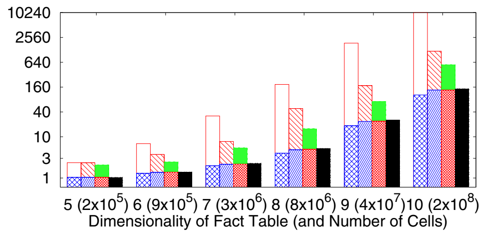
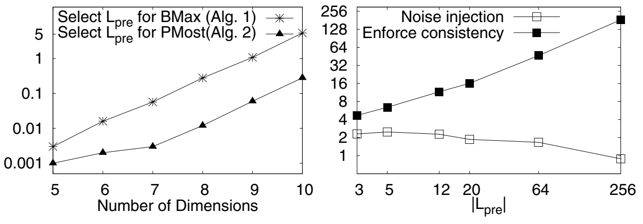


Figure 8: DP data cube publishing time (in seconds)

Summary. Both All and Base are sensitive to the dimensionality of the fact table, and Base is also sensitive to cardinalities of dimensions. All usually has a large average error, as a large amount of noise is injected into all cuboids. Base has a large max error, because noise is aggregated from the base cells; and that is why Base incurs small average errors in the cuboids close to the base cuboid. BMaxC and PMostC are the best most of the time.

All/BMax/PMost run much faster than AllC/BMaxC/PMostC. But with our consistency enforcement, AllC/BMaxC/PMostC reduce error in All/BMax/PMost, respectively, by typically 30%-70% or more, and ensure that the published data is consistent.

The error of BMaxC and PMostC is usually only 5%-30% of



(a) Time (in seconds) (b) Time (in seconds)

Figure 9: Efficiency of different algorithms

the error incurred by All, and 20%-50% of the error of AllC. Note that the y-axis in our figures is always in logarithmic scale.

Because of our consistency-enforcing method, the error of AllC is sometimes comparable to BMax/BMaxC and PMost/PMostC, when the dimensionality of the fact table is low. However, AllC is very expensive because $|\mathcal{L}_{\mathsf{pre}}|$ in AllC is equal to 2^d (recall Theorem 10 for the running time of our consistency enforcement). When there are more than five dimensions, AllC's publishing time is 6-10 times larger than BMaxC's (> 10 times for ten dimensions), and 10-40 times larger than PMostC's (> 40 times for ten dimensions).

7. DISCUSSION AND EXTENSIONS

7.1 Relative Error vs. Absolute Error

The amount of noise $\mathcal{K}_{\mathsf{part}}$ adds to DP cuboids is independent of the number of rows and specific data in the fact table. Instead, the selection of $\mathcal{L}_{\mathsf{pre}}$ and the amount of noise depend only on which cuboids are to be published, the number of dimensions, and their cardinalities. So our expected absolute error is fixed if the structure of the fact table is fixed, no matter how many rows there are. This feature of our approaches is also true in DP-based frameworks for different publishing tasks [4, 19, 23, 36]. The implication is that the expected relative error cannot be bounded in general. Because, with the expected absolute error fixed, some cells may have very small values of the count measure (e.g., 1), while some have very large values (e.g., 10^3). That is also why we report absolute error in Section 6. On the other hand, the advantage is, for a particular cell, it has less relative error if it aggregates more rows.

7.2 Further Reduction of Noise

We can generalize our noise-control framework $\mathcal{K}_{\mathsf{part}}$ by adding different amounts of noise to different cuboids in $\mathcal{L}_{\mathsf{pre}}$ to further reduce noise. Suppose $\mathcal{L}_{\mathsf{pre}} = \{C_1, \ldots, C_s\}$ and noise $\mathrm{Lap}(\lambda_i)$ is injected into each cell in C_i. By the composition property of DP [28], we claim that if $\sum_{i=1}^{s} 1/\lambda_i = \epsilon$, then publishing $\mathcal{L}_{\mathsf{pre}}$ is ϵ-DP. Finding the optimal $\mathcal{L}_{\mathsf{pre}}$ and parameters $\langle \lambda_i \rangle$ to achieve the goals in Problems 1 and 2 is hard, but Algorithms 1 and 2 can be modified to provide approximate solutions. Here, privacy parameter ϵ can be interpreted as the cost budget in WEIGHTED SET COVER.

To enforce consistency and yield an optimality result similar to Theorem 10, we can solve a weighted version of the least squares problem in Section 5.2. Our experiments show this reduces absolute error in BMaxC/PMostC by 10%. We omit the details here.

7.3 Extension to Other Measures

Our techniques for the count measure can be extended to other two basic measures sum and avg. sum can be considered as a generalized count measure, where each row in the fact table is associated with a value instead of just 0-1. Compared to count, the sensitivity of a publishing function for sum is magnified Δ times, where Δ is the range of possible values for any individual fact table tuple. Thus our techniques for count can be applied to sum, with the noise variance magnified Δ^2 times. To handle the average measure avg, we can compute two DP data cubes (partitioning the privacy budget across them), one with sum measure and one with count measure, from the same fact table. The avg measure of a cell can be computed from the two cubes, without violating DP.

7.4 Handling Larger Data Cubes

Our approaches introduced so far are applicable for mid-size data cubes, e.g., with ≤ 12 dimensions or $\leq 10^9$ cells. Since the current framework needs to store all cells for consistency enforcement, it cannot handle data cubes with too many cells.

For even larger data cubes (e.g., with ≥ 20 dimensions and $\geq 2^{20}$ cuboids), it is unnecessary to publish all cuboids at one time, as typical users are likely to query only a very small portion of them. Also, it is impossible to publish all cuboids while ensuring DP, as the huge amount of noise will make the result meaningless. So we outline an online version of our approach as follows.

Initially, $\mathcal{L}_{\mathsf{pre}} = \varnothing$ and we have certain amount ϵ of privacy budget. When a cuboid query C comes, if $C \notin \mathcal{L}_{\mathsf{pre}}$ and C can be computed from some DP cuboid $C^* \in \mathcal{L}_{\mathsf{pre}}$, there are two choices: a) compute C from C^*, with error in C^* magnified in C; or b) compute real cuboid C using high-dimensional OLAP techniques like [25], inject noise into C to obtain a DP cuboid, insert C into $\mathcal{L}_{\mathsf{pre}}$, and deduct a certain amount of privacy budget from ϵ. If $C \notin \mathcal{L}_{\mathsf{pre}}$ but C cannot be computed from any DP cuboid $C^* \in \mathcal{L}_{\mathsf{pre}}$, we have to follow b) above. If $C \in \mathcal{L}_{\mathsf{pre}}$ or C used to be queried, we can directly output the old DP cuboid C. After we run out of privacy budget ϵ, to ensure ϵ-DP, we cannot create new DP cuboids in $\mathcal{L}_{\mathsf{pre}}$ any more and may be unable to answer new queries. How to distribute the privacy budget online so that more queries can be answered with less error is interesting future work.

8. RELATED WORK

Since ϵ-differential privacy (DP) [12] was introduced, many techniques have been proposed for different data publishing and analysis tasks (refer to [10, 11] for a comprehensive review). For example, the notion of DP has been applied to releasing query and click histograms from search logs [18, 22], recommender systems [29], publishing commuting patterns [27], publishing results of machine learning [6, 8, 20], clustering [13, 30], decision tree [14], mining frequent patterns [5], and aggregating distributed time-series [31]. For a single counting query, one method to ensure DP [12] has been shown to be optimal under a certain utility model [17].

Recent works [36, 19, 23] on differentially private count queries over histograms are related to our work. Xiao *et al.* [36] propose to use the Haar wavelet for range count queries. Hay *et al.* [19] propose an approach based on a hierarchy of intervals. Li *et al.* [23] propose a general framework which supports answering a given workload of count queries, and consider the problem of finding optimal strategies for a workload. While [36] and [19] can be unified in the framework in [23], the specific algorithms given in [36, 19] are more efficient than the matrix multiplication used in [23].

Xiao *et al.* extend their wavelet approach to nominal attributes and multidimensional count queries. Their extended approach can be applied in our problem to achieve the same noise bounds as our $\mathcal{K}_{\mathsf{all}}$ (refer to Theorem 3 in [36] and Theorem 2 in this work). In general, our $\mathcal{K}_{\mathsf{part}}$ will add less noise, as shown in Theorems 6-7.

Barak *et al.* [4] show how to publish a set of marginals of a contingency table while ensuring DP. One of their two approaches is similar to $\mathcal{K}_{\mathsf{all}}$: add noise to all the marginals to be published. Their LP-rounding method minimizes the L^∞ distance while enforcing consistency and integrality and removing negative numbers in the publishing. Our Theorem 9 shows that minimizing the L^1 distance yields a much better theoretical bound on error. The other approach in [4] is similar, but moves to the Fourier domain at first. Unlike our work, they do not optimize the publishing strategy (i.e., the selection of $\mathcal{L}_{\mathsf{pre}}$). Moreover, the number of variables in the LP equals the number of cells (often $> 10^6$ in our experiments). So LP-based methods can only handle data cubes with small numbers of cells.

Agrawal *et al.* [3] study how to support OLAP queries while ensuring a limited form of (ρ_1, ρ_2)-privacy, by randomizing each entry in the fact table with a constant probability. An OLAP query on the fact table can be answered from the perturbed table within roughly $\sqrt{|\mathrm{dataset}|}$ [4]. (ρ_1, ρ_2)-privacy is in general not as strong as ϵ-differential privacy. Also, the error incurred by this method [3] depends on the dataset size; in our framework, the amount of noise to be added is data-independent, only determined by the number of cuboids to be published and the structure of the fact table.

Differential privacy provides much stronger privacy guarantees than other privacy concepts based on deterministic algorithms do, such as k-anonymity [32] and its extension l-diversity [26] and t-closeness [24]. [34] studies how to specify authorization and control inferences for OLAP queries in the data cube.

9. CONCLUSIONS

We have introduced a general noise-control framework to release all or part of a data cube in an ϵ-differentially private way. To reduce the noise in the released cuboids, we choose certain cuboids $\mathcal{L}_{\mathsf{pre}}$ to compute directly from the fact table, and add noise to make them differentially private. All the other cuboids are computed directly from the cuboids in $\mathcal{L}_{\mathsf{pre}}$, without adding additional noise or revisiting the fact table. We modeled the selection of $\mathcal{L}_{\mathsf{pre}}$ as two optimization problems, proved them NP-hard, and proposed approximation algorithms. To ensure consistency across different rollups of released cuboids, we proposed consistency-enforcing techniques that have the side benefit of reducing noise.

10. ACKNOWLEDGMENTS

We thank the reviewers for their insightful suggestions that immensely improved the paper. This work was supported in part by NSF IIS-09-05215, U.S. Air Force Office of Scientific Research MURI award FA9550-08-1-0265, the U.S. Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053 (NS-CTA), A*STAR SERC grant 1021580074, and a grant from Boeing. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

11. REFERENCES

[1] www.cs.cmu.edu/ compthink/mindswaps/oct07/difpriv.ppt. 2007.

[2] N. R. Adam and J. C. Wortmann. Security-control methods for statistical databases: A comparative study. *ACM Comput. Surv.*, 21(4):515–556, 1989.

[3] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving OLAP. In *SIGMOD*, pages 251–262, 2005.

[4] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS*, pages 273–282, 2007.

[5] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *KDD*, pages 503–512, 2010.

[6] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, 2004.

[8] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *NIPS*, pages 289–296, 2008.

[9] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge Univ. Press, 2009.

[10] C. Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.

[11] C. Dwork. The differential privacy frontier (extended abstract). In *TCC*, pages 496–502, 2009.

[12] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[13] D. Feldman, A. Fiat, H. Kaplan, and K. Nissim. Private coresets. In *STOC*, pages 361–370, 2009.

[14] A. Friedman and A. Schuster. Data mining with differential privacy. In *KDD*, pages 493–502, 2010.

[15] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey on recent developments. *ACM Comput. Surv.*, 42(4), 2010.

[16] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *KDD*, pages 265–273, 2008.

[17] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. In *STOC*, pages 351–360, 2009.

[18] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Publishing search logs - a comparative study of privacy guarantees. *TKDE*, 2011.

[19] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially-private queries through consistency. In *PVLDB*, pages 1021–1032, 2010.

[20] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? In *FOCS*, pages 531–540, 2008.

[21] D. Kifer. Attacks on privacy and deFinetti's theorem. In *SIGMOD*, pages 127–138, 2009.

[22] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *WWW*, pages 171–180, 2009.

[23] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing histogram queries under differential privacy. In *PODS*, pages 123–134, 2010.

[24] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, pages 106–115, 2007.

[25] X. Li, J. Han, and H. Gonzalez. High-dimensional OLAP: A minimal cubing approach. In *VLDB*, pages 528–539, 2004.

[26] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.

[27] A. Machanavajjhala, D. Kifer, J. M. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, pages 277–286, 2008.

[28] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30, 2009.

[29] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the Netflix prize contenders. In *KDD*, pages 627–636, 2009.

[30] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.

[31] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD*, pages 735–746, 2010.

[32] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *PODS*, page 188, 1998.

[33] S. D. Silvey. *Statistical Inference*. Chapman-Hall, 1975.

[34] L. Wang, S. Jajodia, and D. Wijesekera. Preserving privacy in on-line analytical processing data cubes. In *Secure Data Management in Decentralized Systems*, pages 355–380. 2007.

[35] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Minimality attack in privacy preserving data publishing. In *VLDB*, pages 543–554, 2007.

[36] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, pages 225–236, 2010.

APPENDIX: Proofs

Proof of Theorem 2. The vector $F_{\mathsf{all}}(T)$ has sensitivity $S(F_{\mathsf{all}}) = |\mathcal{L}|$. So from Theorem 1, $\mathcal{K}_{\mathsf{all}}$ is ϵ-DP. As the noise is $\mathrm{Lap}(|\mathcal{L}|/\epsilon)$, we have $\mathrm{E}\left[\tilde{\mathsf{c}}(a)\right] = \mathsf{c}(a)$ and $\mathrm{Var}\left[\tilde{\mathsf{c}}(a)\right] = 2|\mathcal{L}|/\epsilon^2$. □

Proof of Theorem 3. The sensitivity $S(F_{\mathsf{base}}) = 1$. So from Theorem 1, publishing $F_{\mathsf{base}}(T) + \langle \mathrm{Lap}(1/\epsilon)\rangle$ is ϵ-DP. For any other cell a not in the base cuboid, the computation of $\tilde{\mathsf{c}}(a)$ takes the released base cuboid as input, so from the composition property [28], is ϵ-DP. For a cell a in a cuboid C, it aggregates $\Pi_{A_j \notin [C]}|A_j|$ cells $\{a'\}$ in the base cuboid. Then $\mathrm{E}\left[\tilde{\mathsf{c}}(a)\right] = \Sigma_{a'}\mathrm{E}\left[\tilde{\mathsf{c}}(a')\right] = \Sigma_{a'}\mathsf{c}(a') = \mathsf{c}(a)$. And since noise is generated independently, we have the variance $\mathrm{Var}\left[\tilde{\mathsf{c}}(a)\right] = \Sigma_{a'}\mathrm{Var}\left[\tilde{\mathsf{c}}(a')\right] = \left(\Pi_{A_j \notin [C]}|A_j|\right) \cdot 2/\epsilon^2$. □

Proof of Theorem 4.

Consider measures $\langle \mathsf{c}(a)\rangle$ for all cells in the s selected cuboids in $\mathcal{L}_{\mathsf{pre}}$, the sensitivity of publishing $\mathcal{L}_{\mathsf{pre}}$ is $s = |\mathcal{L}_{\mathsf{pre}}|$. So by Theorem 1 and the composition property of DP, adding a noise $\mathrm{Lap}(s/\epsilon)$ to each cell in $\mathcal{L}_{\mathsf{pre}}$ and computing $\mathcal{L}$ from $\mathcal{L}_{\mathsf{pre}}$ is ϵ-DP.

$\mathrm{E}\left[\tilde{\mathsf{c}}(a)\right] = \mathsf{c}(a)$ is also from the linearity of expectation. □

Proof of Theorem 5.

To complete the proof, we reduce the problem VERTEX COVER IN DEGREE-3 GRAPHS to the BOUND MAX VARIANCE problem. Then the hardness of the PUBLISH MOST problem follows.

Following is an instance of the VERTEX COVER problem in a degree-3 graph (where the degree of a vertex is bounded by 3). Given a degree-3 (undirected) graph $G(V, E)$ where $|V| = n$, decide if G has a vertex cover $V' \subseteq V$ with at most m $(< n)$ vertices. $V' \subseteq V$ is said to be a vertex cover of G iff for any edge $uv \in E$, we have either $u \in V'$ or $v \in V'$. Abusing the notations a bit, let $\mathrm{cov}(v)$ be the edges incident on a vertex v, then we want to decide if there is $V' \subseteq V$ such that $\cup_{v \in V'}\mathrm{cov}(v) = E$ and $|V'| \le m$.

We can assume the degree of any vertex in V is larger than 1, since a degree-1 vertex will be never chosen into a minimum vertex cover. And since there is at least one vertex with degree 3 (otherwise G can be decomposed into cycles which are the trivial case for the VERTEX COVER problem), we have $|E| > 2n/2 = n$.

Construct an instance of the BOUND MAX VARIANCE problem from the above instance of VERTEX COVER problem, accordingly:

(i) For each edge $e \in E$, create a dimension A_e with cardinality $|A_e| = 2$, and a 1-dim cuboid $[C_e^1] = \{A_e\}$.

(ii) Create $3n$ distinct dimensions $B_1, \ldots, B_{3n}$, each of cardinality 2. For each of them, create a 1-dim cuboid $[C_i^1] = \{B_i\}$.

(iii) For each vertex $v \in V$, create a 3-dim cuboid $[C_v^3] = \{A_{e_1}, A_{e_2}, A_{e_3}\}$ for each edge e_i incident on v. Note that a vertex may have less than 3 edges incident on it; in this case, we create one or two new distinct dimensions with cardinality 2, and include them in $[C_v^3]$. So we create n 3-dim cuboids here.

(iv) Create another n 3-dim cuboids C_x^3's: each cuboid $[C_x^3] = \{B_{x_1}, B_{x_2}, B_{x_3}\}$, where each B_{x_i}'s are distinct dimensions with cardinality 2 created in (ii).

(v) For each 3-dim cuboid, $[C_v^3] = \{A_{e_1}, A_{e_2}, A_{e_3}\}$ or $[C_x^3] = \{B_{x_1}, B_{x_2}, B_{x_3}\}$, create a new dimension D_v or D_x, respectively, with cardinality 4, and a 4-dim cuboid $[C_v^4] = \{A_{e_1}, A_{e_2}, A_{e_3}, D_v\}$ or $[C_x^4] = \{B_{x_1}, B_{x_2}, B_{x_3}, D_x\}$, respectively. So in total, we have $2n$ 4-dim cuboids created here.

In this instance of the BOUND MAX VARIANCE problem, we want to publish all the 1-dim, 3-dim, and 4-dim cuboids in (i)-(v), denoted by $\mathcal{L}$, and to decide if we can select a set of cuboids $\mathcal{L}_{\mathsf{pre}}$ such that the max noise variance in releasing $\mathcal{L}$ from $\mathcal{L}_{\mathsf{pre}}$ using $\mathcal{K}_{\mathsf{part}}$ is at most $8(3n+m)^2/\epsilon^2$ $(\mathrm{noise}(\mathcal{L}_{\mathsf{pre}}) \le 8(3n+m)^2/\epsilon^2)$.

We prove the VERTEX COVER instance is YES if and only if the BOUND MAX VARIANCE instance is YES, to complete our proof. Due to the space limit, details are deferred to the full version. □

Remark. The main difficulty in the reduction is the lattice structure of cuboids. And, how to prove/disprove the NP-hardness when the number of dimensions d is bounded ($d = O(\log|\mathcal{L}|)$) and how to prove the hardness of approximation are interesting open questions.

Proof of Lemma 1. The proof is from the definition: Let $s = |\mathcal{L}_{\mathsf{pre}}|$. If $\mathrm{noise}(C, \mathcal{L}_{\mathsf{pre}}) \le \theta$, from Equation (2), there is a $C' \in \mathcal{L}_{\mathsf{pre}}$ s.t. $\mathrm{mag}(C, C') \le \frac{\theta\epsilon^2}{2s^2}$. Thus from Equation (4), $C \in \mathrm{cov}(C', \theta, \epsilon, s)$. The converse direction is similar. □

Proof of Theorem 6.

Approximation Ratio. Suppose the optimal solution is θ^*, i.e., there are s^* cuboids $\mathcal{L}_{\mathsf{pre}}$ s.t. for any $C \in \mathcal{L}$, $\mathrm{noise}(C, \mathcal{L}_{\mathsf{pre}}) \le \theta^*$. From Lemma 1, equivalently, there are s^* cuboids $C_1^*, \ldots, C_{s^*}^*$ s.t. $\bigcup_{i=1}^{s^*}\mathrm{cov}(C_i^*, \theta^*, \epsilon, s^*) = \mathcal{L}$. And, from Equation (4), for any cuboid C and $\alpha > 0$, $\mathrm{cov}(C, \alpha^2\theta^*, \epsilon, \alpha s^*) = \mathrm{cov}(C, \theta^*, \epsilon, s^*)$.

Suppose in line 4 of Algorithm 1, we have $\theta = \alpha^2\theta^*$ and $s = \alpha s^*$. In the following part, we will prove when $\alpha = \ln|\mathcal{L}| + 1$, the algorithm FEASIBLE$(\mathcal{L}, \theta, s)$ can find s cuboids $C_1', \ldots, C_s'$, s.t. $\bigcup_{i=1}^{s}\mathrm{cov}(C_i', \theta, \epsilon, s) = \mathcal{L}$, which implies that, with $\mathcal{L}_{\mathsf{pre}} = \{C_1', \ldots, C_s'\}$, $\max_{C \in \mathcal{L}}\mathrm{noise}(C, \mathcal{L}_{\mathsf{pre}}) \le \theta = \alpha^2\theta^*$. So we can conclude that Algorithm 1 finds an $(\ln|\mathcal{L}| + 1)^2$-approximation.

Since for any cuboid C, $\mathrm{cov}(C, \theta, \epsilon, s) = \mathrm{cov}(C, \theta^*, \epsilon, s^*)$ for the selection of θ and p above, we write both of them as $\mathrm{cov}(C)$. Now we only need to prove: if there exists s^* cuboids $C_1^*, \ldots, C_{s^*}^*$ such that $\bigcup_{i=1}^{s^*}\mathrm{cov}(C_i^*) = \mathcal{L}$, the algorithm FEASIBLE finds $s = \alpha s^*$ cuboids $C_1', \ldots, C_s'$ (in this order) s.t. $\bigcup_{i=1}^{s}\mathrm{cov}(C_i') = \mathcal{L}$.

We apply the analysis of the greedy SET COVER algorithm here. Let l_i be the number of cuboids in $\mathcal{L}$ uncovered by $\{C_1', C_2', \ldots, C_i'\}$. Define $l_0 = |\mathcal{L}|$. It is easy to see $l_i \le (1 - 1/s^*)l_{i-1}$, since every iteration we choose C', which covers the most uncovered cuboids in $\mathcal{L}$ ($\{C_1^*, \ldots, C_{s^*}^*\}$ also cover these uncovered cuboids, so the selected C' covers no less than any of them). So $l_i \le (1 - 1/s^*)^i|\mathcal{L}|$. When $i \ge (\ln|\mathcal{L}| + 1)s^*$, $l_i < 1$. So, FEASIBLE finds at most $s = (\ln|\mathcal{L}| + 1)s^*$ cuboids covering $\mathcal{L}$.

Comparison to $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$. $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$ are two special cases of $\mathcal{K}_{\mathsf{part}}$ with $\mathcal{L}_{\mathsf{pre}} = \mathcal{L}$ and $\mathcal{L}_{\mathsf{pre}} = \{\text{the base cuboid}\}$, respectively. Let θ_{all} and θ_{base} be the max noise in $\mathcal{L}$ for these two choices of $\mathcal{L}_{\mathsf{pre}}$ respectively. It is not hard to see, both when $\theta = \theta_{\mathsf{all}}$ and $s = |\mathcal{L}|$, and when $\theta = \theta_{\mathsf{base}}$ and $s = 1$, FEASIBLE will return a solution. So the one found by Algorithm 1 is at least as good.

Time Complexity. If for some θ, the algorithm FEASIBLE returns a feasible solution for some s, then for any $\theta' \ge \theta$, it also returns a feasible solution for some s'. So from the above comparison to $\mathcal{K}_{\mathsf{all}}$, we can use $\theta_R = \theta_{\mathsf{all}} = 2|\mathcal{L}|^2/\epsilon^2$ as upper bound and apply binary search to find the minimum "feasible" θ. Also, it is not hard to see for any two different cuboid sets $\mathcal{L}_{\mathsf{pre}}$ and $\mathcal{L}_{\mathsf{pre}}'$, we have either $\mathrm{noise}(\mathcal{L}_{\mathsf{pre}}) = \mathrm{noise}(\mathcal{L}_{\mathsf{pre}}')$ or $|\mathrm{noise}(\mathcal{L}_{\mathsf{pre}}) - \mathrm{noise}(\mathcal{L}_{\mathsf{pre}}')| \ge 1/\epsilon^2$, for $\mathrm{mag}(C, C')$ is always an integer for any two cuboids C and C'. So we can stop when $|\theta_L - \theta_R| \le 1/\epsilon^2$. Therefore, overall, lines 2-4 in Algorithm 1 repeats at most $O(\log(|\mathcal{L}|))$ times.

In each iteration of lines 2-4, the algorithm FEASIBLE is called $|\mathcal{L}|$ times. Similar to the SET COVER greedy algorithm, a standard implementation of FEASIBLE needs $O(\sum_{\text{all cuboids } C}|\mathrm{cov}(C)|)$ time (using a linked list of cuboids ordered by their coverage).

$\sum_C |\mathrm{cov}(C)|$ is bounded in two ways: i) There are a total of 2^d cuboids C and each $|\mathrm{cov}(C)| \le |\mathcal{L}|$. ii) For a k-dim cuboid C, $|\mathrm{cov}(C)| \le 2^k$; so $\sum_C |\mathrm{cov}(C)| \le \sum_{k=0}^{d}\binom{d}{k}2^k = 3^d$. So we have $\sum_C |\mathrm{cov}(C)| \le \min\{3^d,\ 2^d|\mathcal{L}|\}$. So the overall running time of Algorithm 1 is $O(\min\{3^d,\ 2^d|\mathcal{L}|\}|\mathcal{L}|\log|\mathcal{L}|)$. □

Proof of Theorem 7.

Approximation Ratio. Suppose the optimal solution is to select s^* cuboids $C_1^*, C_2^*, \ldots, C_{s^*}^*$ as $\mathcal{L}_{\mathsf{pre}}$ such that the number of covered cuboids $|\bigcup_{i=1}^{s^*} \mathrm{cov}(C_i^*, \theta_0, \epsilon, s^*)| = \mathrm{OPT}$ is maximized. Of course, $s^* \le |\mathcal{L}|$. When $s = s^*$ in line 1 of Algorithm 2, GreedyCover$(\mathcal{L}, \theta_0, s^*)$ is called. To prove the promised approximation ratio, we only need to prove GreedyCover will select s^* cuboids $C_1', C_2', \ldots, C_{s^*}'$ (in this order) as $\mathcal{L}_{\mathsf{pre}}$ such that $|\bigcup_{i=1}^{s^*} \mathrm{cov}(C_i', \theta_0, \epsilon, s^*)| \ge (1 - 1/e)\mathrm{OPT}$. The proof for this is similar to the one in the proof of Theorem 6. So for the space limit, we omit it here. The weight version (maximize the weight of cuboids covered by $\mathcal{L}_{\mathsf{pre}}$) is very similar to the above.

Comparison to $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$. The solution offered by $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$ is achieved by Algorithm 2 when $s = |\mathcal{L}|$ and 1, respectively. So the final solution output by Algorithm 2 is at least as good.

Time Complexity. Selecting more than $|\mathcal{L}|$ cuboids into $\mathcal{L}_{\mathsf{pre}}$ does not help, since selecting $\mathcal{L}_{\mathsf{pre}} = \mathcal{L}$ is a better solution than that. So we only consider s from 1 up to $|\mathcal{L}|$. Similar to Feasible, GreedyCover can be implemented in $O(\min\{3^d,\ 2^d|\mathcal{L}|\})$ time. For the weighted version, an additional factor $O(\log(2^d)) = O(d)$ is needed since we need to maintain a priority queue for all cuboids. So the overall running time is $O(\min\{3^d,\ 2^d|\mathcal{L}|\}d|\mathcal{L}|)$. □

Proof of Theorem 8.

Independent Laplace noise $\mathrm{Lap}(|\mathcal{L}_{\mathsf{pre}}|/\epsilon)$ is injected to each cell measure $\mathsf{c}(a)$ to publish $\tilde{\mathsf{c}}(a)$. There are a total of $|\mathcal{E}_{\mathsf{pre}}|$ cells; so from the CDF of Laplace distribution and union bound, with probability $1 - \delta$, none of the cells has the absolute noise $|\tilde{\mathsf{c}}(a) - \mathsf{c}(a)|$ larger than $(|\mathcal{L}_{\mathsf{pre}}|/\epsilon)\log(|\mathcal{E}_{\mathsf{pre}}|/\delta)$. $\mathsf{c}(\cdot)$ is a feasible solution to (6) and $\hat{\mathsf{c}}(\cdot)$ is the optimal. So $\forall a \in \mathcal{E}_{\mathsf{pre}}:\ |\hat{\mathsf{c}}(a) - \mathsf{c}(a)| \le |\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)| + |\mathsf{c}(a) - \tilde{\mathsf{c}}(a)| \le 2|\tilde{\mathsf{c}}(a) - \mathsf{c}(a)| \le (2|\mathcal{L}_{\mathsf{pre}}|/\epsilon)\log(|\mathcal{E}_{\mathsf{pre}}|/\delta)$. The conclusion follows by summing the above inequalities up. □

Proof of Theorem 9.

Lemma 2. (Problem 1.10 in [9]) *Let $X = X_1 + X_2 + \ldots + X_n$ where X_i's are independent and identically distributed with the exponential distribution with parameter $\alpha \in (0, 1)$. With probability at least $1 - \delta$, where $\delta = \left(\frac{\eta}{e^{\eta - 1}}\right)^n$, we have $X \le \eta \mathrm{E}[X]$ ($\eta > 1$).*

The above lemma can be proved by considering the moment generating function. We utilize it to prove Theorem 9 as follows:

For each cell $a \in \mathcal{E}_{\mathsf{pre}}$, since $Y_a = \tilde{\mathsf{c}}(a) - \mathsf{c}(a)$ is a Laplace noise $\mathrm{Lap}(|\mathcal{L}_{\mathsf{pre}}|/\epsilon)$, we have $X_a = |Y_a|$ is distributed with the exponential distribution with parameter $\epsilon/|\mathcal{L}_{\mathsf{pre}}| \in (0, 1)$ if $\epsilon \le |\mathcal{L}_{\mathsf{pre}}|$. Let $X = \sum_{a \in \mathcal{E}_{\mathsf{pre}}} X_a$. We have $\mathrm{E}[X] = |\mathcal{E}_{\mathsf{pre}}||\mathcal{L}_{\mathsf{pre}}|/\epsilon$. From Lemma 2, we have, with probability at least $1 - \delta$, where $\delta = (\frac{\eta}{e^{\eta - 1}})^{|\mathcal{E}_{\mathsf{pre}}|}$, $\sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\tilde{\mathsf{c}}(a) - \mathsf{c}(a)| \le \frac{|\mathcal{E}_{\mathsf{pre}}||\mathcal{L}_{\mathsf{pre}}|}{\epsilon}\eta$. So $\sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\hat{\mathsf{c}}(a) - \mathsf{c}(a)| \le \sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)| + \sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\tilde{\mathsf{c}}(a) - \mathsf{c}(a)|$ $\le \sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\mathsf{c}(a) - \tilde{\mathsf{c}}(a)| + \sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\tilde{\mathsf{c}}(a) - \mathsf{c}(a)| \le \frac{|\mathcal{E}_{\mathsf{pre}}||\mathcal{L}_{\mathsf{pre}}|}{\epsilon}2\eta$. The second inequality above is because $\mathsf{c}(\cdot)$ is a feasible solution to (7) but $\hat{\mathsf{c}}(\cdot)$ is the optimal. Set $\eta = \eta/2$ to complete the proof. □

Proof of Theorem 10.

(i) Correctness. We rewrite (8) as an unconstrained version:

$$\text{minimize } f(\hat{\mathsf{c}}(\cdot)) = \sum_{a \in \mathcal{E}_{\mathsf{pre}}} \left[\left(\sum_{a' \in \mathrm{Base}(a)} \hat{\mathsf{c}}(a')\right) - \tilde{\mathsf{c}}(a)\right]^2.$$

To obtain the optimal solution, setting the gradient of $f(\hat{\mathsf{c}}(\cdot))$ w.r.t. $\hat{\mathsf{c}}(a')$ for each base cell a' to be zero. For each base cell a',

$$\frac{\partial f}{\partial \hat{\mathsf{c}}(a')} = 2 \sum_{a:\ a \in \mathcal{E}_{\mathsf{pre}},\ a' \in \mathrm{Base}(a)} \left[\left(\sum_{a' \in \mathrm{Base}(a)} \hat{\mathsf{c}}(a')\right) - \tilde{\mathsf{c}}(a)\right]$$
$$= 2 \sum_{a \in \mathcal{E}_{\mathsf{pre}} \cap \mathrm{Ancs}(a')} (\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)) = 0,$$

which is equivalent to $$\sum_{a \in \mathcal{E}_{\mathsf{pre}} \cap \mathrm{Ancs}(a')} \hat{\mathsf{c}}(a) = \sum_{a \in \mathcal{E}_{\mathsf{pre}} \cap \mathrm{Ancs}(a')} \tilde{\mathsf{c}}(a). \tag{15}$$

For any cell a'' in a cuboid C'', sum up Equation (15) for all base cells $a' \in \mathrm{Base}(a'')$. On the left-hand side, we have (let Base denote the set of all base cells)

$$\begin{aligned}
\mathrm{est}(a'') &= \sum_{a' \in \mathrm{Base}(a'')} \sum_{a \in \mathcal{E}_{\mathsf{pre}} \cap \mathrm{Ancs}(a')} \hat{\mathsf{c}}(a) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \sum_{a' \in \mathrm{Base}(a'')} \sum_{a \in C \cap \mathrm{Ancs}(a')} \hat{\mathsf{c}}(a) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \sum_{\substack{a':\ a' \in \mathrm{Base},\\ a'[C''] = a''[C'']}} \sum_{\substack{a:\ a \in C,\\ a[C] = a'[C]}} \hat{\mathsf{c}}(a) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \sum_{a \in C} \sum_{\substack{a':\ a' \in \mathrm{Base},\ a[C] = a'[C]\\ a'[C''] = a''[C'']}} \hat{\mathsf{c}}(a) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \sum_{\substack{a:\ a \in C,\\ a[C \wedge C''] = a''[C \wedge C'']}} \sum_{\substack{a':\ a' \in \mathrm{Base},\ a[C] = a'[C]\\ a'[C''] = a''[C'']}} \hat{\mathsf{c}}(a) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \sum_{\substack{a:\ a \in C,\\ a[C \wedge C''] = a''[C \wedge C'']}} \left(\hat{\mathsf{c}}(a) \prod_{A_i \in \mathcal{A} - [C] - [C'']} |A_i|\right) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \sum_{\substack{b:\ b \in (C \wedge C''),\\ b[C \wedge C''] = a''[C \wedge C'']}} \left(\hat{\mathsf{c}}(b) \prod_{A_i \in \mathcal{A} - [C] - [C'']} |A_i|\right) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \sum_{\substack{b:\ b \in (C \wedge C''),\\ b[C \wedge C''] = a''[C \wedge C'']}} \deg(C \vee C'')\hat{\mathsf{c}}(b),
\end{aligned} \tag{16}$$

where $\deg(C) = \prod_{A_i \in \mathcal{A} - [C]} |A_i|$ and $\mathcal{A}$ is the set of all dimensions and $|A_i|$ is the cardinality of dimension A_i.

On the right-hand side, we have

$$\mathrm{obs}(a'') = \sum_{a' \in \mathrm{Base}(a'')} \sum_{a \in \mathcal{E}_{\mathsf{pre}} \cap \mathrm{Ancs}(a')} \tilde{\mathsf{c}}(a) = \sum_{\substack{a:\ a \in C\\ a[C''] = a''[C'']}} \mathrm{obs}(a), \tag{17}$$

where C could be any cuboid that is a descendant of C'' and can be computed from $\mathcal{L}_{\mathsf{pre}}$. So $\mathrm{obs}(a'')$ can be computed recursively.

From Equation (15), we know $\mathrm{est}(a'') = \mathrm{obs}(a'')$. Solving a system of linear equations, we can obtain $\hat{\mathsf{c}}(\cdot)$ as in (9)-(14).

(ii) Time Complexity. We consider how to compute $\hat{\mathsf{c}}(a)$ for all cells here (of course including the ones in $\mathcal{L}$). For each base cell a', we compute $\mathrm{obs}(a')$ as in (9) (recall $\mathrm{Base}(a') = \{a'\}$ for a base cell a'), for all base cells using $O(M|\mathcal{L}_{\mathsf{pre}}|)$ time. For other cells a'' in the data cube, we compute $\mathrm{obs}(a'')$ as in (10), from d-dim cuboids to the 0-dim cuboid, using $O(Nd^2)$ time in total.

Now we compute $\hat{\mathsf{c}}(a)$ from the 0-dim cuboid to d-dim cuboids as in (12)-(14): assuming $\deg(\cdot)$ and $\mathrm{ratio}(\cdot)$ have been precomputed, $\mathrm{aux}(a'')$ (and thus $\hat{\mathsf{c}}(a'')$) can be computed in $O(d|\mathcal{L}_{\mathsf{pre}}|)$ time as for each $C \in \mathcal{L}_{\mathsf{pre}}$, there is only one cell b satisfying the summing-up condition in (13). So in total, we need $O(N(d^2 + d|\mathcal{L}_{\mathsf{pre}}|))$ time to compute $\hat{\mathsf{c}}(a)$ for all cells in the data cube.

(iii) Unbiased. We can prove $\mathrm{E}[\hat{\mathsf{c}}(a)] = \mathsf{c}(a)$ by induction on the dimensionality of a. Recall $\mathrm{E}[\tilde{\mathsf{c}}(a)] = \mathsf{c}(a)$. For the 0-dim cell a_0, we have $\mathrm{aux}(a_0) = 0$, and thus from (9) and (14), $\hat{\mathsf{c}}(a_0)$ is nothing but the weighted average of "different ways to obtain it from cuboids in $\mathcal{L}_{\mathsf{pre}}$". We can show $\mathrm{E}[\hat{\mathsf{c}}(a_0)] = \mathsf{c}(a_0)$ using the linearity of expectation. For an i-dim cell, we first take expectation on both sides of (14), and then use the linearity of expectation and the induction assumption to draw the conclusion.

(iv)-(v) Optimality. For (i) and (iii), $\hat{\mathsf{c}}(\cdot)$ is the *ordinary least squares estimator* (minimizing L^2 norm (8)) and unbiased. Independent noise with identical variance is injected. So the promised properties follow from the Gauss-Markov theorem [33]. □