PRIVACY-PRESERVING DATA PUBLISHING AND ANALYTICS USING DATA CUBES

BY

BOLIN DING

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

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 thesis, we address several problems about privacy-preserving publishing of data cubes using differential privacy or its extensions, which provide privacy guarantees for individuals by adding noise to query answers. The first problem is about how to improve the data quality in privacy-preserving data cubes. Our noise-control frameworks choose noise source in a data cube, i.e., an initial subset of cuboids to compute directly from the fact table with certain amount of noise to be injected to each of them, and then compute the remaining cuboids from them. We show that it is NP-hard to choose proper noise source for certain noise-control objectives, but provide efficient approximation algorithms. The second problem is about how to enforce consistency in the published cuboids. We proposed several approaches with provable guarantee on the noise bound and one of them can even improve the utility of differentially private cuboids (reducing error). The third problem is about how to calibrate noise in data cubes subject to certain exact background knowledge while we are trying to improve the data quality. The notation of generic differential privacy is applied, and we generalize its properties to plug it into our noise-control frameworks for handling background knowledge. Techniques proposed in this thesis provide advanced principles and major parts of a complete solution towards privacy-preserving publishing of data cubes.

To Jieqiu and my parents.

Acknowledgments

First and foremost, I would like to express my greatest thanks to my advisor, Professor Jiawei Han, for his continued guidance, support, and encouragement during my Ph.D. study. I am so fortunate to be leaded into the area of data mining by him. I am truly grateful for his vision and direction about research, inspiration, and perfect personality as advisor.

Also I would like to show my deep gratitude to other doctoral committee members, Professor Ashwin Machanavajjhala, Professor Marianne Winslett, and Professor Chengxiang Zhai, for their invaluable help on my research and constructive suggestions on the dissertation.

Many thanks to my mentors at Microsoft Research, Dr. Arnd Christian König, Dr. Vivek Narasayya, Dr. Surajit Chaudhuri, and Dr. Haixun Wang. I learned a lot from them and gained a lot of valuable experiences, which are important to my Ph.D. research.

I also thank Liangliang Cao, Jing Gao, Ruoming Jin, Xin Jin, Zhenhui Li, Cindy Xide Lin, David Lo, Luiz Mendes, Nikunj Oza, Yongxin Tong, Ashok Srivastava, Lu Su, Yintao Yu, Bo Zhao, and Feida Zhu for enjoyable collaborations over the years.

Last but not least, my heartfelt appreciation goes to my wife Jieqiu, my parents, and my whole family. I can always feel their love, inspiration, bless, and support behind me.

Table of Contents

List of Figures

Chapter 1

Introduction

Data cubes play an essential role in multidimensional data analysis and fast OLAP queries. Simply put, a *data cube* of a *fact table* (i.e. a multidimensional table) consists of a collection of *cuboids*, where each cuboid is generated by grouping records in the table by a subset of dimensions. The publishing of data cubes can facilitate and speed up all kinds of data mining algorithms. However, as the underlying data often is sensitive, publishing all or part of a data 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 and high-utility data cubes that also preserve individuals' privacy. This thesis studies several interweaving fundamental problems of data privacy in data cubes, including how to publish a data cube in such a way that it can be browsed by users and utilized by different data mining algorithms without endangering individuals' privacy, how to improve the data quality in such privacy-preserving data cubes, and how to guarantee individuals' privacy information even if adversaries have part of the data cube in their background knowledge.

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. For example, consider the fact table in Figure 1.1(a) with three dimensions, to be aggregated with *count* measure c. As in Figures 1.1(b)-1.1(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.

We aim to preserve data privacy so that individual level properties of the data are not disclosed to adversaries. The definition of *privacy* has needed to become stronger, partly because abundant data is now available from more and more sources on the Internet (adversaries have more *background knowledge*), and computers become more and more powerful (adversaries have stronger *computational power*). For example, according to the definition of privacy (or security) proposed in 1980s [13, 17], individuals' data is said to be secure if it cannot be computed from a linear combination of rows in the released table. However, because of the above two reasons, such protection of data privacy is far from sufficient nowadays, as shown in the following examples in data cubes.

With background knowledge, an adversary can easily infer sensitive information about an individual from a published data cube [33, 42, 78]. For example, in the data cube in Figure 1.1, if we know that Alice is aged 31-40 and is in the table, the count measure c 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 $(\mathsf{Sex}, \mathsf{Age}, \mathsf{Salary})$, and we publish two cells $(*,\ *,\ \text{10-50k})$ and $(*,\ *,\ \text{50-200k})$, 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, it can be inferred that Bob's salary is 10-50k.

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
$*$	...	...	...

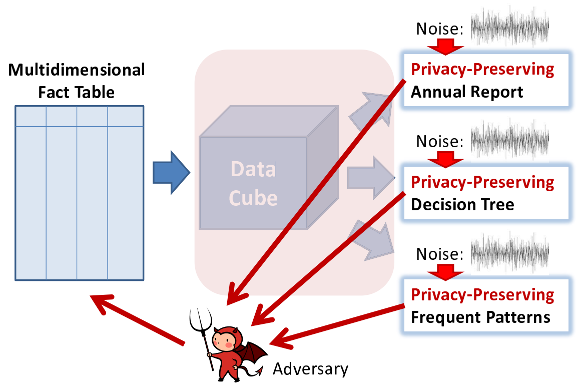
(d) Cuboid {Age, Salary}

Figure 1.1: Fact Table and a *count* data cube

We apply the notion of ϵ-*differential privacy* [21] in data cube publishing. Compared to previous techniques for privacy-preserving data publishing (see [1, 32] for surveys), differential privacy makes very conservative assumptions about the adversary's background knowledge and computational power. It guarantees privacy against adversaries with arbitrary amounts of knowledge about each individual, as long as the data of individuals are independent of each other [44]. In particular, mechanisms satisfying this definition guarantee privacy against adversaries who may know every row in the database except one.

Informally, differential privacy guarantees that the presence/absence or specific value of any particular individual's record has little effect 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 differential privacy is to add random noise to query results, called the Laplacian mechanism [21]. In this mechanism, the noise is carefully calibrated to the query's *sensitivity*, which measures the total change of the query output when one individual record/tuple is deleted or added in the database. For example, if the query is to count how many rows satisfying property P, its sensitivity is one, because deleting/adding one row


Multidimensional Fact Table
Data Cube
Noise:
Privacy-Preserving Annual Report
Privacy-Preserving Decision Tree
Privacy-Preserving Frequent Patterns
Adversary
(a) Privacy-preserving data mining algorithms

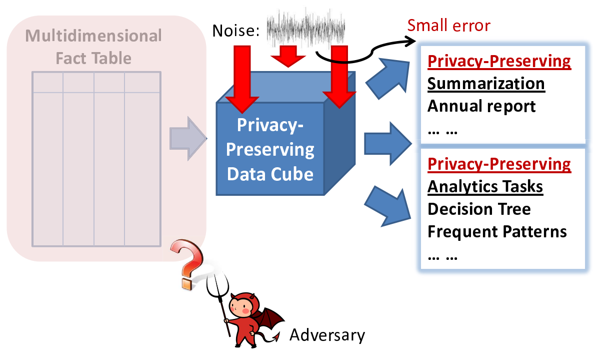

Multidimensional Fact Table
Noise:
Privacy-Preserving Data Cube
Small error
Privacy-Preserving Summarization Annual report … …
Privacy-Preserving Analytics Tasks Decision Tree Frequent Patterns … …
Adversary
(b) Privacy-preserving data cube

Figure 1.2: Advantage of using privacy-preserving data cubes for data analytics

changes the count by at most one. As the variance of the noise increases, the privacy guarantee becomes stronger, but the utility of the result drops. Another way is exponential mechanism, where different outcomes are sampled according to noisy scoring functions [62].

There is a line of works which directly apply the notion of differential privacy in data mining algorithms (e.g. constructing decision trees [31] and mining frequent patterns [8]). The deficiency of such works is depicted in Figure 1.2(a): A data analyst may run different data mining algorithms on the same table. Although each data mining algorithm is claimed to be privacy-preserving independently, the adversary can combine results from different algorithms to enhance their background knowledge about the fact table. Such background knowledge reveals correlation inside the fact table, and thus could be more dangerous and make individuals' privacy more vulnerable. For example, each algorithm reports the number of rows satisfying property P in its output, with a sample drawn from some noise distribution inserted into this number to preserve privacy; each algorithm is privacy-preserving by itself; however, if the adversary averages the answers from multiple algorithms, the expected squared error can be reduced significantly, with individuals' privacy endangered.

Our proposal is depicted in Figure 1.2(b). We first publish a data cube in a privacy-preserving way. This data cube can be reused by infinite number of data mining or processing algorithms. As long as these algorithms do not access the real fact table, the same privacy guarantee is preserved from the data cube to these algorithms. The adversary cannot benefit from combining results from different algorithms (Theorems 2.1.3 and 5.2.5). So **the first question is:** *how to improve the data quality in privacy-preserving data cubes.*

To publish an ϵ–differentially private data cube over d dimensions, there are two natural methods. (i) We can compute the count measure from the fact table, and then add noise to each cell in each cuboid. There are 2^d cuboids and modifying the value of one individual's record could change the count of some cell in every cuboid by 1. So the sensitivity (total change) is 2^d, and according to [26], each cell needs Laplace noise $\mathrm{Lap}(2^d/\epsilon)$[1]. This results in noise with zero mean and variance equal to $2 \cdot 4^d/\epsilon^2$ added to every count. For reasonable values of d, the variance is too large, which destroys the utility of the data cube. The same idea is applied in one of the two approaches proposed in [5] to publish a set of marginals of a contingency table. (ii) Or, we can directly add noise to the fact table, or the *base cuboid* ({Sex, Age, Salary} in Figure 1.1(c)). Then $\mathrm{Lap}(1/\epsilon)$ suffices. We compute the other cuboids from the noisy base cuboid to ensure differential privacy. However, noise in high-level cuboids, such as {Salary}, will be magnified significantly, and thus the utility will be low. For example, if a cell in a high-level cuboid aggregates N cells in the base cuboid, the variance of noise in that cell is magnified by N times as well. This idea can be applied to universal histograms, but noise accumulation also makes them ineffective there [40]. Another possible way is to treat each cell as a query, and apply methods in [51] to answer a workload of count queries while ensuring differential privacy. 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 browse measures across differentially private cuboids, the sums may not match the totals recorded in other higher-level cuboids, because independent noise is inserted into different cells. According to a Microsoft user study [47], users are likely to accept these kinds

[1]A random sample from $\mathrm{Lap}(\lambda)$ has expectation equal to 0 and variance equal to $2\lambda^2$.

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 cuboids to be published. Consistency also boosts accuracy of differentially private data publishing in some cases, e.g., in answering one-dimensional range queries [40]. **The second question is:** *how to enforce consistency in differentially private data cubes while retaining or even improving the data quality.*

Some cuboids within a data cube, especially the high-level ones, may be published exactly to both users and adversaries, either (i) because they are background knowledge that can be easily obtained (for example, how many males or females in a company) or (ii) it is required according to policies or requirement of data users. Such exact partial information or background knowledge about a data cube can cause privacy breaches when combined with differential privacy [43]. Intuitively, the more exact information released from a data cube, the more noise needed to be inserted into the rest part to preserve the same level of privacy. **The third question is:** *how to calibrate noise in data cubes subject to certain exact background knowledge while we are trying to improve the data quality.*

Contributions. Towards answering the above three questions about publishing privacy-preserving data cubes, the studies in this thesis can be summarized as follows.

- 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. This framework can be generalized by allowing injecting different amounts of noise into different cuboids in

$\mathcal{L}_{\mathsf{pre}}$. With a larger search space, the generalized framework may potentially improve the precision in a data cube while preserving the same level of privacy, which will be demonstrated both in theory and in experiments later in this thesis.

- We consider several publishing scenarios that fit the needs of data agents, e.g., the Ministry of Health. In the first scenario, a set of cuboids are identified to 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 differentially private 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. In the third scenario, a threshold of noise variance is specified for each cuboid to be released. The goal is to minimize the max relative ratio between the noise variance and the expected threshold in each released cuboid.

  We formalize these scenarios as three optimization problems for the selection of $\mathcal{L}_{\mathsf{pre}}$, as well as the amount of noise in each cuboid of $\mathcal{L}_{\mathsf{pre}}$ in the generalized framework, prove that they are NP-hard, and give efficient algorithms with provable approximation guarantees. For the first problem or the third, the max noise variance or the max relative ratio, respectively, in 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 show how to enforce consistency over a differentially private data cube by computing a consistent measure from the noisy measure released by a differentially private algorithm. We minimize the L^p distance between the consistent measure and the noisy measure. Publishing consistent measures is differentially private, as we do not revisit the fact table when enforcing consistency. The L^p distance from the consistent measure to the real measure is at most doubled compared with the distance from the inconsistent measure to the real measure, for general p. The consistency-enforcing techniques in [5] are similar to our L^∞ version. But we show that the L^1 version yields a much better theoretical bound on error than the L^∞ version. More surprisingly, we show that in the L^2 version, the consistent measure provides even better utility than the original inconsistent noisy measure, based on the theory of (weighted) least squares and the best linear unbiased estimator (BLUE). We provide efficient algorithms to compute such consistent measures (minimize L^2 distance), with running time linear in the number of cells, while previous techniques have at least quadratic running time.

- We study the relationship between the amount of background knowledge an adversary has and the amount of noise needed to be injected in the scenario of data cubes. We consider some cuboids that have to be released exactly as the background knowledge. We generalize results in [43] about using generic differential privacy to handle such background knowledge. Based these studies, we show how to plug generic differential privacy into our noise control frameworks to limit the variance of the noise added to data cubes subject to background knowledge in forms of exact cuboids.

- All of our techniques proposed in this thesis are both proved to be sound in theory and evaluated with real datasets to verify their effectiveness in practice.

Organization. Chapter 2 provides background for data cubes and differential privacy, together a review of related work. Chapter 3 introduces our noise control publishing frameworks, and different optimization algorithms aiming at different objectives. Chapter 4 shows how to enforce consistency across cuboids based on different L^p distances and analyzes their properties. Chapter 5 studies how to handle adversary's background knowledge. All experiments are reported in Chapter 6. Chapter 7 gives conclusion and future directions.

Chapter 2

Background and Related Work

We present background materials and related work of this thesis in this chapter. We start with a formal definition of *differential privacy* [26, 20] and its application in *data cubes* in Section 2.1. We then introduce some related work in Section 2.2, including relevant techniques to apply this notation of privacy in privacy-preserving data publication (Section 2.2.1) and data mining (Section 2.2.2), and some recent progress on extending the notation of differential privacy for more general context or with weaker/stronger privacy guarantee (Section 2.2.3).

2.1 Background and Notations

The notion of differential privacy was originally introduced in [20, 26]. It is general enough to handle data models including unstructured data like search logs and structured data like histograms and graphs, as long as they can be represented as a vector of entries. In this thesis, we focus on data cubes, a modeling and analytical tool for multidimensional data. We will introduce basic concepts of data cubes and differential privacy in this section.

2.1.1 Data Cubes

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

The *data cube* [36] of a fact table T can be considered as the projections of T onto subset of dimensions. A data cube consists of *cells* and *cuboids*. More formally, a cell a takes the form $(a[1], a[2], \ldots, a[d])$, where $a[i] \in (A_i \cup \{*\})$ denotes the i^{th} dimension's value for this cell, and it is associated with certain aggregate measure of interest. In this thesis, we first focus on the *count measure* c and discuss how to handle other measures like the *summation measure* sum (sum of values associated with all rows in a cell) and the *average measure* avg (average of values associated with all rows in a cell) in Section 3.4.2. The count measure $\mathsf{c}(a)$ is the number of rows r in T that are aggregated in cell a (with the same values on non-$*$ dimensions of a). Formally, let $\mathsf{c}(a) = |\{r \in T \mid \forall 1 \leq i \leq d,\ r[i] = a[i] \vee a[i] = *\}|$.

A cell a is an *m-dim cell* if values of exactly m dimensions $a[i_1], \ldots, a[i_m]$ are *not* $*$. An *m-dim cuboid C* is specified by m dimensions $[C] = \{A_{i_1},\ \ldots,\ A_{i_m}\}$, and it consists of all m-dim cells a such that $\forall 1 \leq k \leq m$, $a[i_k] \in A_{i_k}$, and $a[j] = *$ for all $j \notin \{i_1, \ldots, i_m\}$. We use C and $[C]$ interchangeably to denote either a cuboid or the set of dimensions specifying this cuboid. A cuboid 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 for a large class of measures, such as *count* c, *summation* sum, and *average* avg, cells in C_1 can be computed from C_2 [36]. The cuboid 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 2.1.1 *Fact table T in Table 1.1 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 2.1 shows the lattice of cuboids of T. Cuboid* {Age, Salary} *can be computed from T. For example, to compute the cell* ($*$, 21-30, 10-50k)*, we refer to the table T for rows* (M, 21-30, 10-50k) *and* (F, 21-30, 10-50k)*, and we find three such rows. 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)*, $\ldots$,* ($*$, 60+, 10-50k)*.*

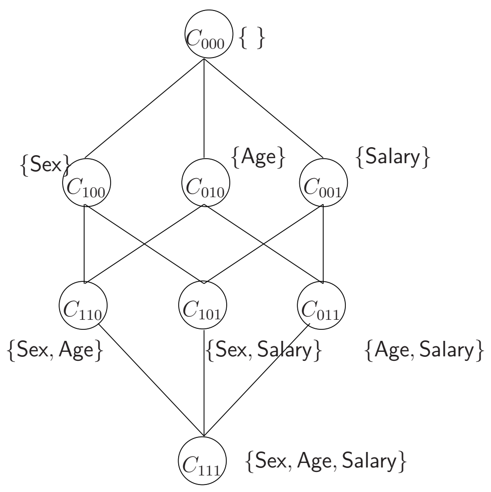


Figure 2.1: Lattice of cuboids in a data cube

2.1.2 Differential Privacy

Differential privacy (*DP* for short in the rest of this thesis) is based on indistinguishability between pairs of neighboring tables. Two tables T_1 and T_2 are *neighbors* if they differ by one row, i.e., inserting or deleting one row from T_1 yields T_2. Formally, let $\mathcal{T}$ be the set of all possible table instances with dimensions $A_1, A_2, \ldots, A_d$, and $\Gamma(T) = \{T' \mid |(T - T') \cup (T' - T)| = 1\} \subseteq \mathcal{T}$ the set of all neighbors of T. Clearly, $T_1 \in \Gamma(T_2)$ implies $T_2 \in \Gamma(T_1)$.

Definition 2.1.1 (Differential privacy [21]) *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 $\mathcal{K}$,*

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

where the probability is taken over the randomness of $\mathcal{K}$ and ϵ is a fixed value.

Consider an individual's concern about the privacy of her/his record r. When the publisher specifies a small ϵ, Definition 2.1.1 ensures that the inclusion or exclusion of r in the fact table makes little difference in the chances of $\mathcal{K}$ returning any particular answer.

Some papers use a slightly different definition of neighbors: 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 [26]). Our algorithms and techniques in this thesis also work with this definition, if we double the noise, because it has been shown that any mechanism satisfies ϵ-indistinguishability if and only if it satisfies 2ϵ-differential privacy.

Tools to Achieve Differential Privacy

There are two popular ways to achieve ϵ-differential privacy, the *Laplacian mechanism* [21] and the *exponential mechanism* [62]. In most (but not all) cases, the Laplacian mechanism is used for data publication, whereas the exponential mechanism is used for optimization problems. And, the Laplace mechanism is a special case of the exponential mechanism.

Throughout this thesis, we use $\mathcal{R}$ to denote the set of all real numbers. Let $F : \mathcal{T} \rightarrow \mathcal{R}^n$ be a function that produces a vector of length n from a table instance. In our context (data cubes), F computes the set of cuboids to be released. In the *Laplacian mechanism*, for a table T, a noisy version $\tilde{F}(T)$ of $F(T)$ is released to preserve differential privacy, and the amount of noise is carefully calibrated according to the sensitivity of F.

Definition 2.1.2 (Sensitivity [21]) *The L^1 global sensitivity of F is:*

$$S(F) = \max_{T_2 \in \mathcal{T}} \left(\max_{T_1 \in \Gamma(T_2)} \|F(T_1) - F(T_2)\|_1 \right),$$

where $\|x - y\|_1 = \sum_{1 \leq i \leq n} |x_i - y_i|$ is the L^1 distance between two n-dimensional vectors $x = \langle x_1, x_2, \ldots, x_n \rangle$ and $y = \langle y_1, y_2, \ldots, y_n \rangle$.

For example, if the function F is to count how many rows satisfying property P, then $S(F) = 1$, because deleting/adding one row in T changes the count $F(T)$ by at most one.

Definition 2.1.3 (Laplacian distribution) *Let* $\mathrm{Lap}(\lambda)$ *denote a sample* Y *taken from a zero-mean Laplace distribution with probability density function* $h(x) = \frac{1}{2\lambda}\exp(-|x|/\lambda)$*. We have* $\Pr[|Y| \geq z] = \exp(-z/\lambda)$*,* $\mathrm{E}[Y] = 0$*, 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* $\langle Y_1, Y_2, \ldots, Y_n\rangle$*, where* $Y_i \sim \mathrm{Lap}(\lambda)$*.*

The following theorem quantifies how much noise to be injected into $F(T)$ is sufficient.

Theorem 2.1.1 (Laplacian mechanism [26]) *Let* $F : \mathcal{T} \to \mathcal{R}^n$ *be a query sequence of length* n *against a table* $T \in \mathcal{T}$*. The randomized algorithm that takes as input table* T *and output* $\tilde{F}(T) = F(T) + \langle \mathrm{Lap}(S(F)/\epsilon)\rangle^n$ *is* ϵ*-differentially private.*

In the *exponential mechanism*, for a given table $T \in \mathcal{T}$, an outcome r from a domain $\mathcal{O}$ is randomly selected according to some scoring function $s : \mathcal{T} \times \mathcal{O} \to \mathcal{R}$ and a base measure $\mu : \mathcal{O} \to \mathcal{R}$. s and μ are public knowledge for everyone including adversaries. To achieve differential privacy, the distribution for selecting (and publishing) r is carefully calibrated with the scoring function s and the base measure μ, as in the next theorem.

Theorem 2.1.2 (Exponential mechanism [62]) *For an input table* $T \in \mathcal{T}$*, the probability of selecting an outcome* $r \in \mathcal{O}$*, is proportional to* $\exp(\epsilon \cdot s(T, r)) \cdot \mu(r)$*. In particular,*

$$\text{the density function} \quad f(r) = \frac{\exp(\epsilon \cdot s(T, r)) \cdot \mu(r)}{\int_{r'} \exp(\epsilon \cdot s(T, r')) \cdot \mu(r')}.$$

Then selecting and publishing r *satisfies* $\epsilon\Delta(s)$*-differential privacy, where*

$$\Delta(s) = \max_{r \in \mathcal{O}} \max_{T_2 \in \mathcal{T}} \max_{T_1 \in \Gamma(T_2)} |s(T_1, r) - s(T_2, r)|.$$

Suppose the quality of outcomes is measured by the scoring function s. Based on the above theorem, [62] shows that, using the exponential mechanism, there are some guarantees on the outcome's quality while the ϵ-differential privacy is preserved.

The *transition* and *composition* properties of differential privacy are frequently used in our and others' research on differentially private algorithms. These two properties allow the combining of the publications or outcomes of several differentially private algorithms.

Theorem 2.1.3 (Transition and composition properties [62, 59]) *Let $\mathcal{K}_i(\cdot)$ or $\mathcal{K}_i(\cdot,\cdot)$ be a randomized algorithm which is ϵ_i-differentially private. Then,*

1. *For a table T, outputting $\mathcal{K}_1(T)$ and $\mathcal{K}_2(T, \mathcal{K}_1(T))$ is $(\epsilon_1 + \epsilon_2)$-differentially private.*

2. *For any input table T and any algorithm $\mathcal{K}$, outputting $\mathcal{K}_1(T)$ and $\mathcal{K}(\mathcal{K}_1(T))$ is ϵ_1-differentially private.*

3. *For any two input tables T_1 and T_2 such that $T_1 \cap T_2 = \emptyset$, outputting $\mathcal{K}_1(T_1)$ and $\mathcal{K}_1(T_2)$ is ϵ_1-differentially private.*

2.2 Related Work

Since ϵ-differential privacy (DP) [26] was introduced, many techniques have been proposed for different data publishing and analysis tasks (refer to [21, 22, 24, 84] for a comprehensive review). We will mainly review three aspects of recent research on developing (differential) privacy-preserved algorithms: the first aspect is about data publishing and query answering algorithms (Section 2.2.1); the second is about data mining algorithms (Section 2.2.2); and the third is extensions to the notion of differential privacy (Section 2.2.3).

2.2.1 Differentially Private Data Publishing

Differential privacy has been applied in various data management and processing systems [3, 4, 11, 59, 81]. In this part, we focus on (differentially) private histograms, count queries, multidimensional data publishing, and batch-query processing techniques, as data cubes are essentially multidimensional count queries in nature, and cuboids to be published can be thought as a batch of queries against the fact table to be answered.

Other mechanism to achieve differential privacy

Some functions F, e.g.., the median of a set of numbers, may have very large sensitivity $S(F)$, e.g., the range of the universe. So the resulting Laplacian mechanism may need to inject too much noise. To achieve differential privacy for a fixed table T, the amount of noise needed to be injected is actually proportional to the *local sensitivity* $S(F, T) = \max_{T' \in \Gamma(T)} \|F(T) - F(T')\|_1$ [66]. However, it may already violate the differential privacy to calibrate noise to $S(F, T)$. So it is proposed in [66] to first obtain an approximate upper bound of $S(F, T)$ (called *smooth sensitivity*) without violating differential privacy, and then calibrate noise to this upper bound. The resulting privacy guarantee, (ϵ, δ)-differential privacy [25], is weaker than ϵ-differential privacy by additive factor δ. Also note that a more direct way to achieve (ϵ, δ)-differential privacy is to add *Gaussian noise* into $F(T)$ [25], and the techniques proposed in this thesis can be also extended for (ϵ, δ)-differential privacy.

For a single counting query, a method to ensure differential privacy (*geometric mechanisms*) [26] has been shown to be optimal under a certain user utility model [34], and the Laplacian mechanism [26] is proved to be nearly optimal.

Recently, Li *et al.* [56] proposes the *compressive mechanism* based on compression technique. The basic idea is to insert noise into the compact synopsis, which is constructed using random projections and has potentially smaller sensitivity than the original table.

Noise complexity of mechanisms for answering linear queries

Hardt and Talwar [39] study the noise complexity of differentially private mechanisms in the setting where the user asks a set of linear queries non-adaptively. The noise complexity means how much noise is necessary and sufficient to make the query answers differentially private. They show that the noise complexity is determined by two geometric parameters associated with the set of queries, and derive nearly matching upper and lower bounds of the amounts of noise needed to make the answer differentially private. This line of research is followed by [38] and [37] to obtain better asymptotic bounds.

Both a one-dimensional histogram and a multidimensional data cube can be considered as sets of linear queries. So the above upper and lower bounds also apply when we want to publish them. But it is possible to utilize the special structure of such queries in histograms or data cubes to further reduce the amount of noise needed. Also, instead of deriving asymptotic bounds of noise, there is a line of research on how to model the amount of noise needed as an objective function and solve an optimization problem to reduce noise in answering batches of linear queries. Those different lines of research will be introduced in the following parts. And it is an interesting and still quite open problem to compare different lines of methods using systematical and comprehensive experiments.

One-dimensional histogram

Histogram can be thought as a one-dimensional data aggregation and is an important tool for answering range queries. Hay *et al.* [40] propose an approach based on a hierarchy of intervals, so-called *universal histogram*. Noise is injected into answers to count queries on these intervals, and these noisy interval can be used to answer all possible range queries. A naive method injects noise directly into the data domain, and answers range queries by aggregating noisy data in the original domain. The error in the answers, if measured by variance of noise, is reduced from $O(m/\epsilon^2)$ in the naive method to $O(\log^3 m/\epsilon^2)$ in the approach of [40]. Xiao *et al.* [80] propose to use the Haar wavelet for range count queries with similar error bound guarantee, which can also handle multidimensional data.

The hierarchy structure in [40] is essentially a balanced binary tree similar to interval tree [14]. A natural idea to reduce error could be optimize this tree structure, for example, by varying the depth or fanout of the tree. However, to guide such optimization, we may need to access the statistics of the origin data, which already violates our privacy guarantee. There are mainly two differentially private ways to do such optimization: the first one is to use exponential mechanism, assigning a score for each possible structure, and selecting the (approximately) best one using randomized algorithms; and the second one borrows the ideas from smooth local sensitivity [66], using a noisy version of the statistics from we need to guide the optimization – accessing such noisy statistics is differentially private, and it is accurate enough for the purpose of optimization. These ideas are used in [67, 82, 83] to reduce error in (multidimensional) histograms for count queries, and used in [79] to optimize parameters in noise injection for reducing relative error. Such differentially private data publication techniques are *data-dependent*. Other techniques that do not rely on statistics about the input database are *data-independent* [84], including [40, 80] and ours.

Multidimensional data aggregation

Some techniques designed for differentially private histogram publication can be extended for multidimensional data aggregation and publication (e.g. [80], [82], and [67]). For example, Xiao *et al.* [80] also extend their wavelet approach to nominal attributes and multidimensional count queries. Some other differentially private techniques are designed particularly for publishing multidimensional data, e.g., [15] and [5] which are introduced below.

Cormode *et al.* [15] consider how to answer multidimensional range queries in a differentially private way. They propose to use a quad-tree structure with Laplacian noise inserted first, and then answer range queries using the noisy quad tree.

Barak *et al.* [5] show how to publish a set of marginals of a contingency table while ensuring differential privacy. Here, marginals of a contingency table are essentially cuboids in our data-cube language. One of their two approaches adds noise to all the cuboids to be published. Since noise is injected into different cuboids independently, there is inconsistency across different noisy cuboids. [5] proposes a method based on linear programming (LP) and rounding, which computes a consistent data cube from the noisy inconsistent one and minimizes their L^∞ distance. At the same time, integrality can be also enforced and negative numbers can be removed in the publishing. In the later part of this thesis, we will show that minimizing the L^1 distance yields a much better theoretical bound on error. The other approach in [5] is similar, but moves to the Fourier domain at first, and add noise to Fourier coefficients. The major reason for moving to the Fourier domain is that the Fourier coefficients are the "sufficient statistics" for marginals of a contingency table, in the sense that if any Fourier coefficient is ignored, there is a free degree of freedom which can lead to unbounded error between true and reported values [60]. But unlike our work, they do not optimize the publishing strategy (the same amount of noise is injected into all cuboids or all Fourier coefficients). Moreover, the number of variables in the linear programming equals the number of cells (often $> 10^6$ in our experiments). So LP-based methods are mainly of theoretical interests and can only handle data cubes with small numbers of cells.

Optimizing noise for batches of linear queries

Linear count queries ask questions like "how many rows in the table satisfying property P." When the database is modeled as a *frequency vector*, a linear count query can be presented as a vector (specifying the property P). Suppose there is a batch of linear count queries to be answered, which is presented as a *query matrix* (with query vectors as its columns), the very basic method is to insert noise into the frequency vector and then compute the multiplication of the noisy frequency vector and the query matrix. There is a line of work [51, 52, 53, 85] on how to reduce error in answers to the given batch of linear queries.

Li *et al.* [51] propose a general framework called *matrix mechanism* to support answering a given workload of count queries while preserving differential privacy. The basic idea is to choose an alternative set of queries, represented as a *strategy matrix*, answer these strategy queries first using the Laplacian mechanism, and compute each query in the input workload using the noisy answers to the strategy queries. The strategy matrix is carefully selected so that every input query can be answered and the total error (sum-square error) is minimized. They [52] propose an efficient algorithm to select a near-optimal strategy matrix, and also give a lower bound of the total error one can get using the matrix mechanism [53]. While [80] and [40] can be unified in the framework of matrix mechanism, the specific algorithms given in [40, 80] are more efficient than the matrix multiplication used in [51, 52].

Yuan *et al.* [85] introduce a variant of the matrix mechanism, called *low-rank mechanism*. They propose practical differentially private technique for answering batch queries with high accuracy, based on a low rank approximation of the query matrix. They prove that the accuracy (measured by sum-square error) provided by their techniques is close to the theoretical lower bound for any mechanism to answer a batch of queries under differential privacy. Note that [52] and [85], aiming at the same goal, are developed independently and to be published in the same year, so it will be very interesting to compare their performance.

Cormode *et al.* [16] apply the matrix mechanism to publish data cubes. A data cube can be modeled as a batch of linear queries, each of which corresponds to a cube cell. But as the number of cells is huge, the original matrix mechanism is inapplicable as the matrices involved are too huge to be manipulated. [16] proposes an efficient noise-optimization algorithm and a consistency-enforcing algorithm via grouping matrix entries that correspond to cells in the same cuboid. So the matrices involved become small enough to be handled. Another approach proposed in [16] tries to optimize the amounts of noise to be injected into Fourier coefficients of cube cells, so that the overall noise is minimize. Different from our work, they use sum-square error to measure the overall noise, while we use max-square error.

Comparison to other privacy notations

Agrawal *et al.* [2] study how to support *OLAP queries* (cells or cuboids in data cubes) 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 $\sqrt{|\text{dataset}|}$ [5]. (ρ_1, ρ_2)-privacy is in general not as strong as ϵ-differential privacy. Also, the error incurred by this method [2] 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.

In data publication, differential privacy provides much stronger privacy guarantees than other privacy concepts based on deterministic algorithms do, such as k-anonymity [69, 70, 73, 72], and its extension l-diversity [57] and t-closeness [54]. These privacy-preserving notations and techniques suppress or generalize entries in the table such that groups of tuples appear indistinguishable or uninformative about sensitive attributes. However, these techniques are vulnerable if the adversary has enough background knowledge. [74, 75, 76] study how to specify authorization and control inferences for OLAP queries in the data cube. They aim to detect OLAP queries with either unauthorized accesses or malicious inferences.

2.2.2 Differentially Private Data Mining

Data mining algorithms usually access sensitive information about individuals, such as medical records and search logs. It is an urgent task to develop privacy-preserving techniques so that data owners are willing to share data miners with more data. For example, the notation of differential privacy has been applied to releasing query and click histograms from search logs [35, 45], recommender systems [61], publishing commuting patterns [58], publishing results of machine learning [9, 12, 41, 86], clustering [30, 66], decision tree [31, 64], mining frequent patterns [8], and aggregating distributed time-series [68].

There are two basic ideas to preserve differential privacy in data mining algorithms, based on the Laplacian mechanism and the exponential mechanism, respectively. Using the Laplacian mechanism, data mining algorithms take Laplacian-noise-injected data as the input. Because of the transition and composition property (Lemma 2.1.3), the same level of differential privacy is preserved in the data mining results as in the noisy data. In another way, the exponential mechanism is applied internally in data mining algorithms. When the algorithm needs to branch, it randomizes the branch selection according to some utility function. For example, in decision tree learning, each branch determines which attribute to be split and the utility function could be information gain of splitting an attribute.

2.2.3 Extension of Differential Privacy

We now introduce some variants of differential privacy, which can be applied for more general context, and may have weaker or stronger privacy guarantee. They may also have different assumptions about the adversary's background knowledge (prior belief on the data).

The first one is (ϵ, δ)-*differential privacy* [25], which was already mentioned in Section 2.1.2. It is a weaker variant of ϵ-differential privacy, and the difference is that, in the definition of ϵ-differential privacy (Definition 2.1.1), the inequality is replaced with $\Pr[\mathcal{K}(T_1) \in S] \leq \exp(\epsilon) \times \Pr[\mathcal{K}(T_2) \in S] + \delta$. (ϵ, δ)-differential privacy trades off privacy for utility, via either smoothing local sensitivity or injecting Gaussian noise.

Generic differential privacy proposed in [43] takes background knowledge of adversaries into consideration. It tries to fix bugs in the assumption of differential privacy, by modeling the *neighbors* as two closest tables (or databases) that are consistent with the background knowledge. Intuitively, the more knowledgeable the adversary is, the more noise needs to be injected, and this notation quantifies how much noise is necessary. It provides us some hints on how to select the mysterious parameter ϵ. We will apply this notation in Chapter 5 of this thesis, and thus more details about this notation will be introduced there.

Pan-privacy [29] provides a stronger guarantee that the pan-private algorithms retain the privacy properties even if their internal state becomes visible to adversaries. The study of this notation was initiated in algorithms for data streams [23, 29, 63], as when sketches used in streaming algorithms are hacked by adversaries, previous privacy guarantees will be easily broken. Also event-level pan-privacy and user-level pan-privacy are distinguished in [28]. Also because real system is usually required to continually generate outputs, the study of (pan-)differential privacy under continual observation is initiated in [28].

Pufferfish [44] is a framework to create new privacy notations that are customized to given applications. The Pufferfish framework provides differential privacy-like probabilistic guarantee but with more expressive power in a general context. It allows expertise to specify: i) what secrets about individual users need to be protected, which are expressed as a potential set of secrets and pairs of secrets needed to be discriminated from each other; and ii) adversaries' belief in how the data were generated and their knowledge about the database. In particular, ii) is specified because as discussed in [27, 43], proper assumption about knowledge of adversary may improve the utility of privacy guarantees. Previous differential privacy can be analyzed within the Pufferfish framework, and this framework also enjoys useful (and more general) properties, like composition and self-composition.

Differential identifiability [48] is proposed recently to give more quantitative guidelines on how to select the value of ϵ based on the level of privacy required. It also has implicit assumptions and models about how the data is generated in adversary's belief.

Chapter 3

Optimizing Noise Sources in Data Cubes

Problem Description

Given a d-dimensional fact table T, we aim to publish a subset $\mathcal{L}$ of all cuboids $\mathcal{L}_{\mathsf{all}}$ with measure $\tilde{\mathsf{c}}(\cdot)$, a noisy version of the *count measure* $\mathsf{c}(\cdot)$ in $\mathcal{L}$, 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 differentially private algorithm $\mathcal{K}$. With the level of privacy fixed (i.e. the parameter ϵ in differential privacy is fixed), we want to minimize the error in $\tilde{\mathsf{c}}(\cdot)$, or the difference between $\mathsf{c}(\cdot)$ and $\tilde{\mathsf{c}}(\cdot)$, so that the utility of data is preserved as well.

In this chapter, we will first discuss how to measure the error in publishing algorithms and possible objectives in Section 3.1, and then introduce noise-control publishing algorithms that (approximately) achieve these objectives in Section 3.2 and Section 3.3.

3.1 Measuring Noise and Publishing Objectives

We *measure the utility of an algorithm by the variance of the noisy measure it publishes.* As we apply the Laplacian mechanism in Theorem 2.1.1, we will show noisy measure $\tilde{\mathsf{c}}(\cdot)$ published by our algorithms is unbiased, i.e., the expectation $\mathrm{E}\left[\tilde{\mathsf{c}}(a)\right] = \mathsf{c}(a)$. So for one cell, the *noise variance* $\mathrm{Var}\left[\tilde{\mathsf{c}}(a)\right]$ is equal to the *mean squared error* of the noisy measure $\tilde{\mathsf{c}}(a)$ we publish, and we use it to measure the noise/error in $\tilde{\mathsf{c}}(a)$ for one cell a:

$$\mathrm{Var}\left[\tilde{\mathsf{c}}(a)\right] = \mathrm{E}\left[(\tilde{\mathsf{c}}(a) - \mathrm{E}\left[\tilde{\mathsf{c}}(a)\right])^2\right] = \mathrm{E}\left[(\tilde{\mathsf{c}}(a) - \mathsf{c}(a))^2\right].$$

Noise-control objectives

Ultimately, the differentially private data cube will be used by users or data mining algorithms. So it is better to provide them enough flexibility to specify the noise-control objectives they want to achieve. We consider two publishing scenarios. In the first scenario, users identify a set of cuboids which must be released, and the issue is how to limit the max noise in these cuboids. In the second scenario, a large body of cross-tabulations can be useful. The question is, how many of these cuboids can be released in a differentially private manner, while respecting a certain noise bound for each cell (in expectation)? Following are two basic objectives we aim at to control the noise in the published cuboids:

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

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

These two objectives can be generalized by assigning a priority weight w_a or w_C on each cell a or cuboid C, respectively. For example, some of these cuboids are more important than others, so they have higher weights than others. Also, the cuboids containing relatively smaller measures need to be published with smaller error inserted so that the overall relative error is limited, so they may be assigned with higher weights. Our publishing algorithms introduced later can be easily generalized to the weighted versions of (i) and (ii).

3.2 Optimizing Noise by Source Selection

Before presenting our first noise-optimizing approach $\mathcal{K}_{\mathsf{part}}$, we introduce two straw man approaches, $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$, and discuss why these baseline solutions are not optimal.

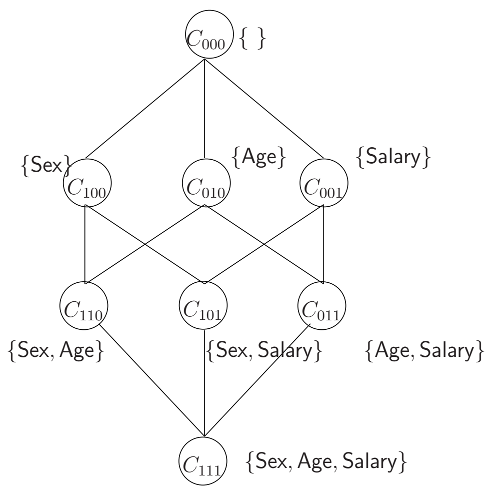
Straw man release algorithms $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$

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 2.1.2), since a row in T contributes 1 to exactly one cell in each cuboid in $\mathcal{L}$. Note that $|\mathcal{L}| = 2^d$ if all the cuboids are to be published. By Theorem 2.1.1, to preserve ϵ-differential privacy, $\mathcal{K}_{\mathsf{all}}$ adds noise drawn from $\text{Lap}(|\mathcal{L}|/\epsilon)$ to each cell–$\tilde{\mathsf{c}}(a) = \mathsf{c}(a) + \text{Lap}(|\mathcal{L}|/\epsilon)$–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 the mean squared error reach $\text{Var}\left[\tilde{\mathsf{c}}(a)\right] = 2 \cdot 2^{16}/\epsilon^2$. Some approaches in [5] and [80] can be adapted in our context with the same asymptotic error bound (e.g., Theorem 3 in [80]) as $\mathcal{K}_{\mathsf{all}}$'s error bound, which is shown below.

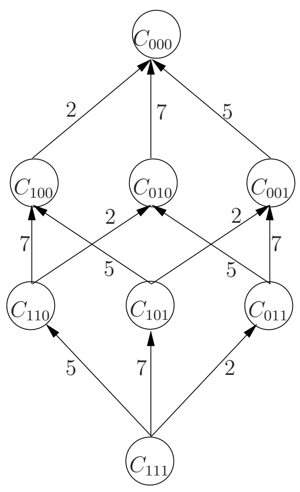
Theorem 3.2.1 $\mathcal{K}_{\mathsf{all}}$ *is ϵ-differentially private. For any cell a to be published, $\tilde{\mathsf{c}}(a)$ is unbiased:* $\text{E}\left[\tilde{\mathsf{c}}(a)\right] = \mathsf{c}(a)$*, and the mean squared error of* $\tilde{\mathsf{c}}(a)$ *is* $\text{Var}\left[\tilde{\mathsf{c}}(a)\right] = 2|\mathcal{L}|^2/\epsilon^2$.

Proof: Adding/deleting a row in T affects the measure $\mathsf{c}(a)$ of exactly one cell in each cuboids by 1. The vector $F_{\mathsf{all}}(T)$, with each entry as $\mathsf{c}(a)$ for each cell of each cuboid in $\mathcal{L}$, has sensitivity $S(F_{\mathsf{all}}) = |\mathcal{L}|$. By Theorem 2.1.1, $\mathcal{K}_{\mathsf{all}}$ is ϵ-differentially private. Since the noise is generated from $\text{Lap}(|\mathcal{L}|/\epsilon)$, we have $\text{E}\left[\tilde{\mathsf{c}}(a)\right] = \mathsf{c}(a)$, $\text{Var}\left[\tilde{\mathsf{c}}(a)\right] = 2|\mathcal{L}|/\epsilon^2$. □

Another option, $\mathcal{K}_{\mathsf{base}}$, computes only the cells in the d-dim cuboid, i.e., the base cuboid, directly from T, and insert Laplace 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 $\tilde{\mathsf{c}}(a) = \mathsf{c}(a) + \text{Lap}(1/\epsilon)$ 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-differential privacy is preserved. However, the noise variance grows quickly as we aggregate more base cells to compute cells in higher levels, as shown below.



(a) Lattice Structure of Cuboids



(b) Variance Magnification

Figure 3.1: Lattice of cuboids and variance magnification of noise

Theorem 3.2.2 $\mathcal{K}_{\mathsf{base}}$ *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* $[C]$*, the mean squared error is* $\mathrm{Var}[\tilde{\mathsf{c}}(a)] = 2\prod_{A_j \notin [C]} |A_j|/\epsilon^2$.

Proof: Since only the base cuboid is computed from T, the sensitivity $S(F_{\mathsf{base}}) = 1$. So from Theorem 2.1.1, publishing $F_{\mathsf{base}}(T) + \langle \mathrm{Lap}(1/\epsilon) \rangle$ satisfies ϵ-differential privacy. 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 [59], is ϵ-differentially private. For a cell a in the base cuboid, $\mathrm{E}[\tilde{\mathsf{c}}(a)] = \mathrm{E}[\mathsf{c}(a) + \mathrm{Lap}(1/\epsilon)] = \mathsf{c}(a)$, and $\mathrm{Var}[\tilde{\mathsf{c}}(a)] = 2/\epsilon^2$. For a cell a in a cuboid C, it aggregates a set $\{a'\}$ of $\prod_{A_j \notin [C]} |A_j|$ cells in the base cuboid. So we have

$$\mathrm{E}[\tilde{\mathsf{c}}(a)] = \mathrm{E}\left[\sum_{a'} \tilde{\mathsf{c}}(a')\right] = \sum_{a'} \mathrm{E}[\tilde{\mathsf{c}}(a')] = \sum_{a'} \mathsf{c}(a') = \mathsf{c}(a),$$

and since noise is generated independently,

$$\mathrm{Var}[\tilde{\mathsf{c}}(a)] = \mathrm{Var}\left[\sum_{a'} \tilde{\mathsf{c}}(a')\right] = \sum_{a'} \mathrm{Var}[\tilde{\mathsf{c}}(a')] = \left(\prod_{A_j \notin [C]} |A_j|\right) \cdot 2/\epsilon^2.$$

□

Example 3.2.1 *For the fact table T in Figure 1.1(a), Figure 3.1(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 $\{\mathsf{Age}, \mathsf{Salary}\}$. The label on an edge from C to C' in Figure 3.1(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 3.1(a). Using $\mathcal{K}_{\mathsf{all}}$, we add Laplace noise $\mathrm{Lap}(8/\epsilon)$ *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* $\mathrm{Lap}(1/\epsilon)$ *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 $\mathrm{Lap}(4/\epsilon)$ *to each (the sensitivity is* 4*). We compute the other cuboids from these, which does not violate differential privacy as we do not touch the fact table. 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}).*

Example 3.2.1 shows that the more initial cuboids we compute from the table T, the higher their sensitivity is (more noise), but the less noise is accumulated when other cuboids are computed from them. $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$ are the two extremes of computing as many or as few cuboids as possible directly from T. There is some strategy between $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$ that gives better bounds for the noise variance of released cuboids. There are two intuitive ideas: the first is to vary the set of initial cuboids ($C_{111}, C_{110}, C_{101}$, and C_{100} in Example 3.2.1); and the second is more general: we can vary the amounts of noise inserted into different initial cuboids. These two ideas will be elaborated in the rest of Sections 3.2 and 3.3, respectively.

3.2.1 Noise Control Framework: Cuboid Selection

We first introduce the framework of a better algorithm $\mathcal{K}_{\mathsf{part}}$ in this subsection. 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 $\mathcal{L}_{\mathsf{pre}} \subseteq \mathcal{L}$. $\mathcal{K}_{\mathsf{part}}$ works as follows.

1. (*Noise Sources*) For each cell a of cuboids in $\mathcal{L}_{\mathsf{pre}}$, $\mathcal{K}_{\mathsf{part}}$ computes $\mathsf{c}(a)$ from T but releases $\tilde{\mathsf{c}}(a) = \mathsf{c}(a) + \mathrm{Lap}(s/\epsilon)$, where the sensitivity $s = |\mathcal{L}_{\mathsf{pre}}|$. $\mathcal{L}_{\mathsf{pre}}$ is selected by our algorithms in Section 3.2.2 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., $\mathrm{E}[\tilde{\mathsf{c}}(a)] = \mathsf{c}(a)$. For a cell a in cuboid $C' \in \mathcal{L}_{\mathsf{pre}}$, $\mathrm{Var}[\tilde{\mathsf{c}}(a)] = 2s^2/\epsilon^2$ ($s = |\mathcal{L}_{\mathsf{pre}}|$). 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

$$\mathrm{mag}(C, C') = \prod_{1 \le i \le q} |A_{k_i}|. \tag{3.1}$$

So the noise variance, the mean squared error, of $\tilde{\mathsf{c}}(a)$ is

$$\mathrm{Var}[\tilde{\mathsf{c}}(a)] = \mathrm{mag}(C, C') \cdot 2s^2/\epsilon^2. \tag{3.2}$$

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

$$\mathrm{noise}(C, \mathcal{L}_{\mathsf{pre}}) = \min_{C' \in \mathcal{L}_{\mathsf{pre}}} \mathrm{mag}(C, C') \cdot 2s^2/\epsilon^2 \tag{3.3}$$

be the smallest possible noise variance when computing C from a single cuboid in the selected cuboid set $\mathcal{L}_{\mathsf{pre}}$. In a special case, if $C' \in \mathcal{L}_{\mathsf{pre}}$, $\mathrm{noise}(C', \mathcal{L}_{\mathsf{pre}}) = 2s^2/\epsilon^2$.

Theorem 3.2.3 *$\mathcal{K}_{\mathsf{part}}$ is ϵ-differentially private. For any released cell a,* $\mathrm{E}[\tilde{\mathsf{c}}(a)] = \mathsf{c}(a)$.

Proof: The proof is similar to the one of Theorem 3.2.2. Consider the 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 $s = |\mathcal{L}_{\mathsf{pre}}|$ (since the measure of exactly one cell in each cuboid in $\mathcal{L}_{\mathsf{pre}}$ is changed by ± 1 if adding/deleting a row in T). So by Theorem 2.1.1 and the composition property of differential privacy, 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-differentially private. $\mathrm{E}[\tilde{\mathsf{c}}(a)] = \mathsf{c}(a)$ is also from the linearity of expectation. □

Example 3.2.2 *Consider the data cube in Example 2.1.1 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 $\mathrm{mag}(C_{000}, C_{100}) = 2$; or from C_{110} with noise variance $14 \times 32/\epsilon^2 = 448/\epsilon^2$, since $\mathrm{mag}(C_{000}, C_{110}) = 2 \times 7 = 14$. So $\mathcal{K}_{\mathsf{part}}$ chooses the best among all cuboids in $\mathcal{L}_{\mathsf{pre}}$, and computes C_{000} from C_{100}. And we have $\mathrm{noise}(C_{000}, \mathcal{L}_{\mathsf{pre}}) = 64/\epsilon^2$.*

$\mathcal{K}_{\mathsf{part}}$ shares some similarities to the *matrix mechanism* in [51]. $\mathcal{K}_{\mathsf{part}}$ chooses cuboids $\mathcal{L}_{\mathsf{pre}}$ to compute $\mathcal{L}$, while [51] chooses a set of queries to answer a given workload of count queries. If we adapt [51] 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 [51] is not directly applicable in our context.

Efficient Implementation of $\mathcal{K}_{\mathsf{part}}$. Suppose $\mathcal{L}_{\mathsf{pre}}$ is given. A naive way to compute differentially private cuboids in $\mathcal{L}$ is to inject noise into each cuboid in $\mathcal{L}_{\mathsf{pre}}$. Then for each

$C \in \mathcal{L}$, query its nearest descendant $C^* \in \mathcal{L}_{\mathsf{pre}}$, and aggregate cells in C^* to compute C.

We can compute differentially private 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 differential privacy, and C'' is computed from C^*. Then computing C from C^* is equivalent to computing it from C'', with differential privacy ensured. So after noise is injected into all cuboids in $\mathcal{L}_{\mathsf{pre}}$, differentially private 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.

Cuboid selection problems: optimizing noise source $\mathcal{L}_{\mathsf{pre}}$

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) *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. $\sum_{C \in \mathcal{L}:\ \text{noise}(C, \mathcal{L}_{\mathsf{pre}}) \leq \theta_0} w(C)$ 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 3.2.4 *The two cuboid selection problems* BOUND MAX *and* PUBLISH MOST *are both NP-hard, if we treat $|\mathcal{L}|$ as the input size.*

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

Construction of Instances. 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 $\text{cov}(v)$ be the edges incident on a vertex v, then we want to decide if there is $V' \subseteq V$ such that $\cup_{v \in V'} \text{cov}(v) = E$ and $|V'| \leq m$.

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 $|E| > 2n/2 = n$.

Now we can construct an instance of the Bound Max problem from the above instance of Vertex Cover problem, correspondingly:

(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 $[C_i^1] = \{B_i\}$ (not related to the graph).

(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 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) (not related to the graph).

(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 Bound Max, we want to publish all the 1-dim, 3-dim, and 4-dim cuboids in (i)-(v), denoted by $\mathcal{L}$. And the questions is 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 $8(3n+m)^2/\epsilon^2$, i.e., noise$(\mathcal{L}_{\mathsf{pre}}) \leq 8(3n+m)^2/\epsilon^2$.

In the following part, we prove the Vertex Cover instance is YES if and only if the Bound Max instance is YES, to complete our proof of the NP-hardness.

One Direction "$\Rightarrow$"

This direction is easy. If there is a vertex cover V' with size m, create $\mathcal{L}_{\mathsf{pre}}$ as follows: We select all the $2n$ 4-dim cuboids in (v) and all the n 3-dim cuboids in (iv) into $\mathcal{L}_{\mathsf{pre}}$. Among 3-dim cuboids in (iii), for each v in the vertex cover V', we select the cuboid C_v^3 into $\mathcal{L}_{\mathsf{pre}}$. So totally, we have $3n+m$ cuboids C in $\mathcal{L}_{\mathsf{pre}}$. We can prove for any cuboid in $\mathcal{L}$, i.e., the ones (i)-(v), we have $\mathrm{mag}(C, C') \leq 4$ for at lease one $C' \in \mathcal{L}_{\mathsf{pre}}$. For any other cuboid C_v^3 in (iii) but not in $\mathcal{L}_{\mathsf{pre}}$, we have a cuboid $C_v^4 \in \mathcal{L}_{\mathsf{pre}}$ in (v) covering C_v^3 such that $\mathrm{mag}(C_v^3, C_v^4) = 4$, since the dimension D_v has cardinality 4; for any cuboid C_i^1 in (ii), we have a cuboid $C_x^3 \in \mathcal{L}_{\mathsf{pre}}$ in (iv) covering C_i^1 ($i = x_1$, x_2, or x_3) such that $\mathrm{mag}(C_i^1, C_x^3) = 2 \times 2 = 4$, since each dimension B_i has cardinality 2; for any cuboid C_e^1 in (i), since V' is a vertex cover, we have a cuboid C_v^3 ($v \in V'$) such that C_v^3 covers C_e^1 and $\mathrm{mag}(C_e^1, C_v^3) = 2 \times 2 = 4$, since each dimension A_e has cardinality 2. Since $|\mathcal{L}_{\mathsf{pre}}| = 3n+m$, we add a random noise $\mathrm{Lap}((3n+m)/\epsilon)$ to each cell of a cuboid in $\mathcal{L}_{\mathsf{pre}}$. So, noise$(\mathcal{L}_{\mathsf{pre}}) \leq 4 \cdot 2(3n+m)^2/\epsilon^2 = 8(3n+m)^2/\epsilon^2$.

The Other Direction "$\Leftarrow$"

Now we prove: If there is a set $\mathcal{L}_{\mathsf{pre}}$ of cuboids such that cuboids in $\mathcal{L}$ can be computed from $\mathcal{L}_{\mathsf{pre}}$ and noise$(\mathcal{L}_{\mathsf{pre}}) \leq 8(3n+m)^2/\epsilon^2$, there is a vertex cover V' in G such that $|V'| \leq m < n$. Define $\mathrm{mag}(C, \mathcal{L}_{\mathsf{pre}}) = \min_{C' \in \mathcal{L}_{\mathsf{pre}}} \mathrm{mag}(C, C')$ to be the minimum variance magnification ratio if C can be computed from some cuboid in $\mathcal{L}_{\mathsf{pre}}$. Define $\mathrm{mag}(C, \mathcal{L}_{\mathsf{pre}}) = \infty$ if C cannot be computed from $\mathcal{L}_{\mathsf{pre}}$. Then noise$(\mathcal{L}_{\mathsf{pre}}) = \max_{C \in \mathcal{L}} \mathrm{mag}(C, \mathcal{L}_{\mathsf{pre}}) \cdot 2|\mathcal{L}_{\mathsf{pre}}|^2/\epsilon^2$.

Structure analysis.

To prepare for case analysis below, we want prove a collection of structural results like: "for any two 4-dim cuboids C_u^4 and C_v^4 in (v), if a cuboid C' covers both C_u^4 and C_v^4, we must have $\mathrm{mag}(C_u^4, C')$, $\mathrm{mag}(C_v^4, C') \geq 16$." For any two 4-dim cuboids C_u^4 and C_v^4 with $u, v \in V$ in (v), we must have the intersection of their dimensions $[C_u^4] \cap [C_v^4]$ contains at most one dimension, since two vertices u and v in G have at most one common edge (if either u or v is not in V, the intersection is empty). So for a cuboid C' covering both C_u^4 and C_v^4 (e.g. $[C'] = [C_u^4] \cup [C_v^4]$), C' must have at least 7 dimensions. Moreover, $\mathrm{mag}(C_u^4, C'), \mathrm{mag}(C_v^4, C') \geq 2^2 \cdot 4^1 = 16$, because C' has 5 dimensions with cardinality 2 and 2 dimensions with cardinality 4 while C_u^4 or C_v^4 has 3 dimensions with cardinality 2 and 1 dimension with cardinality 4.

Similarly, for any two 3-dim cuboids C_u^3 and C_v^3 in (iii), a cuboid C' covering both of them must have $\mathrm{mag}(C_u^3, C')$, $\mathrm{mag}(C_v^3, C') \geq 4$. For any two 3-dim cuboids C_x^3 and C_y^3 in (iv), a cuboid C' covering both of them must have $\mathrm{mag}(C_x^3, C')$, $\mathrm{mag}(C_y^3, C') \geq 8$.

Case analysis of $\max_{C \in \mathcal{L}} \mathrm{mag}(C, \mathcal{L}_{\mathsf{pre}})$.

The goal of the case analysis is to show that, if either $\max_{C \in \mathcal{L}} \mathrm{mag}(C, \mathcal{L}_{\mathsf{pre}}) < 4$ or > 4, it is impossible to select $\mathcal{L}_{\mathsf{pre}}$ with $\mathrm{noise}(\mathcal{L}_{\mathsf{pre}}) \leq 8(3n+m)^2/\epsilon^2$.

1) If $\max_{C \in \mathcal{L}} \mathrm{mag}(C, \mathcal{L}_{\mathsf{pre}}) = 1$, all cuboids in $\mathcal{L}$ have to be selected into $\mathcal{L}_{\mathsf{pre}}$. There are $|E|$ ($> n$) 1-dim cuboids in (i), $3n$ 1-dim cuboids in (ii), n 3-dim cuboids in (iii), n 3-dim cuboids in (iv), $2n$ 4-dim cuboids in (v). There are a total of $|E| + 7n$ ($> 8n$) cuboids in (i)-(v). So $\mathrm{noise}(\mathcal{L}_{\mathsf{pre}}) > 2(8n)^2/\epsilon^2 > 8(3n+m)^2/\epsilon^2$.

2) If $\max_{C \in \mathcal{L}} \mathrm{mag}(C, \mathcal{L}_{\mathsf{pre}}) = 2$: From the structure discussion, all the $2n$ 3-dim cuboids C in (iii) and (iv) and all the $2n$ 4-dim cuboids C in (v) must be selected into $\mathcal{L}_{\mathsf{pre}}$, since there is no cuboid C' covering two of them with $\mathrm{mag}(C, C') \leq 4$. To cover all the 1-dim cuboids in (i), we need at least n 2-dim cuboids (in the best case, their dimensions are disjoint). To cover all the 1-dim cuboids in (ii), we need at least $2n$ 2-dim cuboids to be in $\mathcal{L}_{\mathsf{pre}}$. So we have $|\mathcal{L}_{\mathsf{pre}}| \geq 6n$, and thus $\mathrm{noise}(\mathcal{L}_{\mathsf{pre}}) \geq 2 \cdot 2(6n)^2/\epsilon^2 = 144n^2/\epsilon^2 > 8(3n+m)^2/\epsilon^2$.

3) If $\max_{C\in\mathcal{L}} \operatorname{mag}(C, \mathcal{L}_{\mathsf{pre}}) = 4$: All $2n$ 4-dim cuboids in (v) must be selected into $\mathcal{L}_{\mathsf{pre}}$. We will not choose to use any 2-dim cuboid to cover a 1-dim cuboid in (i)-(ii), since a 3-dim cuboid in (iii) or (iv) covers more. For a similar reason, we will not choose any 3-dim cuboid other than the ones in (iii) and (iv). For any 1-dim cuboid in (i) or (ii), we need to select either itself into $\mathcal{L}_{\mathsf{pre}}$ or some 3-dim cuboid in (iii) or (iv) into $\mathcal{L}_{\mathsf{pre}}$ to cover it; to minimize $|\mathcal{L}_{\mathsf{pre}}|$ (and thus $\operatorname{noise}(\mathcal{L}_{\mathsf{pre}})$), we always select 3-dim cuboids in (iii) and (iv) into $\mathcal{L}_{\mathsf{pre}}$. The n 3-dim cuboids in (iv) must be all selected into $\mathcal{L}_{\mathsf{pre}}$ to cover the $3n$ 1-dim cuboids in (ii). Suppose m' ($\leq n$) cuboids from (iii) are selected to cover all 1-dim cuboids in (i), these m' cuboids corresponds to a vertex cover V' for G ($|V'| = m'$), and $\operatorname{noise}(\mathcal{L}_{\mathsf{pre}}) = 4 \cdot 2|\mathcal{L}_{\mathsf{pre}}|^2/\epsilon^2 = 8(3n + m')^2/\epsilon^2$.

4) If $\max_{C\in\mathcal{L}} \operatorname{mag}(C, \mathcal{L}_{\mathsf{pre}}) = 8$: Again, all $2n$ 4-dim cuboids in (v) must be selected into $\mathcal{L}_{\mathsf{pre}}$. There are totally at least $4n$ 1-dim cuboids uncovered by them in (i) and (ii); to cover all of them, we need at least n 4-dim cuboids to be in $\mathcal{L}_{\mathsf{pre}}$. Therefore, $\operatorname{noise}(\mathcal{L}_{\mathsf{pre}}) \geq 8 \cdot 2(3n)^2/\epsilon^2 = 144n^2\epsilon^2 > 8(3n + m)^2/\epsilon^2$.

5) If $\max_{C\in\mathcal{L}} \operatorname{mag}(C, \mathcal{L}_{\mathsf{pre}}) = 16$: We can choose $\mathcal{L}_{\mathsf{pre}}$ to be the set of all $2n$ 4-dim cuboids in (v) to cover $\mathcal{L}$; however, as $m < n$, we have $\operatorname{noise}(\mathcal{L}_{\mathsf{pre}}) > 8(3n+m)^2/\epsilon^2$. Choosing a 7-dim cuboid to cover two 4-dim cuboids and discarding the two 4-dim ones does not help, since it leaves at least two 3-dim cuboids uncovered. So we still have $\operatorname{noise}(\mathcal{L}_{\mathsf{pre}}) > 8(3n + m)^2/\epsilon^2$.

6) If $\max_{C\in\mathcal{L}} \operatorname{mag}(C, \mathcal{L}_{\mathsf{pre}}) \geq 32$: We have $|\mathcal{L}_{\mathsf{pre}}| < \sqrt{2}n$, since otherwise $\operatorname{noise}(\mathcal{L}_{\mathsf{pre}}) > 8(3n + m)^2/\epsilon^2$. The idea is similar to the above case "$\max_{C\in\mathcal{L}} \operatorname{mag}(C, \mathcal{L}_{\mathsf{pre}}) = 16$". To cover the $2n$ 4-dim cuboids in (v) with fewer than $\sqrt{2}n$ cuboids, we have to choose some 7-dim cuboids or 8-dim cuboids, but it leaves more 3-dim and 1-dim cuboids uncovered, which could not help reduce $\operatorname{noise}(\mathcal{L}_{\mathsf{pre}})$.

Any case with $\max_{C\in\mathcal{L}} \operatorname{mag}(C, \mathcal{L}_{\mathsf{pre}})$ valued between two cases k) and k+1) above can be analyzed in the same way as case k).

From the above case analysis, if there is a set of cuboids $\mathcal{L}_{\mathsf{pre}}$ such that cuboids in $\mathcal{L}$ can be computed from $\mathcal{L}_{\mathsf{pre}}$ and $\text{noise}(\mathcal{L}_{\mathsf{pre}}) \leq 8(3n+m)^2/\epsilon^2$, the only case is "$\max_{C\in\mathcal{L}} \text{mag}(C, \mathcal{L}_{\mathsf{pre}}) = 4$". So we have $m' \leq m$, implying there is a vertex cover in G with $|V'| \leq m$.

The two directions "$\Rightarrow$" and "$\Leftarrow$" completes our proof. $\square$

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 still open.

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.

3.2.2 Optimization Algorithms for Cuboid Selection

A brute force approach for cuboid selection 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 3.2.4, we introduce approximation algorithms for these two problems, with running time polynomial in 2^d.

Bounding the Maximum Variance

We now present a $(\ln(|\mathcal{L}|) + 1)^2$-approximation algorithm for the Bound Max 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 $\mathcal{L}_{\mathsf{pre}}$ s.t. for all $C \in \mathcal{L}$, $\text{noise}(C, \mathcal{L}_{\mathsf{pre}}) \leq \theta$? Let Feasible$(\mathcal{L}, \theta, s)$ return $\mathcal{L}_{\mathsf{pre}}$ 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 $s = 1, \dots, |\mathcal{L}|$ 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 cov($C$) for each cuboid based on $\theta$ and $s$;
7: $\mathcal{R} \leftarrow \emptyset,\ \ \mathcal{COV} \leftarrow \emptyset$;
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 $\mathcal{L}_{\mathsf{pre}}$; else return NO.
```
Algorithm 1: Algorithm for Bound Max problem

Theorem 3.2.4 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 (3.3), we rewrite Feasibility's noise constraint as:

$$\mathrm{noise}(C, \mathcal{L}_{\mathsf{pre}}) \le \theta \Leftrightarrow \min_{C' \in \mathcal{L}_{\mathsf{pre}}} \mathrm{mag}(C, C') \le \tfrac{\theta\epsilon^2}{2s^2}. \tag{3.4}$$

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:

$$\mathrm{cov}(C', \theta, \epsilon, s) = \{C \in \mathcal{L} \mid C \preceq C',\ \mathrm{mag}(C, C') \le \tfrac{\theta\epsilon^2}{2s^2}\}. \tag{3.5}$$

For simplicity, we write $\mathrm{cov}(C')$ for $\mathrm{cov}(C', \theta, \epsilon, s)$ when θ, ϵ, and s are fixed.

Lemma 3.2.5 $\mathrm{noise}(C, \mathcal{L}_{\mathsf{pre}}) \le \theta \Leftrightarrow$ *there exists* $C' \in \mathcal{L}_{\mathsf{pre}}$ *s.t.* $C \in \mathrm{cov}(C', \theta, \epsilon, |\mathcal{L}_{\mathsf{pre}}|)$.

Proof: The proof is simply from the definition: Let $s = |\mathcal{L}_{\mathsf{pre}}|$. If $\mathrm{noise}(C, \mathcal{L}_{\mathsf{pre}}) \le \theta$, from Equation (3.3), there is a $C' \in \mathcal{L}_{\mathsf{pre}}$ such that $\mathrm{mag}(C, C') \le (\theta\epsilon^2)/(2s^2)$. And thus from

Equation (3.5), $C \in \text{cov}(C', \theta, \epsilon, s)$. The converse direction is similar. □

By Lemma 3.2.5, 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 $\mathcal{L}_{\mathsf{pre}}$ 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 6-11 of Algorithm 1.

Theorem 3.2.6 *Algorithm 1 finds an* $(\ln |\mathcal{L}| + 1)^2$*-approximation to Problem 1 (*BOUND MAX*) in* $O(\min\{3^d,\ 2^d|\mathcal{L}|\}|\mathcal{L}| \log |\mathcal{L}|)$ *time. Using the solution* $\mathcal{L}_{\mathsf{pre}}$ *produced by Algorithm 1,* $\mathcal{K}_{\mathsf{part}}$ *is at least as good as* $\mathcal{K}_{\mathsf{all}}$ *and* $\mathcal{K}_{\mathsf{base}}$*, in terms of the objective in Problem 1.*

Proof: The results are proved one by one as follows.

Approximation Ratio. Suppose the optimal solution is θ^*, i.e., there are s^* cuboids $\mathcal{L}_{\mathsf{pre}}$ s.t. for any $C \in \mathcal{L}$, $\text{noise}(C, \mathcal{L}_{\mathsf{pre}}) \leq \theta^*$. From Lemma 3.2.5, equivalently, there are s^* cuboids $C_1^*, \ldots, C_{s^*}^*$ s.t. $\bigcup_{i=1}^{s^*} \text{cov}(C_i^*, \theta^*, \epsilon, s^*) = \mathcal{L}$. And, from the definition (3.5),

$$\forall \text{ cuboid } C \text{ and } \alpha : \ \text{cov}(C, \alpha^2\theta^*, \epsilon, \alpha s^*) = \text{cov}(C, \theta^*, \epsilon, s^*).$$

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. $\bigcup_{i=1}^{s} \text{cov}(C_i', \theta, \epsilon, s) = \mathcal{L}$, which implies that, with $\mathcal{L}_{\mathsf{pre}} = \{C_1',\ \ldots,\ C_s'\}$, $\max_{C \in \mathcal{L}} \text{noise}(C, \mathcal{L}_{\mathsf{pre}}) \leq \theta = \alpha^2\theta^*$. So we can conclude that Algorithm 1 finds an $(\ln(|\mathcal{L}|) + 1)^2$-approximation. Since for any cuboid C, $\text{cov}(C, \theta, \epsilon, s) = \text{cov}(C, \theta^*, \epsilon, s^*)$ for the selection of θ and s above, we write both of them as $\text{cov}(C)$. Now we only need to prove: if there exists s^* cuboids $C_1^*, \ldots, C_{s^*}^*$ such that $\bigcup_{i=1}^{s^*} \text{cov}(C_i^*) = \mathcal{L}$, the algorithm FEASIBLE finds $s = \alpha s^*$ cuboids $C_1', \ldots, C_s'$ (in this order) s.t. $\bigcup_{i=1}^{s} \text{cov}(C_i') = \mathcal{L}$.

We apply the analysis of the greedy SET COVER algorithm here to complete the proof. 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 $l_i \leq (1 - 1/s^*)l_{i-1}$, 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 $l_i \leq (1 - 1/s^*)^i |\mathcal{L}|$. When $i \geq (\ln |\mathcal{L}| + 1)s^*$, $l_i < 1$. Therefore, 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 and returned by Algorithm 1 is at least as good, in terms of the objective in Problem 1.

Time Complexity. If for some θ, the algorithm FEASIBLE returns a feasible solution for some s, then for any $\theta' \geq \theta$, 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 $\text{noise}(\mathcal{L}_{\mathsf{pre}}) = \text{noise}(\mathcal{L}_{\mathsf{pre}}')$ or $|\text{noise}(\mathcal{L}_{\mathsf{pre}}) - \text{noise}(\mathcal{L}_{\mathsf{pre}}')| \geq 1/\epsilon^2$, for $\text{mag}(C, C')$ is always an integer for any two cuboids C and C'. So we can stop when $|\theta_L - \theta_R| \leq 1/\epsilon^2$. So, 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, using a linked list of cuboids ordered by their coverage, a standard implementation of FEASIBLE needs $O(\sum_{\text{all cuboids } C} |\text{cov}(C)|)$ time.

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

Example 3.2.3 *Suppose we want to publish all cuboids in Figure 3.1(a) ($\mathcal{L} = \mathcal{L}_{\mathsf{all}}$) . When* Feasible$(\cdot)$ *is called with $\theta = 80/\epsilon^2$ and $s = 2$, from* (3.5)*, C' covers C iff $C \preceq C'$ and* $\mathrm{mag}(C, C') \leq 10$*.* $\mathrm{mag}(C, C')$ *can be computed from Figure 3.1(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 $\mathcal{L}_{\mathsf{pre}} = \{C_{111}, C_{101}\}$.*

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

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

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 3.2.5, Publish Most is equivalent to finding a set of s cuboids $\mathcal{L}_{\mathsf{pre}}$ 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 $\mathcal{L}_{\mathsf{pre}}$. If some cuboids in $\mathcal{L}$ cannot be computed from $\mathcal{L}_{\mathsf{pre}}$ but we desire to publish them, we can simply add one more cuboid, the base cuboid, into $\mathcal{L}_{\mathsf{pre}}$.

```
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 $\mathcal{L}_{\mathsf{pre}}$;
GreedyCover($\mathcal{L}, \theta_0, s$)
3: Compute coverage $\text{cov}(C)$ for each cuboid based on $\theta_0$ and $s$;
4: $\mathcal{R} \leftarrow \emptyset$,  $\mathcal{COV} \leftarrow \emptyset$;
5: repeat the following steps $s$ times
6:     Select a cuboid $C'$ such that $|\text{cov}(C') - \mathcal{COV}|$ is maximized
                                   (or $\sum_{C \in \text{cov}(C') - \mathcal{COV}} w(C)$ is maximized)
7:     $\mathcal{R} \leftarrow \mathcal{R} \cup \{C'\}$,  $\mathcal{COV} \leftarrow \mathcal{COV} \cup \text{cov}(C')$;
8: return $\mathcal{R}$.
```

Algorithm 2: Algorithm for Publish Most problem

Theorem 3.2.7 *Algorithm 2 finds a* $(1-1/e)$*-approximation to Problem 2 (*Publish Most*) in* $O(\min\{3^d,\ 2^d|\mathcal{L}|\}d|\mathcal{L}|)$ *time. Moreover, using the solution* $\mathcal{L}_{\mathsf{pre}}$ *produced by Algorithm 2,* $\mathcal{K}_{\mathsf{part}}$ *is as least as good as* $\mathcal{K}_{\mathsf{all}}$ *and* $\mathcal{K}_{\mathsf{base}}$*, in terms of the objective in Problem 2.*

Proof: The proof is very similar to the one for Theorem 3.2.6:

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 $|\bigcup_{i=1}^{s^*} \text{cov}(C_i^*, \theta_0, \epsilon, s^*)| = \text{OPT}$ is maximized. Of course, $s^* \leq |\mathcal{L}|$. When $s = s^*$ in line 2 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

$$|\bigcup_{i=1}^{s^*} \text{cov}(C_i', \theta_0, \epsilon, s^*)| \geq (1 - 1/e)\text{OPT}.$$

The proof for this is similar to the one in the proof of Theorem 3.2.6. Using the same notations (l_i is the number of cuboids in $\mathcal{L}$ covered by $C_1^*, \ldots, C_{s^*}^*$ but not covered by $C_1', \ldots, C_i'$), we have $l_{s^*} \leq (1 - 1/s^*)^{s^*}\text{OPT} \leq \text{OPT}/e$. So the number of cuboids in $\mathcal{L}$

covered by $\{C'_1, \ldots, C'_{s^*}\}$ is at least $(1-1/e)\text{OPT}$. The weight version (maximize the weight of cuboids covered by $\mathcal{L}_{\mathsf{pre}}$) is very similar to the above. So we omit it here.

Comparison to $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$. Similarly, 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, in terms of the objective in Problem 2.

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 $O(\min\{3^d,\ 2^d|\mathcal{L}|\}d|\mathcal{L}|)$. □

In practice, we can also use Algorithm 2 to publish all of $\mathcal{L}$ when that is possible. As shown in the experiments, instead of bounding the max noise as Algorithm 1 does, Algorithm 2 tends to bound the average noise. There is no formal guarantee for this, but the situation in experiments is: when the threshold in Problem 2 is set to be lower than the solution found in Problem 1 (max variance when publishing all cuboids in $\mathcal{L}$), a large portion of cuboids can still have variance lower than this threshold, but only a few cuboids exceed this threshold; and thus, the average of noise variance is lower.

Example 3.2.4 *Consider the effect of Algorithm 2 on Example 3.2.3, with $\theta_0 = 40/\epsilon^2$ and all cuboids to be published with equal weights. In the iteration of $s = 2$, when we call the subroutine* 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 $\mathcal{L}_{\mathsf{pre}}$.*

3.3 Optimizing Noise by Distribution Tuning

A possible generalization of our cuboid selection framework $\mathcal{K}_{\mathsf{part}}$ is: for different selected cuboids in $\mathcal{L}_{\mathsf{pre}}$, we add different amounts of noise to further optimize the goals in Problems 1 and 2. Suppose $\mathcal{L}_{\mathsf{pre}} = \{C_1, C_2, \ldots, C_s\}$ and noise $\mathrm{Lap}(\lambda_i/\epsilon)$ is injected to each cell in C_i, from the composition property of differential [59], we claim that, if $\sum_{i=1}^{s} 1/\lambda_i = 1$, the publishing of $\mathcal{L}_{\mathsf{pre}}$ is ϵ-differentially private. We will introduce this generalized framework for optimizing noise in Section 3.3.1. Finding the optimal $\mathcal{L}_{\mathsf{pre}}$ and parameters $\langle \lambda_i \rangle$ is hard, but we will present efficient algorithms to find approximate solutions in Section 3.3.2.

Intuitively, ϵ can be interpreted as the *privacy budget*, which can be spent on different cuboids to achieve the objective of, e.g., minimizing the max noise in the released cuboids. In $\mathcal{K}_{\mathsf{part}}$, ϵ is spent uniformly on selected cuboids; and in $\mathcal{K}_{\mathsf{gen}}$, we relax this constraint and allow to spend different amounts of privacy budget on different cuboids.

Example 3.3.1 *(Continuing Example 3.2.1) In this example, we want to demonstrate that spending different amounts of privacy budget on different cuboids indeed yields better solutions. Consider again the fact table T in Figure 1.1(a), with three dimensions* Sex*,* Age*, and* Salary *which have cardinality 2, 7, and 5, respectively, and its cuboids in Figure 3.1(a).*

Following the same notations as Example 3.2.1, we compute C_{111} and C_{100} from T. We add Laplace noise $\mathrm{Lap}(1.375/\epsilon)$ to C_{111}, and $\mathrm{Lap}(3.666/\epsilon)$ to C_{100}. As $1/1.375 + 1/3.666 = 1$, ϵ-differential privacy is preserved (we will formalize it later). We compute other cuboids from these two noisy cuboids: for each other cuboid, we select the best of C_{111} and C_{100} to compute it, which does not violate differential privacy as we do not touch the fact table. When we compute C_{000}, we select to compute it from C_{100} and the noise variance is $2 \times (2 \times 3.666^2/\epsilon^2) = 53.758/\epsilon^2$ (aggregating 2 cells of C_{100}). It is the max noise variance among all cuboids, and is better than the approaches in Example 3.2.1 ($\mathcal{K}_{\mathsf{all}}$, $\mathcal{K}_{\mathsf{base}}$, and $\mathcal{K}_{\mathsf{part}}$).

3.3.1 A Generalized Framework: Distribution Selection

To publish a set $\mathcal{L} \subseteq \mathcal{L}_{\mathsf{all}}$ of cuboids, $\mathcal{K}_{\mathsf{gen}}$ chooses to compute cuboids $\mathcal{L}_{\mathsf{pre}}$ directly from the fact table T, and then adds noise $\mathrm{Lap}(\lambda_C/\epsilon)$ to cells in $C \in \mathcal{L}_{\mathsf{pre}}$. $\mathcal{L}_{\mathsf{post}} = \mathcal{L} - \mathcal{L}_{\mathsf{pre}}$ includes all the other cuboids in $\mathcal{L}$. Again, we do *not* require $\mathcal{L}_{\mathsf{pre}} \subseteq \mathcal{L}$. $\mathcal{K}_{\mathsf{gen}}$ works as follows.

1. (*Noise Sources*) For each cell a in cuboid $C \in \mathcal{L}_{\mathsf{pre}}$, $\mathcal{K}_{\mathsf{gen}}$ computes $\mathsf{c}(a)$ from T but releases $\tilde{\mathsf{c}}(a) = \mathsf{c}(a) + \mathrm{Lap}(\lambda_C/\epsilon)$. Both $\mathcal{L}_{\mathsf{pre}}$ and noise parameters $\{\lambda_C \mid C \in \mathcal{L}_{\mathsf{pre}}\}$ are selected by our algorithms in Section 3.3.2 s.t. $\mathcal{L}$ can be computed from $\mathcal{L}_{\mathsf{pre}}$.

2. (*Aggregation*) For each cuboid $C \in \mathcal{L}_{\mathsf{post}}$, $\mathcal{K}_{\mathsf{gen}}$ 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 $\mathcal{K}_{\mathsf{gen}}$ is an unbiased estimator of $\mathsf{c}(a)$ for every cell a. We use σ_a^2 to denote the noise variance of a cell a. For each cell a in a cuboid $C' \in \mathcal{L}_{\mathsf{pre}}$,

$$\sigma_a^2 = \mathrm{Var}\left[\tilde{\mathsf{c}}(a)\right] = 2\lambda_{C'}^2/\epsilon^2. \tag{3.6}$$

Suppose cell a in $C \in \mathcal{L}_{\mathsf{post}}$ is computed from $C' \in \mathcal{L}_{\mathsf{pre}}$ by aggregating on dimensions $[C'] - [C] = \{A_{k_1}, \ldots, A_{k_q}\}$, recall the *variance magnification* is defined as

$$\mathrm{mag}(C, C') = \prod_{1 \le i \le q} |A_{k_i}|. \tag{3.7}$$

So the noise variance, the mean squared error, of $\tilde{\mathsf{c}}(a)$ is

$$\mathrm{Var}\left[\tilde{\mathsf{c}}(a)\right] = \mathrm{mag}(C, C') \cdot 2\lambda_{C'}^2/\epsilon^2. \tag{3.8}$$

Similar to $\mathcal{K}_{\mathsf{part}}$, we should compute the cells in $C \in \mathcal{L}_{\mathsf{post}}$ from the (nearest) cuboid $C^* \in \mathcal{L}_{\mathsf{pre}}$ for which $\mathrm{mag}(C, C^*) \cdot \lambda_{C^*}^2$ is minimal (i.e. $C^* = \mathrm{argmin}_{C' \in \mathcal{L}_{\mathsf{pre}}} \mathrm{mag}(C, C') \cdot \lambda_{C'}^2$).

$$\mathrm{noise}(C, \mathcal{L}_{\mathsf{pre}}) = \min_{C' \in \mathcal{L}_{\mathsf{pre}}} \mathrm{mag}(C, C') \cdot 2\lambda_{C'}^2/\epsilon^2 \tag{3.9}$$

is the smallest possible noise variance when computing C from a single cuboid in the selected cuboid set $\mathcal{L}_{\mathsf{pre}}$. In a special case, if $C' \in \mathcal{L}_{\mathsf{pre}}$, $\text{noise}(C', \mathcal{L}_{\mathsf{pre}}) = 2\lambda_{C'}^2/\epsilon^2$.

Theorem 3.3.1 *$\mathcal{K}_{\mathsf{gen}}$ is ϵ-differentially private if $\sum_{C \in \mathcal{L}_{\mathsf{pre}}} 1/\lambda_C = 1$. For any released cell a in $\mathcal{L}$, $\tilde{\mathsf{c}}(a)$ is an unbiased estimator of $\mathsf{c}(a)$, i.e., $\mathrm{E}[\tilde{\mathsf{c}}(a)] = \mathsf{c}(a)$.*

Proof: The first part can be proved using the composition property of differential privacy (Theorem 2.1.3). And the second part is similar to Theorem 3.2.3. □

$\mathcal{K}_{\mathsf{gen}}$ can be also implemented similarly as $\mathcal{K}_{\mathsf{part}}$. We do not repeat the discussion here.

Distribution selection problems: optimizing noise source $\mathcal{L}_{\mathsf{pre}}$ and $\{\lambda_C\}$

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

Problem 3 (General Bound Max) *Choose a set of cuboids $\mathcal{L}_{\mathsf{pre}}$ and corresponding noise parameters $\{\lambda_C \mid C \in \mathcal{L}_{\mathsf{pre}}\}$ s.t. $\sum_{C \in \mathcal{L}_{\mathsf{pre}}} 1/\lambda_C = 1$, 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 4 (General Publish Most) *Given θ_0 and cuboid weighting $w(\cdot)$, choose $\mathcal{L}_{\mathsf{pre}}$ and $\{\lambda_C\}$ s.t. $\sum_{C \in \mathcal{L}_{\mathsf{pre}}} 1/\lambda_C = 1$ and $\sum_{C \in \mathcal{L}:\ \text{noise}(C, \mathcal{L}_{\mathsf{pre}}) \le \theta_0} w(C)$ is maximized. In other words, maximize the number (weight) of the cuboids in $\mathcal{L}$ with noise variance no more than θ_0.*

Theorem 3.3.2 *The two cuboid selection problems* General Bound Max *and* General Publish Most *are both NP-hard, if we treat $|\mathcal{L}|$ as the input size.*

Proof: The proof of NP-hardness is similar to the one for Problems 1 and 2. □

We also have the following assertion about the optimal solution to Problems 3 and 4.

Theorem 3.3.3 *The optimal solutions to problems* General Bound Max *and* General Publish Most *are at least as good as the optimal solutions to* Bound Max *and* Publish Most *in terms of their objective functions, respectively.*

Proof: The conclusion follows simply because any feasible solution to problems Bound Max and Publish Most are also feasible solutions to problems General Bound Max and General Publish Most, respectively. □

3.3.2 Selecting the Best Parameters of Noise

We now present algorithms for Problems 3-4 to choose $\mathcal{L}_{\mathsf{pre}}$ and $\{\lambda_C\}$. They have the same performance guarantee as the algorithms for Problems 1-2, but are even more efficient.

Bounding the Maximum Variance

We now present a $(\ln(|\mathcal{L}|) + 1)^2$-approximation algorithm for the General Bound Max problem, with running time polynomial in $|\mathcal{L}|$ and 2^d. Let's guess the optimal solution value θ and consider the feasibility problem Feasibility$(\mathcal{L}, \theta)$: for a fixed θ, is there a set of cuboids $\mathcal{L}_{\mathsf{pre}}$ and Laplace noise parameters $\{\lambda_C \mid C \in \mathcal{L}_{\mathsf{pre}}\}$ s.t. for all $C \in \mathcal{L}$, noise$(C, \mathcal{L}_{\mathsf{pre}}) \leq \theta$?

For each cuboid C', let's order all the $\Delta(C')$ cuboids $\{C_1, C_2, \ldots, C_{\Delta(C')}\}$ that can be computed from C' by the variance magnification ratio mag$(\cdot, C')$:

$$\mathrm{mag}(C_1, C') \leq \mathrm{mag}(C_2, C') \leq \ldots \leq \mathrm{mag}(C_{\Delta(C')}, C').$$

C is said to be *covered* by C' if C can be computed from C' with noise variance no more than θ. Define $\Delta(C')$ *coverage sets* $\mathrm{cov}(C', 1), \mathrm{cov}(C', 2), \ldots, \mathrm{cov}(C', \Delta(C'))$, where

$$\mathrm{cov}(C', i) = \{C_1, C_2, \ldots, C_i\}. \tag{3.10}$$

Also define the thresholds of noise parameter associated with each set:

$$\bar{\lambda}(C', i) = \sqrt{\frac{\theta\epsilon^2}{2}} \cdot \frac{1}{\sqrt{\mathrm{mag}(C_i, C')}}. \tag{3.11}$$

Lemma 3.3.4 *If Laplace noise* $\mathrm{Lap}(\lambda/\epsilon)$ *with* $\lambda = \bar{\lambda}(C', i)$ *defined above is injected into each cell of cuboids* C'*, then every cuboid in* $\mathrm{cov}(C', i)$ *can be computed from* C' *with noise variance no more than* θ*. The statement is true for* $i = 1, 2, \dots \Delta(C')$*.*

Proof: If $\mathrm{Lap}(\bar{\lambda}(C', i)/\epsilon)$ is injected into each cell of cuboids C', then $a \in C_i$ can be computed from C' with

$$\mathrm{Var}\left[\tilde{\mathsf{c}}(a)\right] = \mathrm{mag}(C_i, C') \cdot 2\bar{\lambda}(C', i)^2/\epsilon^2 = \theta.$$

Similarly, any of cuboids $C_1, C_2, \dots, C_i$ can be computed with noise variance $\leq \theta$. □

Now we reduce General Bound Max (Problem 3) to the Weighted Set Cover problem in order to develop an efficient approximation algorithm for our problem. The reduction needs to preserve the performance ratio. For any instance of Problem 3, let's construct the following instance of the Weighted Set Cover problem.

Problem 5 (Instance of Weighted Set Cover constructed from Problem 3) *For any instance of Problem 3, let's define a collection of sets:*

$$\mathcal{S} = \{\mathrm{cov}(C', i) \mid C' \in \mathcal{L}_{\mathsf{all}},\ 1 \leq i \leq \Delta(C')\}, \tag{3.12}$$

and each set $\mathrm{cov}(C', i)$ *is associated with weight:*

$$\mathrm{weight}(C', i) = \sqrt{\mathrm{mag}(C_i, C')}. \tag{3.13}$$

The goal is to find a sub-collection $\mathcal{S}_0 \subseteq \mathcal{S}$ *covering cuboids in* $\mathcal{L}$, $\bigcup_{\mathrm{cov}(C',i)\in\mathcal{S}^*} \mathrm{cov}(C', i) = \mathcal{L}$, *such that the total weight of sets in* $\mathcal{S}_0$, $\sum_{\mathrm{cov}(C',i)\in\mathcal{S}_0} \mathrm{weight}(C', i)$, *is minimized.*

For reduction, let's construct a one-to-one mapping between feasible solutions to an instance of Problem 3 and feasible solutions to its corresponding instance of Problem 5.

Lemma 3.3.5 *For fixed ϵ, there exists a feasible solution to Problem 5 with weight w if and only if there exists a feasible solution to Problem 3 with max noise variance $\theta = 2w^2/\epsilon^2$.*

Proof: For Problem 3, we only consider such feasible solutions: if $C' \in \mathcal{L}_{\mathsf{pre}}$, we must have $\lambda_{C'} = \bar{\lambda}(C', i)$ that is defined in (3.11) for some i. Because if $\bar{\lambda}(C', i-1) \geq \lambda_{C'} \geq \bar{\lambda}(C', i)$, we can simply set $\lambda_{C'} = \bar{\lambda}(C', i-1)$, so that less privacy budget is used but C' can cover the same set of cuboids (i.e. the same set of cuboids can be computed from C with noise variance no more than θ). For Problem 5, we only consider such feasible solutions $\mathcal{S}_0$: if $\mathrm{cov}(C', i) \in \mathcal{S}_0$, there does not exist $\mathrm{cov}(C', j) \in \mathcal{S}_0$ s.t. $i \neq j$. Because we must have $\mathrm{cov}(C', i) \subseteq \mathrm{cov}(C', j)$ or $\mathrm{cov}(C', j) \subseteq \mathrm{cov}(C', i)$, so one of them is redundant.

Now we construct the one-to-one mapping as follows.

Consider any feasible solution $\mathcal{S}_0$ to Problem 5 with total weight w. Let's construct a feasible solution to Problem 3: for each $\mathrm{cov}(C', i) \in \mathcal{S}_0$, include C' into $\mathcal{L}_{\mathsf{pre}}$, and let $\lambda_{C'} = w/\mathrm{weight}(C', i) = w/\sqrt{\mathrm{mag}(C_i, C')}$. It is easy to verify that $\sum_{C' \in \mathcal{L}_{\mathsf{pre}}} 1/\lambda_{C'} = 1$, so such $\mathcal{L}_{\mathsf{pre}}$ and $\{\lambda_{C'}\}$ are a feasible solution to Problem 3. Also, if we let $\theta = 2w^2/\epsilon^2$, then $\lambda_{C'} = \bar{\lambda}(C', i)$, and thus from Lemma 3.3.4, cuboids in $\mathrm{cov}(C', i)$ can be computed from C' with noise variance no more than θ. Since $\mathcal{S}_0$ is a set cover of $\mathcal{L}$, any cuboid in $\mathcal{L}$ can be computed from some cuboid in $\mathcal{L}_{\mathsf{pre}}$ with noise variance no more then θ.

For another direction, consider any feasible solution $\mathcal{L}_{\mathsf{pre}}$ and $\{\lambda_{C'} = \bar{\lambda}(C', i)\}$ to Problem 3 with max noise variance θ. We can construct a feasible solution $\mathcal{S}_0$ to Problem 5 by including $\mathrm{cov}(C', i)$ (with weight $\sqrt{\mathrm{mag}(C_i, C')}$) into $\mathcal{S}_0$ for each $C' \in \mathcal{L}_{\mathsf{pre}}$. It is not hard to verify that $\mathcal{S}_0$ is a set cover for $\mathcal{L}$, and the total weight is $w = \sqrt{\theta\epsilon^2/2}$.

Is is not hard to see the above two mappings are identical, so they together form a one-to-one mapping, and the conclusion of this lemma follows. □

Corollary 3.3.6 *Letting* $\mathrm{OPT}_{\mathrm{GBM}}$ *be the optimal solution to Problem 3 and* $\mathrm{OPT}_{\mathrm{WSC}}$ *be the optimal solution to Problem 5,* $\mathrm{OPT}_{\mathrm{GBM}} = 2\mathrm{OPT}^2_{\mathrm{WSC}}/\epsilon^2$. *And, for any α-approximation* *for Problem 5, we can construct an α^2-approximation for Problem 3, and vice versa.*

Proof: The conclusion is directly from Lemma 3.3.5. And the construction for the second part is identical to the one-to-one mapping presented in the proof of Lemma 3.3.5. □

We are ready to introduce our efficient approximation algorithm for Problem 3. The basic idea is to solve the corresponding instance of Weighted Set Cover (Problem 5), and transform the solution into a solution to Problem 3, as the one-to-one mapping in the proof of Lemma 3.3.5. Its performance ratio follows from Corollary 3.3.6. The algorithm is described in Algorithm 3. Lines 3-6 solves the Weighted Set Cover instance using the greedy algorithm, and line 7 transforms it into a solution to Problem 3.

```
1: Compute coverage sets $\mathrm{cov}(C,i)$'s for each cuboid $C$ and $1 \le i \le \Delta(C)$;
2: Each coverage set $\mathrm{cov}(C,i)$ is associated with $\mathrm{weight}(C,i) = \sqrt{\mathrm{mag}(C_i, C)}$;
3: $\mathcal{L}_{\mathsf{pre}} \leftarrow \emptyset$, $\mathcal{COV} \leftarrow \emptyset$, $w \leftarrow 0$;
4: repeat until $\mathcal{COV} = \mathcal{L}$
5:     Select a coverage set $\mathrm{cov}(C',i)$ with $C' \notin \mathcal{L}_{\mathsf{pre}}$ maximizing
           $$\frac{|\mathrm{cov}(C',i) - \mathcal{COV}|}{\mathrm{weight}(C',i)} = \frac{|\mathrm{cov}(C',i) - \mathcal{COV}|}{\sqrt{\mathrm{mag}(C_i, C')}};$$
6:     $\mathcal{L}_{\mathsf{pre}} \leftarrow \mathcal{L}_{\mathsf{pre}} \cup \{C'\}$, $\mathcal{COV} \leftarrow \mathcal{COV} \cup \mathrm{cov}(C',i)$, $w \leftarrow w + \mathrm{weight}(C',i)$;
7: for each $\mathrm{cov}(C',i) \in \mathcal{L}_{\mathsf{pre}}$ do $\lambda_{C'} = w/\mathrm{weight}(C',i)$;
8: return $\mathcal{L}_{\mathsf{pre}}$ and $\{\lambda_C\}$.
```

Algorithm 3: Algorithm for General Bound Max problem

Theorem 3.3.7 *Algorithm 3 finds an* $(\ln|\mathcal{L}|+1)^2$*-approximation to Problem 3 in time* $O(2^d|\mathcal{L}|^2)$*. Using* $\mathcal{L}_{\mathsf{pre}}$ *and* $\{\lambda_C\}$ *produced by Algorithm 3,* $\mathcal{K}_{\mathsf{gen}}$ *is at least as good as* $\mathcal{K}_{\mathsf{part}}$ *using* $\mathcal{L}_{\mathsf{pre}}$ *produced by Algorithm 1, in terms of the max noise variance.*

Proof: Lines 3-6 use the $(\ln|\mathcal{L}|+1)$-approximation algorithm [77] to solve the Weighted Set Cover instance. Line 7 transforms the solution into a solution $\mathcal{L}_{\mathsf{pre}}$ and $\{\lambda_C\}$ to Problem 3. From Corollary 3.3.6, it is an $(\ln|\mathcal{L}|+1)^2$-approximation for Problem 3.

In lines 1-2, we need $O(2^d|\mathcal{L}|\log|\mathcal{L}|)$ time to construct coverage sets $\text{cov}(C,i)$'s and $\text{weight}(C,i)$'s (for each cuboid, we need $O(|\mathcal{L}|\log|\mathcal{L}|)$ time to sort $C_1, C_2, \ldots, C_{\Delta(C)}$ according to $\text{mag}(C_i, C)$). Then a standard implementation of lines 3-6 (the greedy algorithm) needs $O(\sum_{C,i}|\text{cov}(C,i)|)$ time. A simple upper bound of $\sum_{C,i}|\text{cov}(C,i)|$ is $2^d|\mathcal{L}|^2$, because there are at most $2^d|\mathcal{L}|$ coverage sets $\text{cov}(C,i)$'s, each of which has size at most $|\mathcal{L}|$. So the total running time of Algorithm 3 is bounded by $O(2^d|\mathcal{L}|^2)$ (not a tight bound).

The search space of Algorithm 1 is actually a subspace of the search space of Algorithm 3. So it is not surprising that the solution produced by Algorithm 3 is better. □

Publishing as Many as Possible

We now revise Algorithm 3 to solve Problem 4 with provable performance guarantee.

```
1: Compute coverage sets $\text{cov}(C,i)$'s for each cuboid $C$ and $1 \le i \le \Delta(C)$;
2: Each coverage set $\text{cov}(C,i)$ is associated with
       $$\text{weight}(C,i) = \frac{1}{\lambda(C,i)} = \sqrt{\text{mag}(C_i, C')} \cdot \sqrt{\frac{2}{\theta_0 \epsilon^2}};$$
3: $\mathcal{L}_{\mathsf{pre}} \leftarrow \emptyset$, $\mathcal{COV} \leftarrow \emptyset$, $w \leftarrow 0$;
4: repeat until $w > 1$
5:     Select a coverage set $\text{cov}(C',i)$ with $C' \notin \mathcal{L}_{\mathsf{pre}}$ maximizing
           $$\frac{|\text{cov}(C',i) - \mathcal{COV}|}{\text{weight}(C',i)} = \frac{|\text{cov}(C',i) - \mathcal{COV}|}{\sqrt{\text{mag}(C_i, C')}} \cdot \sqrt{\frac{\theta_0 \epsilon^2}{2}};$$
6:     $\mathcal{L}_{\mathsf{pre}} \leftarrow \mathcal{L}_{\mathsf{pre}} \cup \{C'\}$, $\mathcal{COV} \leftarrow \mathcal{COV} \cup \text{cov}(C',i)$, $w \leftarrow w + \text{weight}(C',i)$;
7: for each $\text{cov}(C',i) \in \mathcal{L}_{\mathsf{pre}}$ do $\lambda_{C'} = 1/\text{weight}(C',i)$;
8: return the better one of $\mathcal{L}_{\mathsf{pre}} - \{C_0\}$ and $\{C_0\}$ as $\mathcal{L}_{\mathsf{pre}}$, where $C_0$ is the last one added
   into $\mathcal{L}_{\mathsf{pre}}$ in lines 4-6, with the Laplace noise parameters as $\{\lambda_C\}$.
```

Algorithm 4: Algorithm for GENERAL PUBLISH MOST problem

Similar to Algorithm 3, we consider an BUDGETED MAXIMUM COVERAGE instance with $\mathcal{S} = \{\text{cov}(C',i) \mid C' \in \mathcal{L}_{\mathsf{all}},\ 1 \le i \le \Delta(C')\}$ as the collection of sets. As we have the knowledge about θ_0, the upper bound of noise variance, in Problem 4, each set $\mathrm{cov}(C', i)$ is associated with $\mathrm{weight}(C, i) = 1/\bar{\lambda}(C, i)$. Lines 3-6 are almost identical to the greedy algorithm in Algorithm 3, expect the termination condition. Line 8 excludes C_0 from $\mathcal{L}_{\mathsf{pre}}$ to ensure that $\sum_{C \in \mathcal{L}_{\mathsf{pre}}} \lambda_C \leq 1$. It is not hard to prove the following claim.

Theorem 3.3.8 *Algorithm 4 finds a $\frac{1}{4}(1-1/e)$-approximation to Problem 4 in time $O(2^d |\mathcal{L}|^2)$. Using $\mathcal{L}_{\mathsf{pre}}$ and $\{\lambda_C\}$ produced by Algorithm 4, $\mathcal{K}_{\mathsf{gen}}$ is at least as good as $\mathcal{K}_{\mathsf{part}}$ using $\mathcal{L}_{\mathsf{pre}}$ produced by Algorithm 2, w.r.t. the number of precise cuboids (noise variance $\leq \theta_0$).*

Proof: The equivalence of Problem 4 and the BUDGETED MAXIMUM COVERAGE instance is from Lemma 3.3.4. The approximation ratio is from a similar analysis [18, 46] for the approximation ratio of a greedy algorithm for maximizing submodular function under a budget constraint. The rest part is identical to the proof for Theorem 3.3.7. □

Algorithm 4 can be easily generalized for the weighted version of Problem 4.

3.4 Extension of Optimization Framework

We introduce extensions of our approaches to handle relative errors and data cube measures other than the count measure. We also discuss possible ways to handle larger data cubes.

3.4.1 Minimizing Relative Error

The amount of noise $\mathcal{K}_{\mathsf{part}}$ and $\mathcal{K}_{\mathsf{gen}}$ add to differentially private 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 $\{\lambda_C \mid C \in \mathcal{L}_{\mathsf{pre}}\}$ only depends 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 differentially private frameworks for different publishing tasks

[5, 40, 51, 80, 86, 85]. 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). The advantage is, for a particular cell, it has less relative error if it aggregates more rows. To bound the relative error, we introduce the following two extensions of our approaches.

Ratio-Minimization Approach

Since we want to minimize the relative error, one possible model could be: we first ask users to specify their expectations on the noise variances in different cuboids to be released, and it is our task to optimize the noise source so that the max ratio of the actual noise variances over the users' expectations is minimized. The noise variance users expect in low-level cuboids or "important" cuboids may be lower than the noise variance they expect in high-level or "unimportant" cuboids. In this model, we give users the freedom to determine which cuboids are more important (and thus should be more precise with lower noise variance) based their need in practice. To this end, we can apply the noise optimization framework $\mathcal{K}_{\mathsf{gen}}$, but we need to revise the objective of choosing $\mathcal{L}_{\mathsf{pre}}$ and parameters $\{\lambda_C \mid C \in \mathcal{L}_{\mathsf{pre}}\}$ as follows.

Problem 6 (General Bound Max Ratio) *Given the user expectations on noise variances* VAR_C *in each cuboid* $C \in \mathcal{L}$*, choose a set of cuboids* $\mathcal{L}_{\mathsf{pre}}$ *and corresponding noise parameters* $\{\lambda_C \mid C \in \mathcal{L}_{\mathsf{pre}}\}$ *s.t.* $\sum_{C \in \mathcal{L}_{\mathsf{pre}}} 1/\lambda_C = 1$*, all cuboids in* $\mathcal{L}$ *can be computed from* $\mathcal{L}_{\mathsf{pre}}$*, and the max noise* $\mathrm{noise}(\mathcal{L}_{\mathsf{pre}}) = \max_{C \in \mathcal{L}} \mathrm{noise}(C, \mathcal{L}_{\mathsf{pre}})/\mathrm{VAR}_C$ *is minimized.*

In this problem, $\mathrm{noise}(C, \mathcal{L}_{\mathsf{pre}})/\mathrm{VAR}_C$ is the ratio between the actually noise variance and users' expectation on it. The goal is to minimize the max ratio. Our Algorithm 3 can be revised to solve this problem: we only need to redefine $\mathrm{mag}'(C_i, C) = \mathrm{mag}(C_i, C)/\mathrm{VAR}_C$, order cuboids based on it, and plug it into $\mathrm{weight}(C, i)$ (3.13) and Algorithm 3.

Estimation-Optimization Approach

Here is another way to generalize our approaches to bound relative error. As for any empty cell (with $\mathsf{c}(a) = 0$), any small error leads to infinite relative error, we define the *relative error* in a cuboid C as an average of relative errors of cell measures:

$$\mathrm{Err}(C) = \frac{\sum_{a \in C} \mathrm{E}\left[(\tilde{\mathsf{c}}(a) - \mathsf{c}(a))^2\right]}{\sum_{a \in C} \mathsf{c}(a)^2} = \frac{\mathrm{Var}\left[\tilde{\mathsf{c}}(a)\right]}{\sum_{a \in C} \mathsf{c}(a)^2 / |C|}.$$

Borrowing ideas from [79] and [66], we divide the privacy budget into two parts $\epsilon = \epsilon_1 + \epsilon_2$. ϵ_1 is used to get an estimation $\widetilde{\mathrm{L2}}$ of $\sum_{a \in C} \mathsf{c}(a)^2$ using $\mathcal{K}_{\mathsf{all}}$. Let

$$\widetilde{\mathrm{Err}}(C) = \frac{\mathrm{Var}\left[\tilde{\mathsf{c}}(a)\right]}{\widetilde{\mathrm{L2}} / |C|}.$$

Then we can modify the goal of Problems 1 and 3 as minimizing $\max_C \widetilde{\mathrm{Err}}(C)$, and the definition of precise cuboids in Problems 2 and 4 as the ones with $\widetilde{\mathrm{Err}}(C) \leq \theta_0$. Algorithms 1-4 can be also revised accordingly to handle the relative-error versions of Problems 1-4.

3.4.2 Extension to Other Measures

Our techniques for the count measure c can be extended to other two basic measures summation sum and average 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 the count measure c, the sensitivity of a publishing function for sum is magnified Λ times, where Λ is the range of possible values for any individual fact table tuple. Thus our techniques for the count measure c can be applied to sum, with the noise variance magnified Λ^2 times. To handle the average measure avg, we can compute two differentially private data cubes (partitioning the privacy budget across them), one with sum measure and one with count measure c, from the same fact table. The avg measure of a cell can be computed from the two noisy data cubes, without violating differentially private.

3.4.3 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 may not be favorable to publish all cuboids while ensuring differential privacy, as the huge amount of noise will make the result meaningless. So we now outline an online version of our approach as follows.

Initially, $\mathcal{L}_{\mathsf{pre}} = \emptyset$ 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 differentially private 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 [55], inject noise into C to obtain a differentially private 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 differentially private 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 differentially private cuboid C. After we run out of privacy budget ϵ, to ensure ϵ-differential privacy, we cannot create new differentially private cuboids in $\mathcal{L}_{\mathsf{pre}}$ any more and may be unable to answer new queries. How to distribute the privacy budget online in such a way that more queries can be answered with less error is an *online decision problem*, that is, decision must be made without knowledge about queries in future, but should be able to handle future queries reasonably well. It has some connection to the ONLINE SET COVER problem, and we think it is interesting future work.

Chapter 4

Enforcing Consistency in Noisy Cubes

The two publishing schemes, $\mathcal{K}_{\mathsf{part}}$ and $\mathcal{K}_{\mathsf{gen}}$, introduced in Chapter 3 publish differentially private data cubes, but may also generate *inconsistency* across different cuboids in one data cube. If we roll up measures across differentially private cuboids, the sums may not match the totals recorded in other cuboids. The reason for such inconsistency is that the Laplace mechanism add noise independently to each cell. According to a Microsoft user acceptance study [47], users are likely to accept these kinds of small inconsistencies if they trust the original data and understand why the inconsistencies are present. If the users do not trust the original data, they may interpret the inconsistencies as evidence of bad data. In both cases, after generating a set of cuboids to publish, one can enforce a requirement for correct roll-up totals across dimensions while preserving differential privacy, and in some cases also reduce the amount of noise, through a multidimensional extension of the techniques of [40].

In this chapter, 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 (Theorem 2.1.3), it is also ϵ-differentially private. The output consistent measure $\hat{\mathsf{c}}(\cdot)$ should be as close to $\tilde{\mathsf{c}}(\cdot)$ and $\mathsf{c}(\cdot)$ as possible.

In Section 4.1, we will first discuss the reason of inconsistency, formally define the *consistency constraints*, and the goal of our consistency-enforcing schemes. Then in Sections 4.2-4.3, we will introduce severals approaches to enforce consistency with statistical performance guarantee. In particular, in Sections 4.3.1-4.3.2, we will introduce approaches to enforce consistency in the data cubes published by $\mathcal{K}_{\mathsf{part}}$ and $\mathcal{K}_{\mathsf{gen}}$, which can even improve the utility of noisy differentially private data cubes (reducing noise variance).

4.1 Consistency and Distance

Suppose $\mathcal{L}_{\mathsf{pre}} = \{C_1, C_2\}$, 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 may yield different results. A simple way to resolve such inconsistency is to revise $\mathcal{K}_{\mathsf{part}}$ by letting C be the weighted (based on variance) average of the two results. But 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 $\mathcal{K}_{\mathsf{all}}$, $\mathcal{K}_{\mathsf{part}}$, and $\mathcal{K}_{\mathsf{gen}}$, as long as $\mathcal{L}_{\mathsf{pre}}$ contains more than one cuboid.

The basic idea of our approaches to enforce consistency across released cuboids is as follows. Consider the noisy measures $\tilde{\mathsf{c}}(\cdot)$ of cells in each cuboid $C \in \mathcal{L}_{\mathsf{pre}}$ released by $\mathcal{K}_{\mathsf{part}}$ or $\mathcal{K}_{\mathsf{gen}}$, 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-differential privacy is automatically preserved.

Consistency Constraints

Before presenting our approaches to enforce consistency, we first formalize the *consistency constraints*. 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 $\text{Base}(a)$ be the set of all base cells, each of which aggregates a disjoint subset of rows that are contained in a. Formally, $a' \in \text{Base}(a)$ if and only if for any dimension A_i, $a[i] \neq *$ implies $a[i] = a'[i]$. So the consistency constraints we want to enforce for measure $\hat{\mathsf{c}}(\cdot)$ are as follows:

$$\sum_{a' \in \mathrm{Base}(a)} \hat{\mathsf{c}}(a') = \hat{\mathsf{c}}(a), \quad \forall \text{ cells } a. \tag{4.1}$$

Distance Measures

We seek the choice of $\hat{\mathsf{c}}(\cdot)$ that satisfies consistency constraints in (4.1) and is as close as possible to $\tilde{\mathsf{c}}(\cdot)$. The intuition is that, as we must compute $\hat{\mathsf{c}}(\cdot)$ without reexamining $\mathsf{c}(\cdot)$ to preserve differential privacy, we can treat $\tilde{\mathsf{c}}(\cdot)$ as an approximate version of $\mathsf{c}(\cdot)$.

We use L^p *distance* $(p \geq 1)$ to measure how close $\hat{\mathsf{c}}(\cdot)$ and $\tilde{\mathsf{c}}(\cdot)$ are:

$$||\hat{\mathsf{c}}(\cdot) - \tilde{\mathsf{c}}(\cdot)||_p = \sum_{a \in \mathcal{E}_{\mathsf{pre}}} \left(|\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)|^p\right)^{1/p}, \tag{4.2}$$

where $\mathcal{E}_{\mathsf{pre}}$ be the set of cells in cuboids belonging to $\mathcal{L}_{\mathsf{pre}}$, and we treat $\hat{\mathsf{c}}(\cdot)$ and $\tilde{\mathsf{c}}(\cdot)$ as vectors in $\mathcal{R}^{\mathcal{E}_{\mathsf{pre}}}$. We will prove that the utility of optimal solutions for L^1, L^2, and L^∞ distances satisfies certain statistical utility guarantees, and a bit surprisingly, the utility of the L^2 optimal solution $\hat{\mathsf{c}}(\cdot)$ is even better than the utility of $\tilde{\mathsf{c}}(\cdot)$.

Finding $\hat{\mathsf{c}}(\cdot)$ to minimize $||\hat{\mathsf{c}}(\cdot) - \tilde{\mathsf{c}}(\cdot)||_p$ subject to consistency constraints (4.1) can be viewed as a *least-norm problem*. From classic results from convex optimization [10], it can be solved efficiently, at least in theory. We will analyze the utility of L^∞ and L^1 optimal solutions in Section 4.2. However, classic algorithms, such as linear programming, do not work for our context in practice because the number of variables involved in (4.1) 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 in Section 4.3.

4.2 Linear Programming Consistency

Minimizing L^∞ 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.

$$\begin{aligned} &\text{minimize } z \\ &\text{s.t.} \quad |\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)| \le z, \quad \forall \text{ cells } a \in \mathcal{E}_{\mathsf{pre}}; \\ &\sum_{a' \in \mathrm{Base}(a)} \hat{\mathsf{c}}(a') = \hat{\mathsf{c}}(a), \quad \forall \text{ cells } a \in \mathcal{E}_{\mathsf{pre}}. \end{aligned} \tag{4.3}$$

The first set of constraints is used to bound the L^∞ distance, and the second one is used to enforce consistency (refer to (4.1)). This linear program is inspired by one approach in Barak *et al.* [5], which considers a similar consistency enforcing scheme but injects noise into all cuboids instead of the carefully-selected cuboid subset $\mathcal{L}_{\mathsf{pre}}$. Solving the above linear program, we can bound the error of $\hat{\mathsf{c}}(\cdot)$ with high probability as follows.

Theorem 4.2.1 (Generalized Theorem 8 in [5]) *For $\hat{\mathsf{c}}(\cdot)$ obtained by solving (4.3), with probability at least $1-\delta$, where $\delta = \frac{|\mathcal{E}_{\mathsf{pre}}|}{e^{\eta/2}}$ and $\mathcal{E}_{\mathsf{pre}}$ is the set of cells in cuboids $\mathcal{L}_{\mathsf{pre}}$,*

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

Proof: 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 $|\mathcal{E}_{\mathsf{pre}}|$ 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 (4.3) and $\hat{\mathsf{c}}(\cdot)$ is the optimal. So

$$\forall a \in \mathcal{E}_{\mathsf{pre}}: \ |\hat{\mathsf{c}}(a) - \mathsf{c}(a)| \le |\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)| + |\mathsf{c}(a) - \tilde{\mathsf{c}}(a)| \le 2|\tilde{\mathsf{c}}(a) - \mathsf{c}(a)| \le \frac{|\mathcal{L}_{\mathsf{pre}}|}{\epsilon} 2 \log \frac{|\mathcal{E}_{\mathsf{pre}}|}{\delta}.$$

The conclusion follows by summing the above inequalities up. □

Minimizing L^1 Distance

A linear program can also be used to minimize the L^1 distance. As $||\hat{\mathsf{c}}(\cdot)-\tilde{\mathsf{c}}(\cdot)||_1 = \sum_a |\hat{\mathsf{c}}(a)-\tilde{\mathsf{c}}(a)|$, we introduce an auxiliary variable z_a for each cell a in $\mathcal{E}_{\mathsf{pre}}$ (recall $\mathcal{E}_{\mathsf{pre}}$ is the set of cells in cuboids belonging to $\mathcal{L}_{\mathsf{pre}}$, i.e., the cells to be released), together with a constraint $-z_a \le \hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a) \le z_a$. Then 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.

$$\begin{aligned} &\text{minimize} \sum_{a \in \mathcal{E}_{\mathsf{pre}}} z_a \qquad (4.4)\\ &\text{s.t.} \quad |\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)| \le z_a, \quad \forall \text{ cell } a \in \mathcal{E}_{\mathsf{pre}};\\ &\qquad \sum_{a' \in \text{Base}(a)} \hat{\mathsf{c}}(a') = \hat{\mathsf{c}}(a), \quad \forall \text{ cell } a \in \mathcal{E}_{\mathsf{pre}}. \end{aligned}$$

The error of the solution $\hat{\mathsf{c}}(\cdot)$ is shown in the following theorem.

Theorem 4.2.2 *For $\hat{\mathsf{c}}(\cdot)$ obtained by solving (4.4), if $\epsilon \le 1$, with probability at least $1-\delta$, where $\delta = (\frac{\eta}{2e^{\eta/2-1}})^{|\mathcal{E}_{\mathsf{pre}}|}$ ($\eta > 2$) and $\mathcal{E}_{\mathsf{pre}}$ is the set of cells in cuboids $\mathcal{L}_{\mathsf{pre}}$,*

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

Proof: Let's start with generalizing the Chernoff inequality for exponential distribution.

Lemma 4.2.3 (Problem 1.10 in [19]) *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 $X \leq \eta \mathrm{E}\left[X\right]$ ($\eta > 1$).*

The above lemma can be proved by considering the moment generating function. We will utilize it to prove Theorem 4.2.2 in the following part:

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 $\epsilon \leq |\mathcal{L}_{\mathsf{pre}}|$. Let $X = \sum_{a \in \mathcal{E}_{\mathsf{pre}}} X_a$. We have $\mathrm{E}\left[X\right] = |\mathcal{E}_{\mathsf{pre}}||\mathcal{L}_{\mathsf{pre}}|/\epsilon$. From Lemma 4.2.3, we have, with probability at least $1-\delta$, where $\delta = (\frac{\eta}{e^{\eta-1}})^{|\mathcal{E}_{\mathsf{pre}}|}$, $\sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\tilde{\mathsf{c}}(a) - \mathsf{c}(a)| \leq \frac{|\mathcal{E}_{\mathsf{pre}}||\mathcal{L}_{\mathsf{pre}}|}{\epsilon}\eta$. So

$$\begin{aligned}
\sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\hat{\mathsf{c}}(a) - \mathsf{c}(a)| &\leq \sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a)| + \sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\tilde{\mathsf{c}}(a) - \mathsf{c}(a)| \\
&\leq \sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\mathsf{c}(a) - \tilde{\mathsf{c}}(a)| + \sum_{a \in \mathcal{E}_{\mathsf{pre}}} |\tilde{\mathsf{c}}(a) - \mathsf{c}(a)| \\
&\leq \frac{|\mathcal{E}_{\mathsf{pre}}||\mathcal{L}_{\mathsf{pre}}|}{\epsilon} 2\eta.
\end{aligned}$$

The second inequality above is because $\mathsf{c}(\cdot)$ is a feasible solution to (4.4) but $\hat{\mathsf{c}}(\cdot)$ is the optimal. Set $\eta = \eta/2$ to complete the proof. □

For a data cube where the cardinality of every dimension is two, Theorem 8 in [5] yields a bound similar to that of our Theorem 4.2.1, by discarding the integrality constraint (i.e., counts are integers). Theorem 4.2.2 mirrors Theorem 4.2.1, but replaces the upper tail $\delta = \frac{|\mathcal{E}_{\mathsf{pre}}|}{e^{\eta}}$ with $(\frac{\eta}{2e^{\eta/2-1}})^{|\mathcal{E}_{\mathsf{pre}}|}$. As $|\mathcal{E}_{\mathsf{pre}}|$ is large, linear program (4.4) and Theorem 4.2.2 give a much better bound on the average error in $\hat{\mathsf{c}}(\cdot)$ than (4.3) and Theorem 4.2.1.

To enforce the integrality constraint for $\hat{\mathsf{c}}(\cdot)$ obtained from linear program (4.3) or (4.4), 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 4.2.1 and 4.2.2.

4.3 Least Squares Consistency

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. A bit surprisingly, this problem has a much more efficient solution with better statistical guarantees than the LP-based methods introduced in Section 4.2.

In $\mathcal{K}_{\mathsf{part}}$, a subset of cuboids $\mathcal{L}_{\mathsf{pre}}$ is selected with equal amount of noise inserted into each of them, and other cuboids are computed from the noisy cuboids $\mathcal{L}_{\mathsf{pre}}$ to preserve differential privacy. To further optimize the overall noise, in $\mathcal{K}_{\mathsf{gen}}$, for the selected subset $\mathcal{L}_{\mathsf{pre}}$ of cuboids, we also determine how to spend the privacy budget among them, and thus each cuboid may be inserted with a different amount of noise. Recall that we use $\mathcal{E}_{\mathsf{pre}}$ to denote all cells in $\mathcal{L}_{\mathsf{pre}}$; and in $\mathcal{K}_{\mathsf{gen}}$, we use σ_a^2 to denote the noise variance of a cell a and σ_C^2 to denote the noise variance of cells in a cuboid C. Section 4.3.1 introduces how to enforce consistency in $\mathcal{K}_{\mathsf{part}}$ by solving the unweighted version of the L^2-distance minimization problem, and Section 4.3.2 introduce how to enforce consistency in $\mathcal{K}_{\mathsf{gen}}$ by solving the weighted version.

4.3.1 Unweighted Version: Enforcing Consistency in $\mathcal{K}_{\mathsf{part}}$

Program (4.5) 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 [10]. 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.

$$\begin{aligned} &\text{minimize} \sum_{a \in \mathcal{E}_{\mathsf{pre}}} (\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a))^2 \qquad (4.5)\\ &\text{s.t.} \sum_{a' \in \text{Base}(a)} \hat{\mathsf{c}}(a') = \hat{\mathsf{c}}(a), \quad \forall \text{ cell } a \in \mathcal{L}_{\mathsf{pre}}. \end{aligned}$$

We define $\text{Ancs}(a')$ to be the set of *ancestor cells* of cell a': $a \in \text{Ancs}(a')$ 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, $a \in \text{Ancs}(a')$ if and only if for any dimension A_i, $a[i] \neq *$ implies $a[i] = a'[i]$. If a' is a base cell, $a \in \text{Ancs}(a') \Leftrightarrow a' \in \text{Base}(a)$.

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 (4.5):

1. Bottom-up stage: We first compute $\text{obs}(a'')$ for each cell a'' in cuboids $\mathcal{L}$ to be released:

$$\text{obs}(a'') = \sum_{a' \in \text{Base}(a'')} \sum_{a \in \mathcal{E}_{\mathsf{pre}} \cap \text{Ancs}(a')} \tilde{\mathsf{c}}(a). \tag{4.6}$$

The quantity $\text{obs}(a'')$ is well-defined if a'' is in a cuboid C'' that can be computed from $\mathcal{L}_{\mathsf{pre}}$. A cuboid $C \in \mathcal{L}_{\mathsf{pre}}$ is *maximal in* $\mathcal{L}_{\mathsf{pre}}$ if there is no $C' \in \mathcal{L}_{\mathsf{pre}}$ s.t. $C \prec C'$. In all maximal cuboids C'' in $\mathcal{L}_{\mathsf{pre}}$, $\text{obs}(a'')$ can be computed as in Formula (4.6). In all the other cuboids that can be computed from $\mathcal{L}_{\mathsf{pre}}$, $\text{obs}(a'')$ can be computed recursively as follows: suppose $a'' \in C''$ and C'' can be computed from $\mathcal{L}_{\mathsf{pre}}$, there must be some cuboid C with dimensionality $\dim(C'') + 1$ s.t. either $C \in \mathcal{L}_{\mathsf{pre}}$ or C can be computed from $\mathcal{L}_{\mathsf{pre}}$; then,

$$\text{obs}(a'') = \sum_{\substack{a:\ a \in C \\ a[C''] = a''[C'']}} \text{obs}(a). \tag{4.7}$$

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

$$\text{est}(a'') = \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \sum_{\substack{b:\ b \in (C \wedge C''), \\ b[C \wedge C''] = a''[C \wedge C'']}} \deg(C \vee C'')\hat{\mathsf{c}}(b), \tag{4.8}$$

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 obs$(\cdot)$'s are computed in the first stage as constants and $\hat{\mathsf{c}}(\cdot)$'s are variables. Solving the equations est(a'') = obs(a''), we can compute $\hat{\mathsf{c}}(\cdot)$'s in a top-down manner (from ancestors to descendants) as follows:

$$\text{ratio}(C'') = \sum_{\substack{C:\ C \in \mathcal{L}_{\mathsf{pre}}, \\ C'' \preceq C}} \deg(C \vee C''), \tag{4.9}$$

$$\text{aux}(a'') = \sum_{\substack{C:\ C \in \mathcal{L}_{\mathsf{pre}}, \\ C'' \npreceq C}} \sum_{\substack{b:\ b \in (C \wedge C''), \\ b[C \wedge C''] = a''[C \wedge C'']}} \deg(C \vee C'')\hat{\mathsf{c}}(b), \tag{4.10}$$

$$\hat{\mathsf{c}}(a'') = \frac{1}{\text{ratio}(C'')} \left(\text{obs}(a'') - \text{aux}(a'')\right). \tag{4.11}$$

We note that this approach essentially generalizes the consistency-enforcing scheme in [40] from a tree-like structure (hierarchy of intervals to answer one-dimensional range queries) to a lattice. The method in [40] cannot be directly applied here.

The above two-stage approach can be also applied to $\mathcal{K}_{\mathsf{all}}$ because $\mathcal{K}_{\mathsf{all}}$ is a special case of $\mathcal{K}_{\mathsf{part}}$ obtained by setting $\mathcal{L}_{\mathsf{pre}} = \mathcal{L}$ and injecting equal amount of noise into each cuboid in $\mathcal{L}_{\mathsf{pre}}$. We will show how to generalize it for $\mathcal{K}_{\mathsf{gen}}$ in Section 4.3.2.

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 mean squared error.

Theorem 4.3.1 *(i) The above algorithm correctly computes a value for* $\hat{\mathsf{c}}(\cdot)$ *that is consistent and solves the* L^2 *minimization problem* (4.5). *(ii) The above algorithm requires* $O(N(d^2 + d|\mathcal{L}_{\mathsf{pre}}|))$ *time to compute* $\hat{\mathsf{c}}(\cdot)$ *for all the* N *cells to be released. (iii) For any cell of cuboids* *in $\mathcal{L}_{\mathsf{pre}}$, $\hat{\mathsf{c}}(a)$ is an unbiased estimator of $\mathsf{c}(a)$, i.e.,* $\mathrm{E}\left[\hat{\mathsf{c}}(a)\right] = \mathsf{c}(a)$. *(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 $\tilde{\mathsf{c}}(\cdot)$.*

Proof: We will prove the correctness, efficiency, and statistical optimality one by one.

(i) Correctness. We rewrite (4.5) 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',

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

which is equivalent to

$$\sum_{a \in \mathcal{E}_{\mathsf{pre}} \cap \mathrm{Ancs}(a')} \hat{\mathsf{c}}(a) = \sum_{a \in \mathcal{E}_{\mathsf{pre}} \cap \mathrm{Ancs}(a')} \tilde{\mathsf{c}}(a). \tag{4.12}$$

For any cell a'' in a cuboid C'', sum up Equation (4.12) 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):

$$
\begin{aligned}
\mathrm{est}(a'') &= \sum_{a' \in \mathrm{Base}(a'')} \; \sum_{a \in \mathcal{E}_{\mathsf{pre}} \cap \mathrm{Ancs}(a')} \hat{\mathsf{c}}(a) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \; \sum_{a' \in \mathrm{Base}(a'')} \; \sum_{a \in C \cap \mathrm{Ancs}(a')} \hat{\mathsf{c}}(a) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \; \sum_{\substack{a':\ a' \in \mathrm{Base}, \\ a'[C'']=a''[C'']}} \; \sum_{\substack{a:\ a \in C, \\ a[C]=a'[C]}} \hat{\mathsf{c}}(a) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \; \sum_{a \in C} \; \sum_{\substack{a':\ a' \in \mathrm{Base},\ a[C]=a'[C] \\ a'[C'']=a''[C'']}} \hat{\mathsf{c}}(a) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \; \sum_{\substack{a:\ a \in C, \\ a[C \wedge C'']=a''[C \wedge C'']}} \; \sum_{\substack{a':\ a' \in \mathrm{Base},\ a[C]=a'[C] \\ a'[C'']=a''[C'']}} \hat{\mathsf{c}}(a) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \; \sum_{\substack{a:\ a \in C, \\ a[C \wedge C'']=a''[C \wedge C'']}} \left(\hat{\mathsf{c}}(a) \prod_{A_i \in \mathcal{A}-[C]-[C'']} |A_i| \right) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \; \sum_{\substack{b:\ b \in (C \wedge C''), \\ b[C \wedge C'']=a''[C \wedge C'']}} \left(\hat{\mathsf{c}}(b) \prod_{A_i \in \mathcal{A}-[C]-[C'']} |A_i| \right) \\
&= \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \; \sum_{\substack{b:\ b \in (C \wedge C''), \\ b[C \wedge C'']=a''[C \wedge C'']}} \deg(C \vee C'') \hat{\mathsf{c}}(b), \qquad (4.13)
\end{aligned}
$$

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

$$
\mathrm{obs}(a'') = \sum_{a' \in \mathrm{Base}(a'')} \; \sum_{a \in \mathcal{E}_{\mathsf{pre}} \cap \mathrm{Ancs}(a')} \tilde{\mathsf{c}}(a) = \sum_{\substack{a:\ a \in C \\ a[C'']=a''[C'']}} \mathrm{obs}(a), \qquad (4.14)
$$

where C could be any cuboid that is a descendant of C'' and can be computed from $\mathcal{L}_{\mathsf{pre}}$. So for each cell a'', we can compute the value of $\mathrm{obs}(a'')$ recursively from $\tilde{\mathsf{c}}(\cdot)$. Starting from the base cuboid, $\mathrm{obs}(\cdot)$ can be computed level-by-level from bottom to top.

From Equation (4.12), we know $\text{est}(a'') = \text{obs}(a'')$. So we can construct a linear system with $\hat{\mathsf{c}}(\cdot)$ as variables. From this linear system, we can obtain $\hat{\mathsf{c}}(\cdot)$ as in (4.6)-(4.11).

(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 $\text{obs}(a')$ as in (4.6) (recall $\text{Base}(a') = \{a'\}$ 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 $\text{obs}(a'')$ as in (4.7), 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 (4.9)-(4.11): assuming $\deg(\cdot)$ and $\text{ratio}(\cdot)$ have been precomputed, $\text{aux}(a'')$ (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 (4.10). 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 $\text{E}[\hat{\mathsf{c}}(a)] = \mathsf{c}(a)$ by induction on the dimensionality of a. Recall $\text{E}[\tilde{\mathsf{c}}(a)] = \mathsf{c}(a)$. For the 0-dim cell a_0, we have $\text{aux}(a_0) = 0$, and thus from (4.6) and (4.11), $\hat{\mathsf{c}}(a_0)$ is nothing but the weighted average of "different ways to obtain it from cuboids in $\mathcal{L}_{\mathsf{pre}}$". We can show $\text{E}[\hat{\mathsf{c}}(a_0)] = \mathsf{c}(a_0)$ using the linearity of expectation. For an i-dim cell, we first take expectation on both sides of (4.11), and then use 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 (4.5)) and unbiased. Independent noise with identical variance is injected. So the promised properties follow from the Gauss-Markov theorem [71]. $\square$

4.3.2 Weighted Version: Enforcing Consistency in $\mathcal{K}_{\mathsf{gen}}$

In $\mathcal{K}_{\mathsf{gen}}$, each cell a in a cuboid $C \in \mathcal{L}_{\mathsf{pre}}$ is injected with different amount of noise with variance $\sigma_a^2 = \text{Var}[\tilde{\mathsf{c}}(a)] = 2\lambda_C^2/\epsilon^2$. If cells are from the same cuboid C, the noise variances are the same, so we denote it as σ_C^2. We will use σ_a^2 and σ_C^2 interchangeably if a is from C. The generalized least squares or Aitken estimator extends the Gauss-Markov theorem [71] for the case when the injected Laplacian noise samples are uncorrelated with each other and with the independent variables but have different variances. So we now revise the L^2-distance minimization problem in (4.5) for generalized least squares, to guarantee the statistical optimality of the solution

$$\text{minimize} \sum_{a \in \mathcal{E}_{\mathsf{pre}}} \frac{(\hat{\mathsf{c}}(a) - \tilde{\mathsf{c}}(a))^2}{\sigma_a^2} \tag{4.15}$$

$$\text{s.t.} \sum_{a' \in \text{Base}(a)} \hat{\mathsf{c}}(a') = \hat{\mathsf{c}}(a), \quad \forall \text{ cell } a \in \mathcal{L}_{\mathsf{pre}}.$$

The only difference between programs (4.15) and (4.5) is that we have a weight $1/\sigma_a^2$ for each term in (4.15)'s objective function. So the algorithm for solving (4.5) can be also revised to solve (4.15) with the same time complexity and the same statistical optimality.

1. Bottom-up stage: We first compute obs(a'') for each cell a'' in cuboids $\mathcal{L}$ to be released:

$$\text{obs}(a'') = \sum_{a' \in \text{Base}(a'')} \sum_{a \in \mathcal{E}_{\mathsf{pre}} \cap \text{Ancs}(a')} \frac{\tilde{\mathsf{c}}(a)}{\sigma_a^2}. \tag{4.16}$$

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

$$\text{est}(a'') = \sum_{C \in \mathcal{L}_{\mathsf{pre}}} \sum_{\substack{b:\ b \in (C \wedge C''),\\ b[C \wedge C''] = a''[C \wedge C'']}} \frac{\deg(C \vee C'')\hat{\mathsf{c}}(b)}{\sigma_C^2}. \tag{4.17}$$

Similarly, suppose obs$(\cdot)$'s are computed in the first stage as constants and $\hat{\mathsf{c}}(\cdot)$'s are variables. Solving the equations est(a'') = obs(a''), we can compute $\hat{\mathsf{c}}(\cdot)$'s in a top-down manner (from ancestors to descendants) as follows:

$$\mathrm{ratio}(C'') = \sum_{\substack{C:\ C\in\mathcal{L}_{\mathsf{pre}},\\ C''\preceq C}} \frac{\deg(C\vee C'')}{\sigma_C^2}, \tag{4.18}$$

$$\mathrm{aux}(a'') = \sum_{\substack{C:\ C\in\mathcal{L}_{\mathsf{pre}},\\ C''\not\preceq C}} \sum_{\substack{b:\ b\in(C\wedge C''),\\ b[C\wedge C'']=a''[C\wedge C'']}} \frac{\deg(C\vee C'')\hat{\mathsf{c}}(b)}{\sigma_C^2}, \tag{4.19}$$

$$\hat{\mathsf{c}}(a'') = \frac{1}{\mathrm{ratio}(C'')}\left(\mathrm{obs}(a'') - \mathrm{aux}(a'')\right). \tag{4.20}$$

Theorem 4.3.2 *(i) The above algorithm correctly computes a value for $\hat{\mathsf{c}}(\cdot)$ that is consistent and solves the L^2 minimization problem (4.15). (ii) The above algorithm requires $O(N(d^2 + d|\mathcal{L}_{\mathsf{pre}}|))$ time to compute $\hat{\mathsf{c}}(\cdot)$ for all the N cells to be released. (iii) For any cell of cuboids in $\mathcal{L}_{\mathsf{pre}}$, $\hat{\mathsf{c}}(a)$ 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 $\tilde{\mathsf{c}}(\cdot)$.*

Proof: The proof to this theorem is almost identical to the one for Theorem 4.3.1. The key idea is to show that finding the optimal solution to (4.15) is equivalent to finding the unique solution for the following linear system (with variables as $\hat{\mathsf{c}}(a)$'s for all cells a's):

$$\sum_{a\in\mathcal{E}_{\mathsf{pre}}\cap\mathrm{Ancs}(a')} \frac{\hat{\mathsf{c}}(a)}{\sigma_a^2} = \sum_{a\in\mathcal{E}_{\mathsf{pre}}\cap\mathrm{Ancs}(a')} \frac{\tilde{\mathsf{c}}(a)}{\sigma_a^2}. \tag{4.21}$$

We omit the details in this proof for its similarity to the proof for Theorem 4.3.1. $\square$

Chapter 5

Background Knowledge on Cuboids

Since the notion of *differential privacy* was introduced in [26, 20], it has become more and more popular recently. As indicated by its name, it protects privacy of any differential user, or in other words, including into or excluding any particular user from the database does not affect the output of a differentially private algorithm to a sensible degree. However, as shown in [43], certain types of background knowledge may break differential privacy. An undesirable consequence is that, with certain background knowledge, an adversary could reduce the noise variable by a factor proportional to the size of the dataset.

In this chapter, we first review the notation of *generic differential privacy* [43] in Section 5.1. This notation of privacy aims to incorporate adversary's background knowledge into the model of differential privacy. In the contexts of data cubes, we are interested in background knowledge in the form of some cuboids that need to be released exactly (because of laws of policies). Adversaries can utilize these exact cuboids to attack individuals' privacy. Intuitively, the more background knowledge adversaries have, the more noise needs to be injected to preserve individuals' privacy. By studying generic differential privacy, we want to answer questions about how much noise is necessary for certain amount/type of adversary's background knowledge. We introduce our new results on ways to preserve generic differential privacy and its properties in Section 5.2. We also propose techniques to optimize noise source in data cubes to be released subject to background knowledge (i.e., with some exact cuboids released) while preserving generic differential privacy in Section 5.3.

5.1 Background: Generic Differential Privacy

Generic differential privacy was proposed by Kifer and Machanavajjhala in [43]. It generalizes the definition of *neighborhood* in the original differential privacy, which take the adversary's background knowledge into consideration. The intuition is that, representing the background knowledge as a set of constraints, the adversary knows that some databases are *infeasible* (which do not satisfy some constraints); and the adversary can then focus on all *feasible* databases instead, which satisfy all constraints. So we define T and T' to be neighbors iff *they are both feasible, and there is no other feasible table on the shortest path between T and T'*. We formally define *neighbors* according to the following definitions.

Definition 5.1.1 (Background Knowledge) *For a fact table T, background knowledge $\mathcal{Q}$ is a set of cuboids $\{Q_1, Q_2, \ldots, Q_k\} \subseteq \mathcal{L}_{\mathsf{all}}$ that are released exactly.*

Definition 5.1.2 (Move [43]) *Given a fact table T, a move m is a process that adds or deletes a tuple from T, resulting in a fact table $T' = \mathrm{m}(T)$.*

Definition 5.1.3 (Feasible Tables $\mathcal{T}_{\mathcal{Q}}$) *Given background knowledge $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_k\}$, a fact table is feasible if the count measures in its corresponding cuboids are identical to the ones in $Q_1, Q_2, \ldots, Q_k$. Let $\mathcal{T}_{\mathcal{Q}}$ be the set of all feasible tables w.r.t. $\mathcal{Q}$.*

Definition 5.1.4 (Induced Neighbors $\mathcal{N}_{\mathcal{Q}}$ [43]) *Given background knowledge $\mathcal{Q}$, a pair of fact tables T_a and T_b are said to be neighbors induced by $\mathcal{Q}$, denoted as $\mathcal{N}_{\mathcal{Q}}(T_a, T_b) = \mathrm{true}$, iff (i) $T_a, T_b \in \mathcal{T}_{\mathcal{Q}}$, and (ii) for any shortest sequence of moves $M_{ab} = \mathrm{m}_1, \mathrm{m}_2, \ldots, \mathrm{m}_{d_{ab}}$ that transform T_a into T_b, no subsequence of M_{ab} can transform T_a into some other $T_c \in \mathcal{T}_{\mathcal{Q}}$. We also denote the set of all induced neighbors of T by $\mathcal{Q}$ as $\mathcal{N}_{\mathcal{Q}}(T)$.*

Definition 5.1.5 (Generic Differential Privacy [43]) *Given background knowledge $\mathcal{Q}$, a randomized algorithm $\mathcal{K}$ is ϵ-differentially private if for any two tables T_1 and T_2 that are* *neighbors induced by $\mathcal{Q}$ and any subset S of the output of $\mathcal{K}$,*

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

where the probability is taken over the randomness of $\mathcal{K}$ and ϵ is a fixed value.

Definition 5.1.5 can be specialized into the original notation of differential privacy in Definition 2.1.1, if we set the background knowledge $\mathcal{Q}$ to be empty.

To preserve generic differential in data publishing subject to background knowledge, the Laplace mechanism [26] can be generalized as follows.

Definition 5.1.6 (Generic Sensitivity [43]) *Consider any function $F : \mathcal{T} \rightarrow \mathcal{R}^n$ and background knowledge $\mathcal{Q}$, the L^1 global generic sensitivity of F w.r.t. $\mathcal{Q}$ is:*

$$S_{\mathcal{Q}}(F) = \max_{T_2 \in \mathcal{T}} \left(\max_{T_1 \in \mathcal{N}_{\mathcal{Q}}(T_2)} \|F(T_1) - F(T_2)\|_1 \right),$$

where $\|x - y\|_1 = \sum_{1 \leq i \leq n} |x_i - y_i|$ is the L^1 distance between two n-dimensional vectors $x = \langle x_1, x_2, \ldots, x_n \rangle$ and $y = \langle y_1, y_2, \ldots, y_n \rangle$.

Theorem 5.1.1 (Generic Laplacian mechanism [43]) *Let $F : \mathcal{T} \rightarrow \mathcal{R}^n$ be a query sequence of length n against a table $T \in \mathcal{T}$, and $\mathcal{Q}$ be the background knowledge that has to be released exactly. The randomized algorithm that takes as input table T and output $\tilde{F}(T) = F(T) + \langle \mathrm{Lap}(S_{\mathcal{Q}}(F)/\epsilon) \rangle^n$ is ϵ-differentially private.*

Example 5.1.1 *Consider the fact table T in Table 1.1 with three dimensions:* Sex, Age, *and* Salary. *Background knowledge is a set of cuboids that are released to or known to adversaries exactly. For example, if the adversary knows how many people are male and how many are female, the background knowledge $\mathcal{Q}$ contains one cuboid* {Sex}. *If in addition to that, the adversary knows the count histogram on people's age, i.e., the cuboid* {Age} *is released exactly, then the background knowledge $\mathcal{Q} = \{\{\text{Sex}\}, \{\text{Age}\}\}$.*

5.2 Preserving Generic Differential Privacy

We need to compute generic sensitivity $S_{\mathcal{Q}}$ in order to apply the generic Laplacian mechanism, and [43] shows that it is hard to estimate the generic sensitivity for general data type and background knowledge. But, in Section 5.2, we will show that it is possible to estimate $S_{\mathcal{Q}}$ in data cubes when the background knowledge is a set of exact cuboids (Definition 5.1.1). In Section 5.2.2, we will introduce two useful properties: *projection* and *composition*.

5.2.1 Estimating Generic Sensitivity

Consider a d-dim fact table T with attributes $\mathcal{A} = \{A_1, A_2, \ldots, A_d\}$. Recall $|A_i|$ is the cardinality of dimension A_i. Let C_{base} be the d-dim base cuboid. For a cuboid C, recall $[C]$ is the set of its dimensions. For a subset of dimensions $\mathcal{A}' \subseteq \mathcal{A}$, let $\text{size}(\mathcal{A}') = \prod_{A_i \in \mathcal{A}'} |A_i|$. Define $\text{size}(\emptyset) = 1$. $\text{size}([C])$ is essentially the number of all possible cells in cuboids C.

Kifer and Machanavajjhala in [43] study a special case for 2-dim fact tables.

Lemma 5.2.1 ([43]) *Consider a 2-dim fact table T with two attributes A_1 and A_2. (i) If the background knowledge $\mathcal{Q} = \{\{A_1\}\}$ (row sums or column sums), the generic sensitivity of publishing C_{base} is $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 2$. (ii) If $\mathcal{Q} = \{\{A_1\}, \{A_2\}\}$ (both row sums and column sums), the generic sensitivity of publishing C_{base} is $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 2\min\{|A_1|, |A_2|\}$.*

We will generalize Lemma 5.2.1 for d-dim fact table T, with background knowledge $\mathcal{Q}$ as any one or two cuboids. Again we want to publish the base cuboid C_{base}, subject to $\mathcal{Q}$.

Lemma 5.2.2 *Consider a d-dim fact table T. (i) If the background knowledge $\mathcal{Q} = \{C\}$ for any cuboid C, the generic sensitivity of publishing C_{base} is $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 2$. (ii) If $\mathcal{Q} = \{C_1, C_2\}$ for any two cuboids, then $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 2\min\{\text{size}([C_1] - [C_2]), \text{size}([C_2] - [C_1])\}$.*

Proof: The proof for (i) is trivial and is very similar to the proof for (i) in Lemma 5.2.1.

For (ii), let's first prove $S_{\mathcal{Q}}(C_{\mathsf{base}}) \leq 2\min\{\text{size}([C_1] - [C_2]), \text{size}([C_2] - [C_1])\}$.

Recall $\mathcal{T}$ is the set of all d-dim fact tables with attributes $\mathcal{A} = \{A_1, A_2, \ldots, A_d\}$, and $\mathcal{T}_{\mathcal{Q}} \subseteq \mathcal{T}$ is the set of all feasible fact tables subject to background knowledge $\mathcal{Q}$. Consider any two feasible fact tables $T_a, T_b \in \mathcal{T}_{\mathcal{Q}}$, it suffices to show that, if a shortest sequence M_{ab} of moves that transforms T_a into T_b is longer than $2\min\{\text{size}([C_1] - [C_2]), \text{size}([C_2] - [C_1])\}$, a subsequence of these moves can transform T_a into another feasible table $T_c \in \mathcal{T}_{\mathcal{Q}}$. Then, for any two (induced) neighboring fact table T_a and T_b, we must have the length of the shortest move sequence $|M_{ab}| \leq 2\min\{\text{size}([C_1] - [C_2]), \text{size}([C_2] - [C_1])\}$, and thus we can have $S_{\mathcal{Q}}(C_{\mathsf{base}}) \leq |M_{ab}| \leq 2\min\{\text{size}([C_1] - [C_2]), \text{size}([C_2] - [C_1])\}$, because the L^1 distance between the base cuboids on the two tables T_a and T_b is no more than $|M_{ab}|$.

For a cell a in C_{base}, and the two cuboids C_1 and C_2, recall $a[C_i]$ is the projection of a on $[C_i]$, i.e., the cell in C_i which has the same attributes values as a in $[C_i]$.

We construct a directed bipartite graph as follows. The vertices are cells in C_1 and C_2. In M_{ab}, for a move which increases the count $\mathsf{c}(a)$ of a cell a in C_{base} by one (*positive move*), we create an edge from $a[C_1]$ to $a[C_2]$; and for a move which decreases the count $\mathsf{c}(a)$ by one (*negative move*), we create an edge from $a[C_2]$ to $a[C_1]$. Because both T_a and T_b are feasible, we have the same number of positive moves incident on $a[C_i]$ as the number of negative moves; or, in this graph, each vertex has the same in-degree as the out-degree.

A directed sub-cycle in this graph corresponds to a subset of the moves which transform T_a into another feasible table T_c (because one positive move and one negative move incident on cells of C_1 and C_2 do not violate the background knowledge). So we want to show that, if $|M_{ab}| > 2\min\{\text{size}([C_1] - [C_2]), \text{size}([C_2] - [C_1])\}$, there exists such a nontrivial proper sub-cycle, which contracts to the fact that T_a and T_b are induced neighbors.

It is easy to see that, if $|M_{ab}| > 2\min\{\text{size}([C_1]), \text{size}([C_2])\}$, there exists such a sub-cycle. There is an Eulerian directed walk W (starting and ending at the same vertex) in this graph, because each vertex has the same number of in-degree as out-degree. $|M_{ab}| > 2\min\{\text{size}([C_1]), \text{size}([C_2])\}$ implies that some vertex v in C_1 or C_2 has degree at least 4, and thus it is visited at least twice in W (i.e. $W = u \dots vW'v \dots u$). So we can extract a nontrivial proper sub-cycle from W at vertex v (it could be $vW'v$).

To get the tight bound $|M_{ab}| \leq |2 \min\{\text{size}([C_1] - [C_2]), \text{size}([C_2] - [C_1])\}$, we note that the graph has been divided into $\text{size}([C_1] \cap [C_2])$ connected components in nature. Because, according to the construction, any two cells $a_1 \in C_1$ and $a_2 \in C_2$ are connected only if a_1 and a_2 have the same values in the attributes $[C_1] \cap [C_2]$. And in each component, we have $\text{size}([C_1] - [C_2])$ vertices from C_1 and $\text{size}([C_2] - [C_1])$ vertices from C_2 (and each vertex has the same in-degree as the out-degree). So based on an argument similar to the one for $|M_{ab}| \leq 2 \min\{\text{size}([C_1]), \text{size}([C_2])\}$, we can prove this tight bound as well.

To prove $S_{\mathcal{Q}}(C_{\mathsf{base}})$ is indeed equal to $2 \min\{\text{size}([C_1] - [C_2]), \text{size}([C_2] - [C_1])\}$, we construct a cycle with length $2 \min\{\text{size}([C_1] - [C_2]), \text{size}([C_2] - [C_1])\}$, which corresponds to a shortest sequence M_{ab} of moves between two feasible tables. It satisfies two properties: (i) the base cuboids on the two tables have L^1 distance equal to $|M_{ab}|$; and (ii) no subset of M_{ab} can yield another feasible table. It can be constructed as follows: Number all possible assignments of values to attributes in $[C_1] - [C_2]$ as $1, 2, \dots, \text{size}([C_1] - [C_2])$, and all possible assignments of values to attributes in $[C_2] - [C_1]$ as $1, 2, \dots, \text{size}([C_2] - [C_1])$. Consider such a cycle: $(1, x) \to (x, 1) \to (2, x) \to (x, 2) \to \dots \to (1, x)$, where (i, x) is a cell in C_1 with value i in attributes $[C_1] - [C_2]$ and value x in $[C_1] \cap [C_2]$, and similarly (x, j) is a cell in C_2 with j in $[C_2] - [C_1]$ and x in $[C_1] \cap [C_2]$. This cycle has length $2 \min\{\text{size}([C_1] - [C_2]), \text{size}([C_2] - [C_1])\}$. Easy to very this cycle satisfies properties (i) and (ii). So the proof is completed. $\square$

Note that Lemma 5.2.2 also considers a special case: when $C_1 \preceq C_2$, $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 2$.

Example 5.2.1 *Consider the fact table T in Table 1.1 with three dimensions:* Sex*,* Age*, and* Salary*. Two cuboids $C_1 = \{\mathsf{Sex}, \mathsf{Age}\}$ and $C_2 = \{\mathsf{Age}, \mathsf{Salary}\}$ are released exactly ($\mathcal{Q} = \{C_1, C_2\}$). From Lemma 5.2.2, the sensitivity of releasing $C_{\mathsf{base}} = \{\mathsf{Sex}, \mathsf{Age}, \mathsf{Salary}\}$ subject to $\mathcal{Q}$ is* $2 \min\{\text{size}([C_1] - [C_2]), \text{size}([C_2] - [C_1])\} = 2 \min\{|\mathsf{Sex}|, |\mathsf{Salary}|\} = 4$.

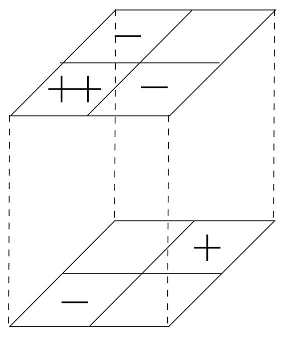


Figure 5.1: A shortest sequence of moves for $2 \times 2 \times 2$ case

When $|\mathcal{Q}| \geq 3$, the problem of estimating the sensitivity $S_{\mathcal{Q}}(C_{\mathsf{base}})$ is quite open. We only have results for some special cases. For example, when $\mathcal{Q} = \{\{A_1\}, \{A_2\}, \{A_3\}\}$, where the cardinalities of attributes are $|A_1| = 2$, $|A_2| = 2$, and $|A_3| = x$: if $x = 2$ or 3, we have $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 6$ (refer to Figure 5.1), and if $x = 4$, we have $S_{\mathcal{Q}}(C_{\mathsf{base}}) = 8$.

From these example, we may observe that, for background knowledge with more than two cuboids, the sensitivity $S_{\mathcal{Q}}$ is *not* linearly dependent on the cardinalities of two or three dimensions in the table, but dependent on some "blocking" structures.

An Extremal Combinatorial Problem

Let's model/generalize the sensitivity computation problem in data cubes as an extremal combinatorics problem, which is a widely open problem even in combinatorial mathematics.

Let $\mathcal{V}(n,k) = \{v \mid v \in \{0,1\}^n \cup \{0,-1\}^n, \text{ and } \|v\|_1 = k\}$ be the set of all n-dim vectors, each with $(n-k)$ '0'-entries and either k '1'-entries or k '-1'-entries. A vector set $V \subseteq \mathcal{V}(n,k)$ is said to be *stationary* if $\sum_{v \in V} v = \mathbf{0}$. A stationary vector set V is *irreducible* if no non-empty proper subset $V' \subset V$ is stationary. We consider such an extremal problem: given n and k, what is the maximal size of an irreducible stationary vector set $V \subseteq \mathcal{V}(n,k)$? Denote this maximal size by $S(n,k)$. Our original sensitivity computation problem actually corresponds to a restricted version, in which the n coordinates are partitions into k sets, and we considered only vectors $\mathcal{V}(n,k)$ that have one nonzero entry in each vector. Here, corresponding to our problem, n is the sum of cardinalities of attributes and $k = |\mathcal{Q}|$ is the number of cuboids the adversary has as background knowledge. Any meaning lower/upper bound on $S(n, k)$ implies lower/upper bound on the sensitivity in our problem.

Adapting the proof for Lemma 5.2.2, we actually have proved the following result.

Lemma 5.2.3 *$S(n, 1) = 2$ and $S(n, 2) = n$.*

Without coming up with the proof, we conjecture that:

Conjecture 1 *Given n and k, $S(n, k) = \Theta(n^{k/2})$.*

A relevant extremal problem. In a relevant problem, we have $\mathcal{V}(n, k) = \{v \mid v \in \{0, 1\}^n, \text{ and } ||v||_1 \leq k\}$. Let $N(n, k, r)$ be the maximal size of a vector set $\subseteq \mathcal{V}(n, k)$ s.t. each r vectors in this set are linearly independent modulo 2 (i.e., even number of ones in each of the n coordinates). Suppose r is closed to n, this number is known as $n^{\Theta(k/2)}$ according to a sequence of work [49, 50, 6, 7, 65]. This problem seem to be similar to our problem, but their techniques cannot be directly applied.

5.2.2 Useful Properties of Generic Differential Privacy

There are two useful properties of generic differential privacy, *projection* and *composition*.

The *projection* property is about how much the background knowledge is projected into a subspace (cuboid) of the fact table. Note that Lemma 5.2.2 is about the sensitivity of releasing the base cuboid C_{base} subject to the background knowledge $\mathcal{Q}$. A natural question is: subject to $\mathcal{Q}$, how much is the sensitivity of releasing any cuboid C? The following lemma says that, the sensitivity stays almost unchanged for all the cuboids C (except the ones that can be computed from $\mathcal{Q}$), if the background knowledge $\mathcal{Q}$ is fixed.

Lemma 5.2.4 (Projection property) *In a data cube, if the background knowledge $\mathcal{Q}$ is fixed, for any cuboid C that cannot be computed from $\mathcal{Q}$, we have $S_{\mathcal{Q}}(C) \leq S_{\mathcal{Q}}(C_{\mathsf{base}})$. If $[C] - \bigcup_{C_i \in \mathcal{Q}}[C_i] \neq \emptyset$, $S_{\mathcal{Q}}(C) = S_{\mathcal{Q}}(C_{\mathsf{base}})$; otherwise, $\frac{1}{2}S_{\mathcal{Q}}(C_{\mathsf{base}}) \leq S_{\mathcal{Q}}(C) \leq S_{\mathcal{Q}}(C_{\mathsf{base}})$*

Proof: $S_{\mathcal{Q}}(C) \leq S_{\mathcal{Q}}(C_{\mathsf{base}})$ is obvious. For any two induced neighbors T_a and T_b, let C^a and C^b be the cuboids constructed from T_a and T_b, respectively, on dimensions $[C] = [C^a] = [C^b]$. Similarly for C^a_{base} and C^b_{base}. Then we must have $\|C^a - C^b\|_1 \leq \|C^a_{\mathsf{base}} - C^b_{\mathsf{base}}\|_1$.

If there is an attribute A_0 in $[C] - \bigcup_{C_i \in \mathcal{Q}}[C_i]$, consider any two induced neighbors T_a, T_b, and a shortest sequence M_{ab} of moves that transforms T_a into T_b. A_0 has at least two values, say, x and y. Then for each positive move in M_{ab}, we can enforce it add a tuple with $A_0 = x$, and for each negative move in M_{ab}, we can enforce it delete a tuple with $A_0 = y$. In other words, positive moves and negative moves do not cancel out each other in C. So we must have $\|C^a - C^b\|_1 = |M_{ab}| = \|C^a_{\mathsf{base}} - C^b_{\mathsf{base}}\|_1$, and thus $S_{\mathcal{Q}}(C) = S_{\mathcal{Q}}(C_{\mathsf{base}})$.

If $[C] - \bigcup_{C_i \in \mathcal{Q}}[C_i] = \emptyset$, we can carefully arrange the positive moves and negative moves in M_{ab} within C, so that most of them do not cancel out each other. In most cases, we still can show that $S_{\mathcal{Q}}(C) = S_{\mathcal{Q}}(C_{\mathsf{base}})$, and in the worst case, $S_{\mathcal{Q}}(C) \geq \frac{1}{2}S_{\mathcal{Q}}(C_{\mathsf{base}})$. □

The *transition* and *composition* properties allow the combining of the publications or outcomes of several differentially private algorithms. These two properties still hold for generic differential privacy as long as the background knowledge $\mathcal{Q}$ is fixed. The statement and the proof are similar to Theorem 2.1.3. We repeat the statement for completeness:

Theorem 5.2.5 (Transition and composition properties) *Let $\mathcal{K}_i(\cdot)$ or $\mathcal{K}_i(\cdot,\cdot)$ be a randomized algorithm which is ϵ_i-differentially private subject to background knowledge $\mathcal{Q}$.*

1. *For a table T, outputting $\mathcal{K}_1(T)$ and $\mathcal{K}_2(T, \mathcal{K}_1(T))$ is $(\epsilon_1 + \epsilon_2)$-differentially private subject to background knowledge $\mathcal{Q}$.*

2. *For any input table T and any algorithm $\mathcal{K}$, outputting $\mathcal{K}_1(T)$ and $\mathcal{K}(\mathcal{K}_1(T))$ is ϵ_1-differentially private subject to background knowledge $\mathcal{Q}$.*

3. *For any two input tables T_1 and T_2 such that $T_1 \cap T_2 = \emptyset$, outputting $\mathcal{K}_1(T_1)$ and $\mathcal{K}_1(T_2)$ is ϵ_1-differentially private subject to background knowledge $\mathcal{Q}$.*

5.3 Optimizing Noise on Background Knowledge

For a fact table T, to publish a set $\mathcal{L} \subseteq \mathcal{L}_{\mathsf{all}}$ of cuboids subject to certain background knowledge $\mathcal{Q}$, we can inserted Laplace noise into C_{base} according to Lemma 5.2.2 and Theorem 5.1.1, and compute other cuboids from it. A better way is to apply the noise optimization technique $\mathcal{K}_{\mathsf{part}}$ introduced in Section 3.2 or $\mathcal{K}_{\mathsf{gen}}$ introduced in Section 3.3. Following is a revised version of $\mathcal{K}_{\mathsf{gen}}$, denoted as $\mathcal{K}^{\mathcal{Q}}_{\mathsf{gen}}$ to handle the background knowledge.

Recall $\mathcal{K}^{\mathcal{Q}}_{\mathsf{gen}}$ chooses to compute cuboids $\mathcal{L}_{\mathsf{pre}}$ directly from the fact table T, and then adds noise $\mathrm{Lap}(\lambda_C/\epsilon)$ to cells in $C \in \mathcal{L}_{\mathsf{pre}}$. $\mathcal{L}_{\mathsf{post}} = \mathcal{L} - \mathcal{L}_{\mathsf{pre}}$ includes all the other cuboids in $\mathcal{L}$. Again, we do *not* require $\mathcal{L}_{\mathsf{pre}} \subseteq \mathcal{L}$. $\mathcal{K}^{\mathcal{Q}}_{\mathsf{gen}}$ works as follows.

1. (*Noise Sources*) For each cell a in cuboid $C \in \mathcal{L}_{\mathsf{pre}}$, $\mathcal{K}^{\mathcal{Q}}_{\mathsf{gen}}$ computes $\mathsf{c}(a)$ from T but releases $\tilde{\mathsf{c}}(a) = \mathsf{c}(a) + \mathrm{Lap}(\lambda_C/\epsilon)$. $\mathcal{L}_{\mathsf{pre}}$ is selected s.t. $\mathcal{L}_{\mathsf{post}}$ can be computed from $\mathcal{L}_{\mathsf{pre}}$.

2. (*Aggregation*) For each cuboid $C \in \mathcal{L}_{\mathsf{post}}$, $\mathcal{K}^{\mathcal{Q}}_{\mathsf{gen}}$ 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^*. C^* is picked in the same way as in $\mathcal{K}_{\mathsf{gen}}$ (refer to Section 3.3.1).

The major difference between $\mathcal{K}_{\mathsf{gen}}$ and $\mathcal{K}^{\mathcal{Q}}_{\mathsf{gen}}$ is the constraint which the noise parameters $\{\lambda_C\}$ must satisfy to preserve (generic) ϵ-differential privacy.

Theorem 5.3.1 *$\mathcal{K}^{\mathcal{Q}}_{\mathsf{gen}}$ is ϵ-differentially private subject to $\mathcal{Q}$ if $\sum_{C \in \mathcal{L}_{\mathsf{pre}}} 1/\lambda_C = 1/S_{\mathcal{Q}}(C_{\mathsf{base}})$. For any released cell a in $\mathcal{L}$, $\tilde{\mathsf{c}}(a)$ is an unbiased estimator of $\mathsf{c}(a)$, i.e., $\mathrm{E}\left[\tilde{\mathsf{c}}(a)\right] = \mathsf{c}(a)$.*

Proof: The first part can be proved using the composition property of generic differential privacy (Theorem 5.2.5). And the second part is similar to Theorem 3.2.3. □

Also similar to $\mathcal{K}_{\mathsf{gen}}$, we can define different noise-control objectives of $\mathcal{K}^{\mathcal{Q}}_{\mathsf{gen}}$ as in Problems 3 and 4. Our algorithms introduced in Section 3.3.2 can be revised to find such $\mathcal{L}_{\mathsf{pre}}$ and noise parameters $\{\lambda_C \mid C \in \mathcal{L}_{\mathsf{pre}}\}$, aiming at minimizing the max noise in the released cuboids, or maximizing the number (weight) of cuboids with noise variance $\leq \theta_0$.

Chapter 6

Experimental Study

We will exam techniques proposed in this thesis using both real datasets and synthetic datasets. Our goal is to verify the analytical and theoretical results about our techniques, and demonstrate their effectiveness on real data. Different techniques are compared experimentally to show their own advantages and disadvantages.

6.1 Experiment Setting and Datasets

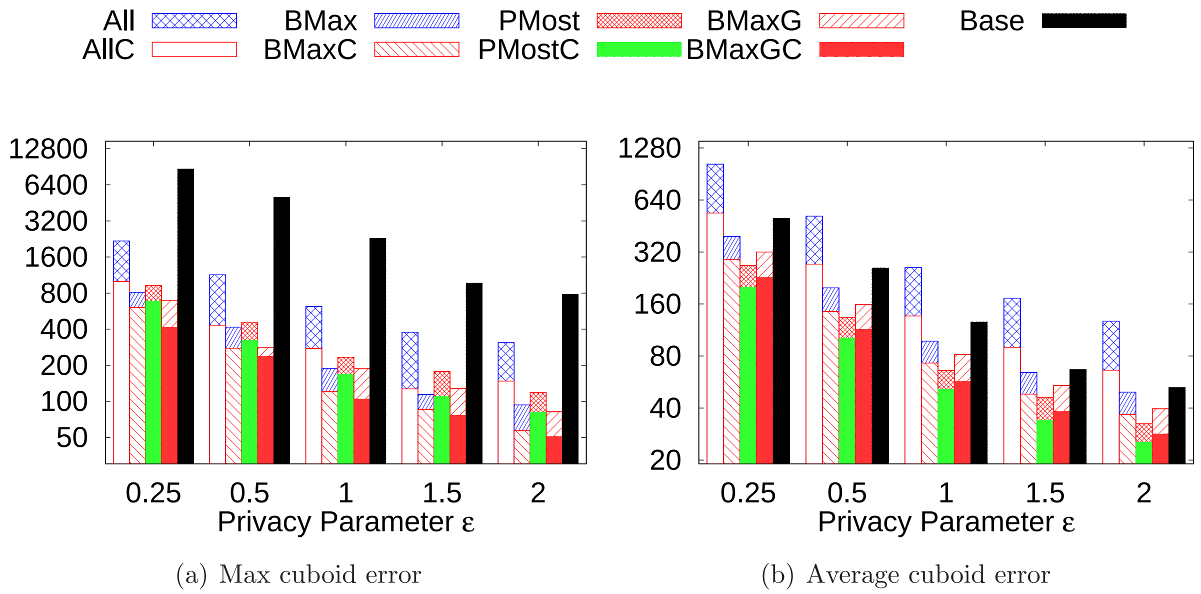
Techniques for comparison. We evaluate and compare nine proposed techniques on both real datasets and synthetic datasets, together with their application to publish data cubes subject to exact background knowledge. The first two techniques are $\mathcal{K}_{\mathsf{all}}$ and $\mathcal{K}_{\mathsf{base}}$, which were defined at the beginning of Section 3.2. We denote them as All and Base, respectively, in this chapter. Another two are based on our noise-control framework, cuboid selection, $\mathcal{K}_{\mathsf{part}}$. One of these has the objective of bounding the max noise variance (Problem 1) and is denoted by BMax ($\mathcal{K}_{\mathsf{part}}$ + 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 ($\mathcal{K}_{\mathsf{part}}$ + Algorithm 2). We also test our more general noise-control framework, distribution selection, $\mathcal{K}_{\mathsf{gen}}$ (different amounts of Laplace noise are inserted into different initial cuboids). We implement the algorithm that selects an initial set of cuboids and the amounts of noise to be inserted into each of them with the goal of minimizing the max noise variance (Problem 3), and denote is as BMaxG ($\mathcal{K}_{\mathsf{gen}}$ + Algorithm 3). The last four techniques apply the methods in Section 4.3 to enforce consistency in the data cubes published by All ($\mathcal{L}_{\mathsf{pre}} = \mathcal{L}$), BMax, PMost, and BMaxG. In particular, the unweighted version of the L^2-minimization consistency-enforcing method in Section 4.3.1 is applied for All, BMax, and PMost, resulting in three techniques denoted as AllC, BMaxC, and PMostC, respectively. The weighted version in Section 4.3.2 is applied for BMaxG because different amounts of noise are injected into different cuboids; and the resulting technique is denoted as BMaxGC. The LP-based techniques in Section 4.2 are not practical for large tables. All seven algorithms are coded in C++ and evaluated on an 8GB 64-bit 2.40GHz PC.

We will specify $\mathcal{L}$, the set of cuboids to published, and the privacy parameter ϵ, for these algorithms. All the algorithms are evaluated on the same set of published cuboids.

Real dataset. We use the Adult dataset and the Nursery dataset from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/) in the experiments. The Adult dataset was extracted from the census bureau database with 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). The Nursery dataset was derived from a hierarchical decision model originally developed to rank applications for nursery schools. It has 12,960 rows and 9 categorical dimensions: parents (cardinality 3), has_nurs (5), form (4), children (4), housing (3), finance (2), social (3), health (3), and class (5).

Synthetic dataset. To generate synthetic fact tables, 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 nine ϵ-differentially private algorithms. The *cuboid error* is the average error for all cells in this cuboid. In the following experiments, we report both the *max cuboid error* and the *average cuboid error* among all published cuboids in $\mathcal{L}$.



(a) Max cuboid error

(b) Average cuboid error

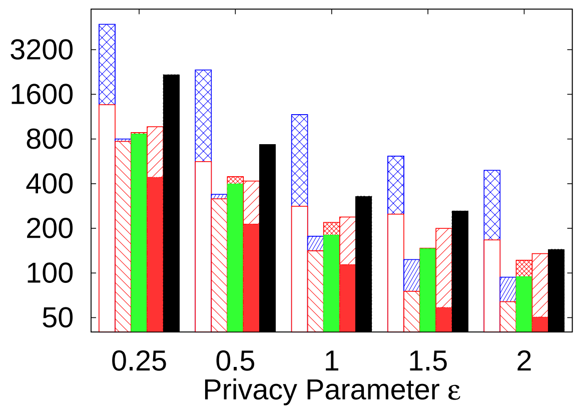
Figure 6.1: Varying privacy parameter ϵ in the Adult dataset

6.2 Experiments

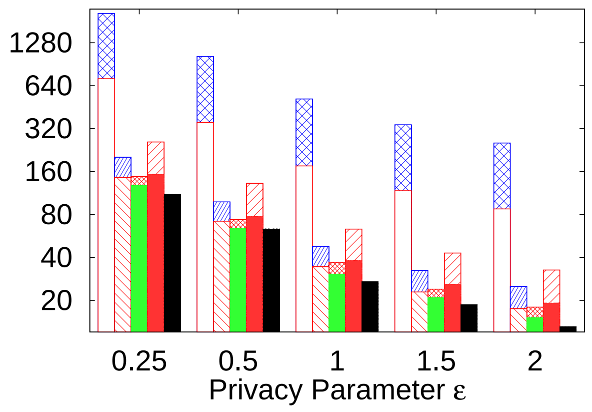
6.2.1 Varying Privacy Parameter ϵ

We vary privacy parameter ϵ from 0.25 to 2. Smaller ϵ implies more privacy and thus more noise. We use our techniques to publish all cuboids ($\mathcal{L} = \{\text{all cuboids}\}$). 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).

For the Adult dataset, Figure 6.1 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, PMost/PMostC, and BMaxG/BMaxGC) are stacked together on a single bar in every histogram in Section 6. BMax, PMost, and BMaxG always incur much less error than the two baselines Base and All. BMaxG is better than BMax and PMost for bounding the max error, while PMost and BMaxG are better 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



(a) Max cuboid error



(b) Average cuboid error

Figure 6.2: Varying privacy parameter ϵ in the Nursery dataset

noise is magnified significantly when cuboids at higher levels are computed by aggregating cells in the base cuboid. All is the worst 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 4.3.1 (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/BMaxGC reduces error by 30%-50%, compared to All/BMax/PMost/BMaxG, while providing the same privacy guarantee. Among all the algorithms, with consistency enforced, BMaxGC performs the best in terms of the max cuboid error, and PMostC is the best in terms of the average cuboid error (BMaxGC is close).

The results on the Nursery datasets are depicted in Figure 6.2. We can observe similar trends. BMaxGC is the best in terms of the max cuboid error, while PMostC performs well for bounding the average cuboid error. We note that, in this dataset, Base is the best in terms of the average cuboid error, but it performs badly for bounding the max cuboid error, which implies it is not stable – sometimes, it may generate small error (in low-level cuboids), but sometimes it may also generate unacceptably large error (in high-level cuboids). Such observations coincide with the analysis in Section 3.2. BMaxC, PMostC, and BMaxGC are close to Base for bounding the average error, but are much better for the max error.

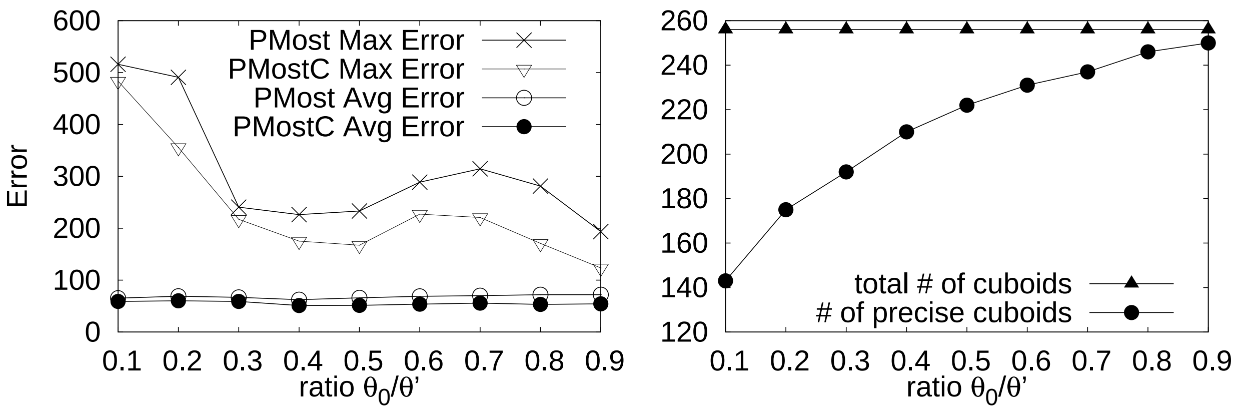
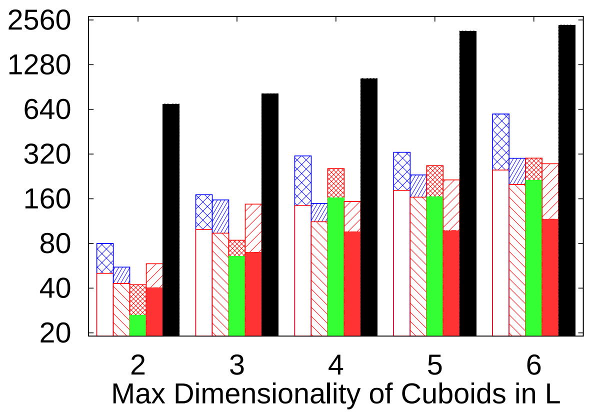


Figure 6.3: Varying θ_0 in Algorithm 2 (for PMost and PMostC)

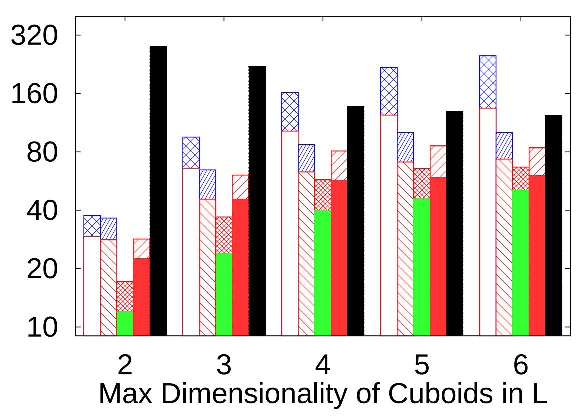
6.2.2 Varying Variance Threshold θ_0

We use the Adult dataset to study how to set the parameter θ_0 in Algorithm 2. Recall that θ_0 is a variance threshold. An initial set $\mathcal{L}_{\mathsf{pre}}$ of cuboids is selected by Algorithm 2, which maximizes the number of *precise cuboids* in $\mathcal{L}$, i.e., cuboids that can be computed from $\mathcal{L}_{\mathsf{pre}}$ with noise variance no more than θ_0 (Problem 2). PMost and PMostC use those cuboids $\mathcal{L}_{\mathsf{pre}}$ to compute all cuboids in $\mathcal{L}$ (in this experiment, let $\mathcal{L} = \{\text{all cuboids}\}$).

Note that performance of PMost and PMostC depends on the input threshold θ_0 in Problem 2, as θ_0 determines $\mathcal{L}_{\mathsf{pre}}$. 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 6.3 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 $\mathcal{L}_{\mathsf{pre}}$ 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 $\theta_0 = 0.5\theta'$.



(a) Max cuboid error



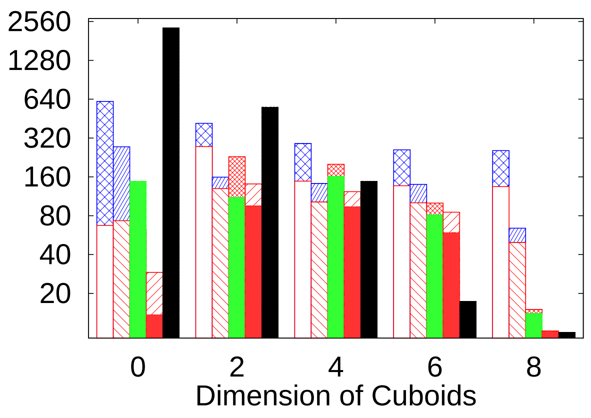
(b) Average cuboid error

Figure 6.4: Varying the set $\mathcal{L}$ of cuboids to be published

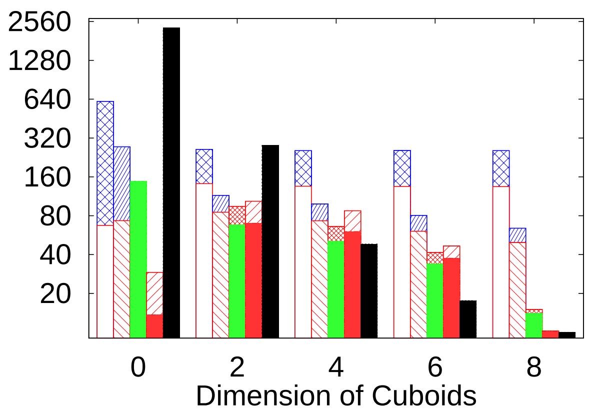
6.2.3 Varying Set of Cuboids to be Published

In this experiment, we vary $\mathcal{L}_{\mathsf{pre}}$, the set of cuboids to be published, in the Adult dataset, and fix $\epsilon = 1$. The results are reported in Figure 6.4 for $\mathcal{L} = \{$cuboids with dimensionality no more than 2$\}$, which aggregate the fact table on no more than 2 dimensions, $\mathcal{L} = \{$cuboids with dimensionality no more than 3$\}$, $\ldots$, and $\mathcal{L} = \{$cuboids with dimensionality no more than 6$\}$. Recall in an m-dim cuboid, a cell has non-$*$ values on exactly m dimensions. We now focus on these higher-level cuboids, because they are frequently used in data mining algorithms, for example, decision tree construction and naive Bayes classifier.

From the results, we can find that Base performs badly for these high-level cuboids. With the consistency enforced, AllC performs better, but the best two are always PMostC and BMaxGC. When $|\mathcal{L}|$ is smaller, e.g., it contains only cuboids with no more than 2 dimensions, PMostC is the best. When $|\mathcal{L}|$ becomes larger, e.g., from 4 to 6, BMaxGC becomes much better especially in terms of the max noise. This is because the distribution selection framework $\mathcal{K}_{\mathsf{gen}}$ is more general, which enable BMaxGC to find a better choice of the initial set of cuboids and the amounts of noise to be inserted into each of them.



(a) Max cuboid error



(b) Average cuboid error

Figure 6.5: Max cuboid error as dimensionality varies, when all cuboids are released

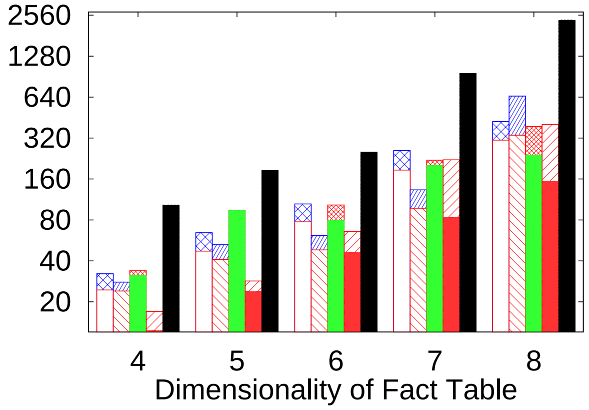
6.2.4 Noise in Different Cuboids

Now suppose all cuboids in the Adult dataset are to be published ($\mathcal{L} = \{\text{all cuboids}\}$). Figure 6.5 shows the max and average error in cuboids of different dimensionalities, for $\epsilon = 1$. The max error and the average error are calculated for cuboids with m dimensions, separately, to illustrate how noise is distributed among different levels of the data cube.

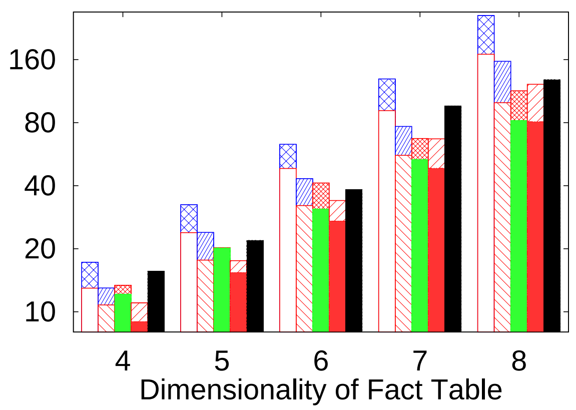
As m decreases, a cell in an m-dim cuboid aggregates more rows and base cells. So Base aggregates more noise from base cells as m drops, and performs the worst for $m < 3$. Although it is the best for low-level cuboids with $m \geq 6$, its performance deteriorates very quickly as m decreases. The consistency-enforcing techniques, AllC/BMaxC/PMostC/BMaxGC, are very effective for small m, reducing error by up to 70%. BMaxGC is the best when $0 < m < 5$ in terms of max error, and changes little in all different cuboids. PMostC and BMaxGC are the best two when $0 < m < 4$ in terms of average error.

6.2.5 Scalability: Number and Cardinality of Dimensions

We generate synthetic tables with 4 to 8 dimensions, where each dimension has the same cardinality 7, and add 10^8 rows randomly into each of them. We fix $\epsilon = 1$ and aim to

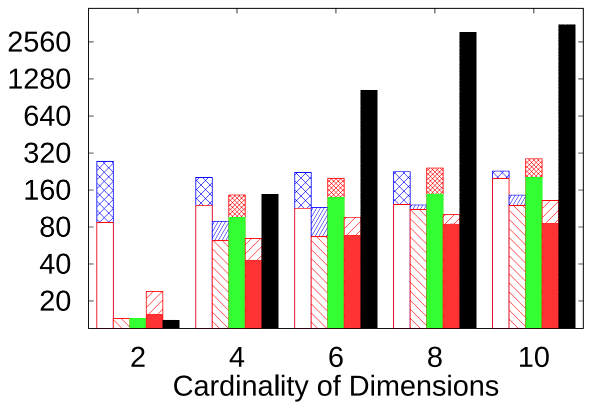


(a) Max cuboid error

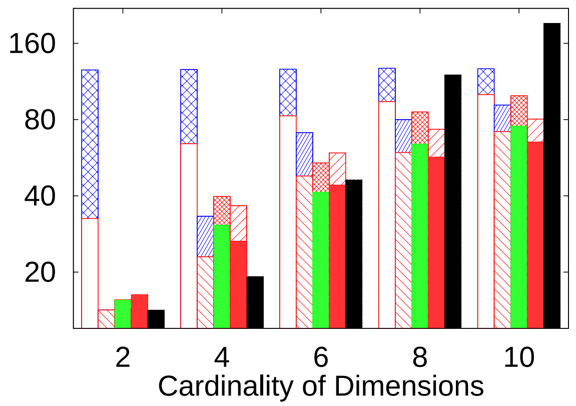


(b) Average cuboid error

Figure 6.6: Varying dimensionality of fact table (cardinality equals to 7)



(a) Max cuboid error



(b) Average cuboid error

Figure 6.7: Varying cardinality of dimensions (7 dimensions)

publish all cuboids. We report the max error and the average error of the seven approaches in Figure 6.6. As the number of dimensions increases, BMaxGC is always the best one, in terms of both max and average error. BMaxC is better than PMostC in terms of max error, while PMostC is better than BMaxC in terms of average error. The error in both All and Base is much larger and increases faster than the error in others.

We then 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 6.7, for publishing all cuboids with

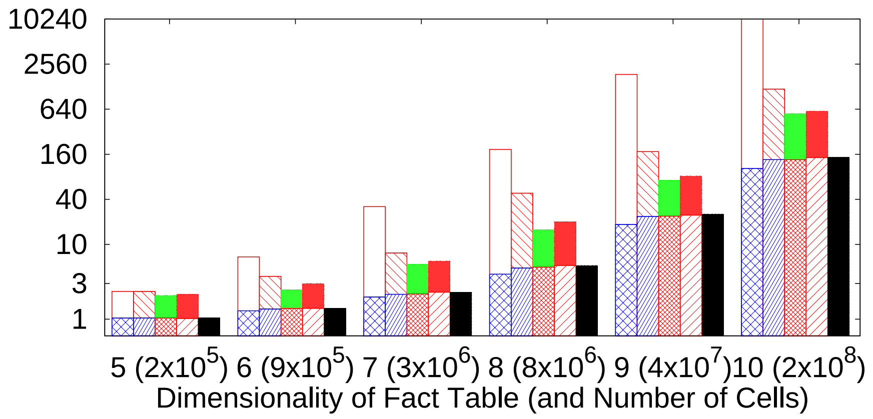
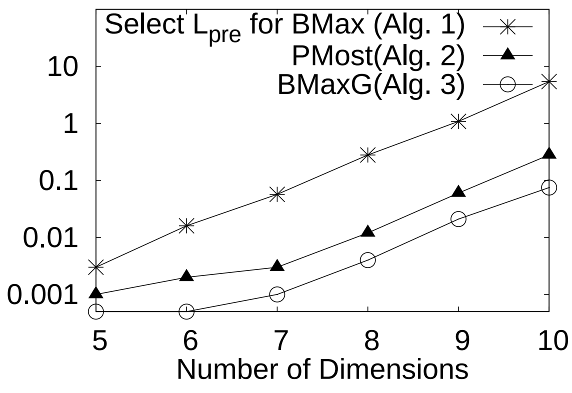
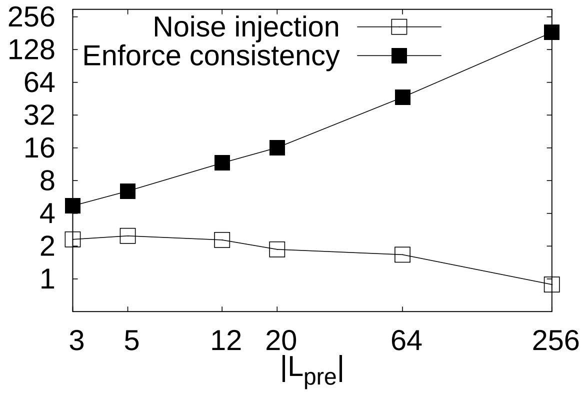


Figure 6.8: Differentially private data cube publishing time (in seconds)



(a) Time (in seconds)



(b) Time (in seconds)

Figure 6.9: Efficiency of different algorithms (subroutines of publishing algorithms)

$\epsilon = 1$. 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, because its performance is mainly determined by the number of cuboids, which is determined by the total number of dimensions and is fixed here. However, All is not as good as approaches based on our noise-control frameworks. Again, BMaxGC is the best almost all the time, and otherwise, BMaxC and PMostC perform the best.

6.2.6 Efficiency of Different Publishing Algorithms

To test the efficiency of different publishing algorithms, we create two more dimensions, each of which has cardinality 4, on the fact table in the Adult dataset, and generate their values randomly for each row. We consider the data cubes generated from the fact table on the first 5 to all the 10 dimensions. The time to publish the whole data cube is reported in Figure 6.8, which can be decomposed as follows. Recall that Algorithms 1-3 select $\mathcal{L}_{\mathsf{pre}}$ for BMax/BMaxC, PMost/PMostC, and $\{\lambda_C\}$ for BMaxG/BMaxGC, respectively. The running time of Algorithms 1-3 is reported in Figure 6.9(a). For different choices of $\mathcal{L}_{\mathsf{pre}}$ on the 8-dim table, the time needed for noise injection ($\mathcal{K}_{\mathsf{part}}$ and $\mathcal{K}_{\mathsf{gen}}$) and consistency enforcement (the L^2-minimization method in Section 4.3) in all cuboids is reported in Figure 6.9(b).

From Figure 6.9(a), although the running time of Algorithms 1-3 is polynomial in 2^d, they are not the bottleneck in data cube publishing. Their running time is always a very small portion of the overall differentially private data cube publishing time.

From Figure 6.9(b), it is shown that the consistency-enforcement time increases linearly with $|\mathcal{L}_{\mathsf{pre}}|$, as predicted by Theorem 4.3.1 (ii). The time for noise injection decreases as $|\mathcal{L}_{\mathsf{pre}}|$ increases. This is because when more cuboids are initially injected with noise, less aggregation of noise is needed to compute all the other cuboids later on.

From Figure 6.8, using consistency enforcement, AllC is especially expensive, because $|\mathcal{L}_{\mathsf{pre}}| = 2^d$ in AllC. BMaxC, PMostC, and BMaxGC, although using consistency enforcement, usually only need 3%-10% time of AllC, because they use smaller $\mathcal{L}_{\mathsf{pre}}$'s. For all approaches, the publishing time increases exponentially with the dimensionality, mainly because the total number of cells increases exponentially, which is depicted on the x-axis.

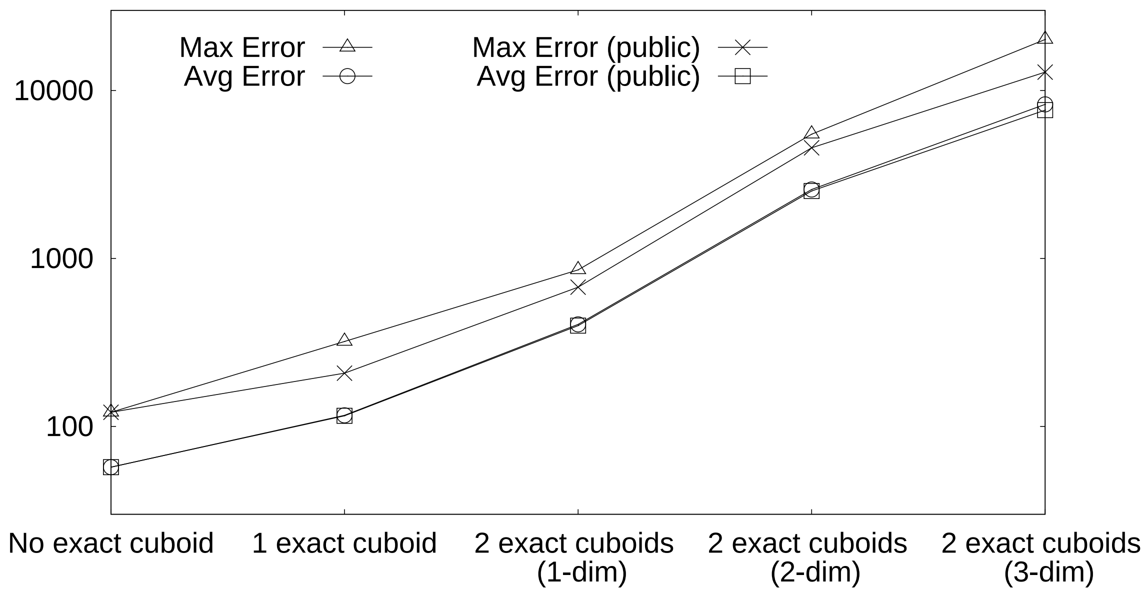


Figure 6.10: Exact cuboids (background knowledge) v.s. noise

6.2.7 Background Knowledge v.s. Noise

We now study the relationship between the background knowledge and the error incurred in the published cuboids. We use the Adult dataset, fix $\epsilon = 1$, and aim to publish all the cuboids subject to some background knowledge (some exact cuboids). We consider five cases: 1) no background knowledge, 2) one exact cuboid as the background knowledge, 3) two exact cuboids each with one dimensions, 4) two exact cuboids each with two dimensions, and 5) two exact cuboids each with three dimensions. The one or two cuboids for each case are selected randomly from all such cuboids. We apply our BMaxGC algorithm, as it is the best one according to our previous experiments. Noise is calibrated based on Lemma 5.2.2. Assuming the exact cuboids are known only to adversaries, we report both max error and average error in Figure 6.10, as "Max Error" and "Avg Error", respectively.

We observe that, with two exact cuboids, the amount of error increases exponentially in the number of dimensions. This is because according to Lemma 5.2.2, to preserve individuals' privacy, the amount of noise needed is proportional to the number of cells in the two exact cuboids. So the direct implication is that, when we have to release two or more cuboids exactly to adversaries, we should avoid releasing the ones with more dimensions (i.e., low-level cuboids with more details, e.g., $\{\mathsf{Age}, \mathsf{Sex}, \mathsf{Salary}, \mathsf{Company}\}$). It is acceptable to release one cuboid or two high-level cuboids (e.g., $\{\mathsf{Age}\}$ and $\{\mathsf{Sex}\}$) while preserving privacy.

On the other hand, we also note that if the exact cuboids are public knowledge, i.e., known to both adversaries and normal users, publishing them can reduce noise as their ancestor cuboids can be computed from them exactly. More specifically, if $[C_1] \subseteq [C_2]$ and C_2 is known exactly to everyone as the background knowledge, then C_1 can be published exactly without violating the promised privacy guarantee. Keeping this in mind, we re-calculate the max error and average error and report them in Figure Figure 6.10, as "Max Error (public)" and "Avg Error (public)", respectively. We find that, in this case, the max error drops while the average error almost stays the same. This is because a few cuboids can be computed directly from the exact cuboids with zero error, and these cuboids are usually high-level ones which should have had large absolute error, so the max error is reduced. But the number of such cuboids is small, so the average cuboid error is almost unchanged.

6.3 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. BMaxGC and PMostC are the best most of the time. The data cube published by them has reasonably small amounts of noise in all cuboids. Although the base cuboid published by BMaxGC and PMostC may have more noise than the one published by Base, both the average noise and the max noise across all cuboids published by BMaxGC and PMostC are significantly smaller than noise in all cuboids published by Base. This is because in Base, other cuboids aggregate lots of noise from the base cuboid.

BMax/PMost/BMaxG run much faster than BMaxC/PMostC/BMaxGC. But with our consistency enforcement, BMaxC/PMostC/BMaxGC reduce error in BMax/PMost/BMaxG, 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 the error of All, and 20%-50% of the error of AllC. Note that the y-axis in our figures is always in logarithmic scale. Using a more general noise-control framework, BMaxGC is even better than BMaxC and PMostC, especially in terms of the max noise, and reduces error by an addition of 20%-40%.

Because of our consistency-enforcing method, the error of AllC is sometimes comparable to BMaxG/BMaxGC 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 4.3.1 for the running time of our consistency enforcement). When there are more than five dimensions, AllC's publishing time is 10-30 times larger than BMaxGC's ($>$ 30 times for ten dimensions), and 10-40 times larger than PMostC's ($>$ 40 times for ten dimensions).

It is also verified that exact cuboids incur large error in private publishing.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we study how to publish a differentially private data cube in order to support privacy-preserving data analytics tasks. We address three key issues: data quality (control noise variance), data consistency, and robustness to background knowledge.

For the issue of data quality, we propose noise control frameworks which insert noise to some selected cuboids. These initial cuboids together with different amounts of noise injected into them are called "noise sources". The remaining cuboids are computed from these noise sources. Three optimization problem of choosing the noise sources with different objectives are considered: (i) minimizing the maximal noise over all published cuboids; (ii) maximizing the number of cuboids with noise below a given threshold; and (iii) minimizing the maximal relative ratio between real noise variances and user-specified variance thresholds over all published cuboids. Although problems (i)-(iii) are all NP-hard, we propose efficient approximation algorithms for them ($(\ln|\mathcal{L}|+1)^2$-approximation for (i) and (iii), where $|\mathcal{L}|$ is the number of cuboids to be published, and $(1-1/e)$-approximation for (ii)).

For the issue of data consistency, we introduce L^p-distance-minimization framework to compute a consistent measure from the noisy measure released from our noise control frameworks. Previous work in [5] is a special case in our framework ($p=\infty$), and we show that the L^1 version yields a much better theoretical bound on error than the L^∞ version. More surprisingly, we show that in the L^2 version, the consistent measure provides even better utility than the original inconsistent noisy measure, while the same level of privacy is preserved. The main idea is based on the theory of (weighted) least squares, but we provide efficient algorithms to compute such consistent measures, with running time linear in the number of cells, while previous techniques have at least quadratic running time.

For the robustness to background knowledge, we try to plug the notation of generic differential privacy into our noise-control frameworks. We consider some cuboids that have to be released exactly as the background knowledge, and generalize results in [43] about estimating generic sensitivity to preserve differential privacy subject to the background knowledge. The generic sensitivity essentially stands for the relationship between the amount of background knowledge an adversary has and the amount of noise needed to be injected.

Techniques proposed in this thesis are both proved to be sound in theory and evaluated experimentally to verify their effectiveness in practice. They provide advanced principles and major parts of a complete solution to privacy-preserving publishing of data cubes.

7.2 Future Work

We list several interesting future research directions in this part.

- *Online noise source optimization problem.* As discussed in Section 3.4.3, cuboid queries may be asked one by one in large data cubes, and we have no knowledge about which queries will be asked in future. Noise sources $\mathcal{L}_{\mathsf{pre}}$ and $\{\lambda_C \mid C \in \mathcal{L}_{\mathsf{pre}}\}$ need to be updated in an online fashion so that once a cuboid queries comes, it must be answered at once, i.e., covered by $\mathcal{L}_{\mathsf{pre}}$ and $\{\lambda_C \mid C \in \mathcal{L}_{\mathsf{pre}}\}$. It is an online decision problem of how to updating $\mathcal{L}_{\mathsf{pre}}$ and $\{\lambda_C \mid C \in \mathcal{L}_{\mathsf{pre}}\}$ so that, e.g., the max noise over all the queries (to be answered or have been answered) is minimize. It has some connection to the Online Set Cover problem, but is even harder. Our preliminary result is an $O(\log^5 |L|)$-competitive online algorithm (i.e. the solution found online is within a factor $O(\log^5 |L|)$ of the offline optimal after we see all queries). Whether this competitive ratio can be improved and whether certain assumption on the distribution of incoming queries helps make the problem easier are open questions.

- *Estimating generic sensitivity.* Lemma 5.2.2 gives the generic sensitivity subject to background knowledge of two cuboids (i.e. two cuboids are published exactly). How to compute the generic sensitivity subject to more than two cuboids is still open.

- *Generic differential privacy subject to approximate background knowledge.* As indicated by Lemma 5.2.2, to preserve generic differential privacy subject to non-trivial background knowledge, we often need to inject a large amount of noise into the cuboids to be released. The noise variance is proportional to the size of cuboids (the number of cells) in the case of two exact cuboids, and thus may make the released cuboids meaningless. A natural idea is to relax the background knowledge from exact cuboids to approximate cuboids. For example, instead of releasing exact cuboids, we may release information like "number of rows satisfying property P is within a range $[x-\delta, x+\delta]$" as the background knowledge. Whether and how much such approximate background knowledge helps reduce the sensitivity and thus the noise variance are open.

- *Assumption about adversaries.* In many studies, even using the original notation of privacy, differentially private algorithms usually insert too much noise to the released data or the query answers. The reason is simply that the strongest possible adversaries are considered in differential privacy, with the infinite computational power and infinite background knowledge. Varying the mysterious parameter ϵ does not help as this assumption stays the same. An interesting and fundamental question is: whether we can build a hierarchy of privacy notations and guarantees based on the amount of background knowledge and the computational power available to adversaries. Then users can select one based their need for balance between privacy and utility.

- *Privacy on emerging data types and system models.* This is a more general question. It is worth our effort to study data privacy on emerging data types, like Big Data, and system models, like Cloud Computing and Storage, in both of which privacy becomes an even more important issue. Differential privacy will meet lots of new challenges, and we are wondering whether it can be adapted there, maybe after relaxation (we will have more flexibility on choosing noise sources in those scenarios), or we need to propose completely new notations or policies about “privacy”.

References

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

[2] Rakesh Agrawal, Ramakrishnan Srikant, and Dilys Thomas. Privacy preserving olap. In *SIGMOD*, pages 251–262, 2005.

[3] Sumeet Bajaj and Radu Sion. Trusteddb: a trusted hardware based database with privacy and data confidentiality. In *SIGMOD Conference*, pages 205–216, 2011.

[4] Sumeet Bajaj and Radu Sion. Trusteddb: A trusted hardware based outsourced database engine. *PVLDB*, 4(12):1359–1362, 2011.

[5] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS*, pages 273–282, 2007.

[6] Claudia Bertram-Kretzberg, Thomas Hofmeister, and Hanno Lefmann. Sparse 0-1-matrices and forbidden hypergraphs (extended abstract). In *SODA*, pages 181–187, 1998.

[7] Claudia Bertram-Kretzberg, Thomas Hofmeister, and Hanno Lefmann. Sparse 0-1-matrices and forbidden hypergraphs. *Combinatorics, Probability & Computing*, 8(5):417–427, 1999.

[8] Raghav Bhaskar, Srivatsan Laxman, Adam Smith, and Abhradeep Thakurta. Discovering frequent patterns in sensitive data. In *KDD*, pages 503–512, 2010.

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

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge Univ. Press, 1 edition, 2004.

[11] T.-H. Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *Privacy Enhancing Technologies*, pages 140–159, 2012.

[12] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *NIPS*, pages 289–296, 2008.

[13] Francis Y. L. Chin and Gultekin Özsoyoglu. Auditing and inference control in statistical databases. *IEEE Trans. Software Eng.*, 8(6):574–582, 1982.

[14] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* The MIT Press and McGraw-Hill Book Company, 2001.

[15] Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *ICDE*, pages 20–31, 2012.

[16] Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Grigory Yaroslavtsev. Accurate and efficient private release of datacubes and contingency tables. *arXiv*, arXiv:1207.6096v1, 2012.

[17] János Demetrovics, Gyula O. H. Katona, and Dezsö Miklós. On the security of individual data. *Ann. Math. Artif. Intell.*, 46(1-2):98–113, 2006.

[18] Bolin Ding, Haixun Wang, Ruoming Jin, Jiawei Han, and Zhongyuan Wang. Optimizing index for taxonomy keyword search. In *SIGMOD Conference*, pages 493–504, 2012.

[19] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms.* Cambridge Univ. Press, 1 edition, 2009.

[20] Cynthia Dwork. Differential privacy. In *ICALP (2)*, pages 1–12, 2006.

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

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

[23] Cynthia Dwork. Differential privacy in new settings. In *SODA*, pages 174–183, 2010.

[24] Cynthia Dwork. The promise of differential privacy: A tutorial on algorithmic techniques. In *FOCS*, pages 1–2, 2011.

[25] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

[26] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[27] Cynthia Dwork and Moni Naor. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality*, 2(1):93–107, 2010.

[28] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *STOC*, pages 715–724, 2010.

[29] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N. Rothblum, and Sergey Yekhanin. Pan-private streaming algorithms. In *ICS*, pages 66–80, 2010.

[30] Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. Private coresets. In *STOC*, pages 361–370, 2009.

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

[32] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4), 2010.

[33] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *KDD*, pages 265–273, 2008.

[34] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. In *STOC*, pages 351–360, 2009.

[35] Michaela Götz, Ashwin Machanavajjhala, Guozhang Wang, Xiaokui Xiao, and Johannes Gehrke. Publishing search logs - a comparative study of privacy guarantees. *IEEE Trans. Knowl. Data Eng.*, 24(3):520–532, 2012.

[36] Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In *ICDE*, pages 152–159, 1996.

[37] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. *CoRR*, abs/1012.4763, 2010.

[38] Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, pages 61–70, 2010.

[39] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *STOC*, pages 705–714, 2010.

[40] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially-private queries through consistency. In *PVLDB*, pages 1021–1032, 2010.

[41] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *FOCS*, pages 531–540, 2008.

[42] Daniel Kifer. Attacks on privacy and definetti's theorem. In *SIGMOD*, pages 127–138, 2009.

[43] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *SIGMOD*, pages 193–204, 2011.

[44] Daniel Kifer and Ashwin Machanavajjhala. A rigorous and customizable framework for privacy. In *PODS*, pages 77–88, 2012.

[45] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately. In *WWW*, pages 171–180, 2009.

[46] Andreas Krause and Carlos Guestrin. A note on the budgeted maximization of submodular functions. Technical Report CMU-CALD-05-103, 2005.

[47] Denny Lee. Differential privacy: Case studies. In *http://www.cs.cmu.edu/ CompThink/mindswaps/oct07/*, 2007.

[48] Jaewoo Lee and Chris Clifton. Differential identifiability. In *KDD*, 2012.

[49] Hanno Lefmann, Pavel Pudlák, and Petr Savický. On sparse parity chack matrices (extended abstract). In *COCOON*, pages 41–49, 1996.

[50] Hanno Lefmann, Pavel Pudlák, and Petr Savický. On sparse parity check matrices. *Des. Codes Cryptography*, 12(2):107–130, 1997.

[51] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. Optimizing histogram queries under differential privacy. In *PODS*, pages 123–134, 2010.

[52] Chao Li and Gerome Miklau. An adaptive mechanism for accurate query answering under differential privacy. *PVLDB*, 5(6):514–525, 2012.

[53] Chao Li and Gerome Miklau. Measuring the achievable error of query sets under differential privacy. *CoRR*, abs/1202.3399, 2012.

[54] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, pages 106–115, 2007.

[55] Xiaolei Li, Jiawei Han, and Hector Gonzalez. High-dimensional olap: A minimal cubing approach. In *VLDB*, pages 528–539, 2004.

[56] Yang D. Li, Zhenjie Zhang, Marianne Winslett, and Yin Yang. Compressive mechanism: utilizing sparse representation in differential privacy. In *WPES*, pages 177–182, 2011.

[57] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.

[58] Ashwin Machanavajjhala, Daniel Kifer, John M. Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, pages 277–286, 2008.

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

[60] Frank McSherry. Personal communication. 2012.

[61] Frank McSherry and Ilya Mironov. Differentially private recommender systems: building privacy into the netflix prize contenders. In *KDD*, pages 627–636, 2009.

[62] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[63] Darakhshan J. Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca N. Wright. Pan-private algorithms via statistics on sketches. In *PODS*, pages 37–48, 2011.

[64] Noman Mohammed, Rui Chen, Benjamin C. M. Fung, and Philip S. Yu. Differentially private data release for data mining. In *KDD*, pages 493–501, 2011.

[65] Assaf Naor and Jacques Verstraëte. Parity check matrices and product representations of squares. *Combinatorica*, 28(2):163–185, 2008.

[66] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.

[67] Shangfu Peng, Yin Yang, Zhenjie Zhang, Marianne Winslett, and Yong Yu. Dp-tree: indexing multi-dimensional data under differential privacy (abstract only). In *SIGMOD Conference*, page 864, 2012.

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

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

[70] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *Technical Report SRI-CSL-98-04*, 1998.

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

[72] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):571–588, 2002.

[73] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[74] Lingyu Wang, Sushil Jajodia, and Duminda Wijesekera. Securing olap data cubes against privacy breaches. In *IEEE Symposium on Security and Privacy*, pages 161–175, 2004.

[75] Lingyu Wang, Sushil Jajodia, and Duminda Wijesekera. Preserving privacy in on-line analytical processing data cubes. In *Secure Data Management in Decentralized Systems*, pages 355–380. 2007.

[76] Lingyu Wang, Duminda Wijesekera, and Sushil Jajodia. Cardinality-based inference control in data cubes. *Journal of Computer Security*, 12(5):655–692, 2004.

[77] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[78] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. Minimality attack in privacy preserving data publishing. In *VLDB*, pages 543–554, 2007.

[79] Xiaokui Xiao, Gabriel Bender, Michael Hay, and Johannes Gehrke. ireduct: differential privacy with reduced relative errors. In *SIGMOD Conference*, pages 229–240, 2011.

[80] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. In *ICDE*, pages 225–236, 2010.

[81] Yonghui Xiao, James J. Gardner, and Li Xiong. Dpcube: Releasing differentially private data cubes for health information. In *ICDE*, pages 1305–1308, 2012.

[82] Yonghui Xiao, Li Xiong, and Chun Yuan. Differentially private data release through multidimensional partitioning. In *Secure Data Management*, pages 150–168, 2010.

[83] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Ge Yu. Differentially private histogram publication. In *ICDE*, pages 32–43, 2012.

[84] Yin Yang, Zhenjie Zhang, Gerome Miklau, Marianne Winslett, and Xiaokui Xiao. Differential privacy in data publication and analysis. In *SIGMOD Conference*, pages 601–606, 2012.

[85] Ganzhao Yuan, Zhenjie Zhang, Marianne Winslett, Xiaokui Xiao, Yin Yang, and Zhifeng Hao. Low rank mechanism: Optimizing batch queries under differential privacy. *PVLDB*, 2012.

[86] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. Functional mechanism: Regression analysis under differential privacy. *PVLDB*, 2012.