Optimizing the Range-Count Queries in Differential Privacy

Ç®ÀÚ[1,*]

近些年来数据应用领域中的数据隐私保护问题越来越引起大家的关注。ε-differential privacy 提供了强有力的隐私保障，但由于噪音的干扰使得它在保护隐私的同时影响了数据的可用性和一致性。这对于 Differential Privacy 框架在对隐私数据的处理上提出了新的要求，如何在保护隐私的前提下提升发布数据的可用性和分析准确度成了 Differential Privacy 框架研究的一个关键。

DiffGen 是一种非交互式的 ε-differential privacy 数据发布框架，基于属性值重匿名方式对隐私数据进行处理并加以发布，提供给用户进行数据挖掘等后续使用。但它对于数据条目计数值的直接加噪处理会影响范围计数（rang-count）类分析的准确性，随着查询范围的增加，加噪数据的可用性将逐渐降低。本文针对范围计数查询，在保障 ε-differential privacy 的提前下，利用数据关系间的一致性特征优化 DiffGen 的构建模型，赋以新的敏感性定义，针对树型处理结构调整噪音的分布替代原先简单在叶节点加噪的方案，使之返回更加准确的范围计数结果，最终达到能在后续的数据分析应用中提升数据可用性的目的。

´ [1] 差分隐私；范围计数；一致性；敏感性；后续利用

1 INTRODUCTION

对于数据拥有者而言，数据中所包含的隐私信息是在数据发布中特别需要注重和保护的问题。传统的保护方式虽然能保护数据的隐私信息，但是它们都是基于特殊的攻击假设和背景知识，一旦攻击者掌握了某些隐私数据的前景信息 [6] 或者进行组合攻击 [5] 的时候，传统的隐私保护方法就变得不那么安全可靠了。ε-differential privacy 已经成为了新的隐私保护模型，它提供了强有力的隐私保障，通过在隐私数据中添加适当的噪音达到保护的目的，而不管攻击者拥有多少的隐私数据相关的背景知识和攻击手段。

不同于每次答复数据访问者的交互式框架,DiffGen[7] 是一种满足 ε-differential privacy 的非交互式的隐私保护框架。通过对各属性建立新的区间抽象 (Partition) 规则，对原始数据集构建树形处理结构，从根节点开始按照规则对数据集进行重新划分 (Specialization)，最后处理落在最底层，即叶节点上的重匿名数据集并发布，并且发布出数据集为加噪处理后的安全数据集，供用户使用。期间的隐私代价一半用在选择划分属性和处理连续值上，另

[*] µç×ÓÓÊŒþ£º lywu@amt.ac.cn

一半作为拉普拉斯因子往每条数据项的计数上加噪。但这种往最终得到的每个数据项上直接加噪的处理方法，对于后续的范围计数类型应用而言，由于噪音叠加问题会产生较差的应用效果，这和 Dwork et al.'s[8]、Barak et al.'s[**?**] 采用的列联表的加噪方式在原理上是一样的。

表 1: Real Estate Records and Frequency Matrix

(a) Real Estate Records

Age	$HasHouse(s)$
<20	No
20-30	Yes
30-40	No
20-30	No
40-50	Yes
30-40	Yes
<20	No
30-40	No

(b) Frequency Matrix

Age	$YesCount$	$NoCount$
<20	0	2
20-30	1	1
30-40	1	2
40-50	1	0

1. 如表 1 所示，随机统计的 8 个不同年龄层人房产有无情况的调查，统计结果如表 (a)，表 (b) 为其频率矩阵，YesCount 表示有房产的计数，NoCount 表示无房产的计数，每个条目的表示记为 {年龄范围，Yes 计数，No 计数}，如小于 20 岁行项的条目为 {<20,0,2}，加噪之后可能变成 {<20,4,1}。

在例 1 中，为了保护所统计的年龄-房产情况表的隐私，Dwork et al.'s 的方法是往频率矩阵的每个行项添加方差为 $\Theta(1)$ 的噪音，也就是敏感性为 1，隐私数据之间相互独立。但是，这种逐行添加独立噪音的方式，对于范围查询会有 $\Theta(m)$ 的噪音方差，其中 m 为频率矩阵的行数，也就是噪音会按行叠加，带来查询维度相关的问题 [16]。如查询 20-40 岁人群的房产情况即为一个范围查询，需要对表 (b) 的年龄属性 {20-30} 以及 {30-40} 的 YesCount 和 NoCount 进行累加返回。显然，查询涉及的行数越多，就会发生越多次的噪音叠加，使得返回结果准确性降低，而这也正是 DiffGen 框架在涉及范围查询时面临的问题。

1.1 Contributions

很多的算法和应用是对加噪数据的后续利用，这对于 Differential Privacy 框架在对隐私数据的处理上提出了新的要求，仅仅为了保护隐私可能会打破数据的内在联系，使得发布出的数据无法使用，如何提升后续分析应用的准确度成了 Differential Privacy 框架研究的一个关键。对于范围计数类的查询应用，DiffGen 的加噪模式碰到了和上节提到的频率矩阵的加噪模式一样的问题——返回结果的噪音误差并不独立于数据集本身，和查询范围相关。

本文针对决策树构建、分类算法等需要涉及范围计数类型的应用，在不改变 DiffGen 框架数据发布模式的前提下，利用数据关系间的一致性联系对其加噪环节进行优化，使得噪音的分布更加合理，尽可能减少数据项叠加操作带来的影响。本文贡献总结如下：

1. 针对发布数据后的范围计数类应用，总结了 DiffGen 框架存在着和列联表加噪一样的问题，噪音方差是与查询维度相关，噪音的叠加将降低发布数据的可用性。
2. 改进 DiffGen 模型。归纳出 DiffGen 框架上的数据集存在的一致性特性，可以利用这个特性优化后续的范围查询应用，提升发布数据的可用性，但需要解决敏感性改变之后噪音的优化问题和尽量不暴露内部节点的安全性问题。结合 Hay et al.[12] 的分析，可对 DiffGen 数据结构的噪音分布进行调整，使之在满足 ε-differential privacy 的前提下，减少查询维度相关带来的影响；并且由于节点集在加噪后重获一致性特性，因此可以采用原 DiffGen 的数据发布模式，安全性得到了保障。
3. 优化误差估计。对于范围查询，原先 DiffGen 的加噪误差是和查询行数相关的方差估计，现优化为 DiffGen 算法构建的数据集树结构的树高相关的方差估计。
4. 改进 Hay et al. 的 UNIVERSAL HISTOGRAMS 模型，拓展完全 κ 叉树情况至任意树情况。

1.2 Related Work

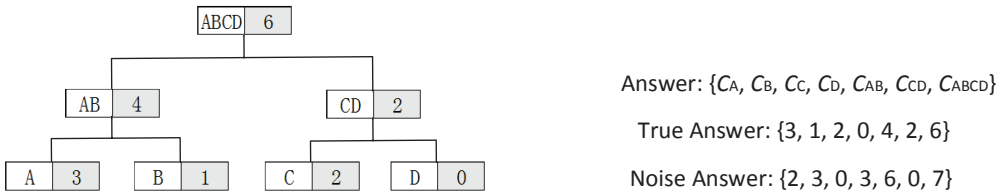
噪音叠加是由于范围查询涉及到多个行项，简单的单属性列联表不能很好地说明可能涉及行项之多，但在多属性数据集中往往随着查询范围的增加使得涉及到的行项数目呈爆炸式递增，并且最简单的单属性查询此时也变得复杂起来。因此，对于属性及属性维度较多的实际使用中的数据集来说，噪音叠加是个不可避免的问题。

Cormode et al.'s[**?**] 用 domain size 表示整个数据集空间的维度大小。假设数据集有 d 个属性，A_1，A_2，...，A_2，每个属性 A_i 有 $|A_i|$ 个属性值，那么这个数据集的 domain size 为 $\prod_{i=1}^{d} |A_i|$。如数据集有 10 个属性，每个属性维度为 10，这将是个 10^{10} 大小的维度，维护数据关系的表格庞大，并且各属性关联复杂，而这种数据集在现实情况中很常见。

Zhang et al.'s[9] 讨论了这种数据维度的影响，由于 domain size 大小的缘故哪怕仅查询单个属性，可能由其他属性的影响仍需要涉及众多行项。如例 1 中，此时添加了一个"国籍"属性, 它有 10 个属性值, 那么当查询 20-30 年龄层的房产情况时, 由原先仅需关注 20-30 年龄层这一行数据，变成可能需要涉及额外的 10 行数据项——在"国籍"属性列中找出 Age 值为 20-30 的所有行项, "国籍"共 10 个属性值，最多可涉及 10 行。

因此，当确定了查询范围为 A_k，A_{k+1}，...，A_j，$k \leqslant j$，那么额外涉及到的行项数目为 $\prod_{i\in(1,d)-(k,j)} |A_i|$，显然查询涉及到的行项数目是和数据集维度相关的，随着属性以及属性值维度的增加势必会导致更多的行项参与叠加操作。

不仅查询涉及范围广这一方面，在大维度数据集中查询相应范围的属性值也是个消耗巨大的工程。为了加速查询过程，通常会对所有的查询情况进行统计存储，即采用空间换



(a) Tree Structure of String (b) Query Answer

图 1: Example for Consistency

时间的做法，生成各种属性组合的列联表 (Cuboid)[15]，对各种查询 marginals[13] 做应答。但是这些辅助查询的列联表 (Cuboid) 的维护可能会变得很困难，因为较大 domain size 的列联表所占空间大小常常超过了数据集本身的大小。若这个频率矩阵是个稀疏矩阵 [16] 的时候，会有很多行项的计数值为零，加噪使得结果不为零，因此叠加计数的时候会使得噪音占了计数的主导部分 [11]。

Zhang et al.'s[9] 通过建立贝叶斯网络模型来优化这些问题。他们把高纬度的列联表加噪方式转换为低维度的贝叶斯网络上的条件概率加噪，然后根据加噪概率构建新数据集并发布。由于贝叶斯网络相对于原数据集是个低维空间，在低维空间上的加噪操作能够极大地限制 domain size 产生的影响。Xiao et al.'s[14] 在 Wavelet 模型中也指出针对范围计数情况下的噪音叠加问题。由于噪音的方差为查询涉及到的行数 m 相关，他们通过小波变换原理给隐私代价设置权值，减小噪音方差，降低拉普拉斯算子计算出的噪音量级，达到优化范围查询的目的。

因此，减少噪音叠加或针对范围查询问题，多属性数据集本身可能就决定了查询往往涉及众多的行项，往频率矩阵每个行项独立加噪的方法会严重影响发布数据的可用性，而这也是 DiffGen 算法在返回范围查询时存在的问题。Hay et al.[12] 就直方图的计数应用问题，提出了围绕数据关系间的一致性联系来增加处理后数据的准确度，给 DiffGen 的优化提供了思路，在本文的接下来章节中详细阐述。

2 PRELIMINARIES

在这一节中，首先展示差分隐私的概念以及满足差分隐私的核心机制；然后就之前的讨论，总结 DiffGen 的加噪模式在范围类计数应用中存在着的问题；最后介绍相关的改进思路。

2.1 Differential Privacy

¶¨ **1.** (ε-differential privacy)

¶¨ **2.** (Sensitivity)

LaplaceMechanism.

Exponential Mechanism.

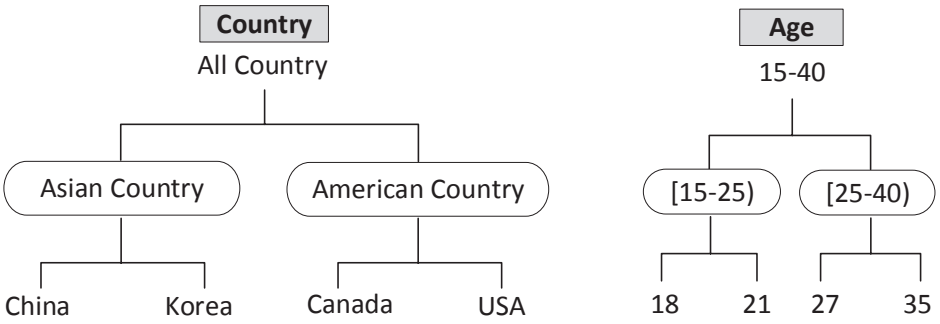


图 2: Taxonomy tree of attributes

2.2 DiffGen

DiffGen 是 Noman Mohammed et al.[7] 提出的一个基于匿名算法的非交互式差分隐私框架。数据拥有者分别对隐私数据的各个属性进行属性值的区间构建处理 (Partition)，构建具有层级关系的属性值层次结构, 针对隐私数据集建立了树形的划分规则 (Specialization)。然后通过指数机制挑选分裂属性, 对原始数据集依照这个树形结构自上而下地进行划分, 最后收集叶节点数据条目，通过拉普拉斯机制加噪后发布。以下简要介绍这个框架。

表 2: Raw data to be generalized with DiffGen

(a) Raw data table

Country	***Age***	***Class***
China	18	N
Korea	21	Y
Canada	27	N
USA	35	N
USA	29	Y
China	39	Y
Korea	22	N
China	28	N

(b) Generalized data table

Country	***Age***	***ClassCount***
Asian Country	[15-25)	3
Asian Country	[25-40)	2
American Country	[15-25)	0
American Country	[25-40)	3

2. 一家公司的某个岗位聘用求职者情况如表 2(a) 所示，共有 8 个求职者，有 2 个属性 :“国家”和“年龄”，分别表示他们的国籍和年龄，类属性为 Y 和 N，表示是否被录用。

对例 2 的“国家”和“年龄”属性进行匿名抽象处理，按表 2(a) 构建树形分裂规则——**属性树**,树中的每个非叶子节点均为重匿名的抽象属性值。就像树结构父子节点间的联系,这些属性值之间存在着一致性联系——从叶节点自下而上来看,这是属性值的抽象归并 (Generalization) 形成新的区间的过程，决定着原始属性如何组合并入新属性；从根节点自上而

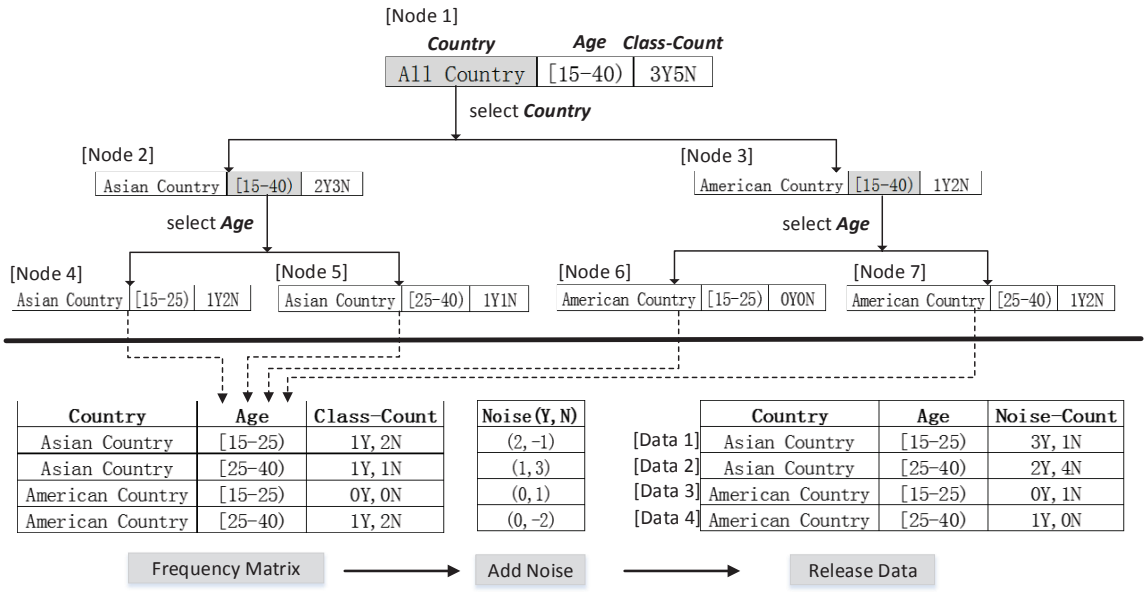


图 3: Partitioning records with tree structure

下来看这就是对节点制定了一套分裂 (Specialization) 规则，即从新属性节点如何分裂产生子属性集。

DiffGen 的核心思想就是基于这些属性树，依照属性值的划分规则，对原始数据集从根节点开始的划分，最终发布出重匿名的新数据集。

3. 如表 2 的数据集所示，属性“国家”具有属性值中国、美国、韩国、加拿大四个属性值，通过事先定义的匿名规则可抽象出图 1 的层次结构，如中国和韩国抽象并入亚洲国家这个更大的区间，而亚洲国家和美洲国家最终归并入根节点————所有国家，成为这个属性最大的区间，这样就完成了属性“国家”的属性树的构建，年龄属性树同理。在选择最终发布数据时，可以指定发布层数，但最多发布到气泡框层次为止，即发布出底层的“最精确”的抽象属性区 (Attribute Partition)。

¶Ô±ÈÓëÇ°ÎÄÂÛÊöµÄÆµÂÊŸØÕóµÄŽŠÀí¿ÉÒÔ¿Žµœ£¬·¢²Œ³ö4žöÊýŸÝœÚµãNode4£¬Node5£¬Nod

表 2(b) 总结了对原始数据集依照属性树划分的统计结果，即未经加噪保护处理的发布数据情况，其中的类计数为 Y 和 N 之和。显然，直接发布这个结果不是个安全的选择，还需要对最终的类计数按照拉普拉斯机制加噪音。如图 2 所示，它演示了 DiffGen 就图 1 的属性树规则对表 2(a) 数据集进行重匿名发布的过程，图 2 的上半部分 Node[1-7] 组成的树形结构称为 **DiffGen Tree**。

4. 如图 2 的 DiffGen Tree 中，首先形成根节点 Node1，此节点上各属性的属性值为所有属性树的根节点属性值，即所有属性的最大区间，这样它就把所有的训练数据归纳到根节点上。现在从根节点开始自上而下地逐层进行属性分裂操作，对数据集进行新的划分，

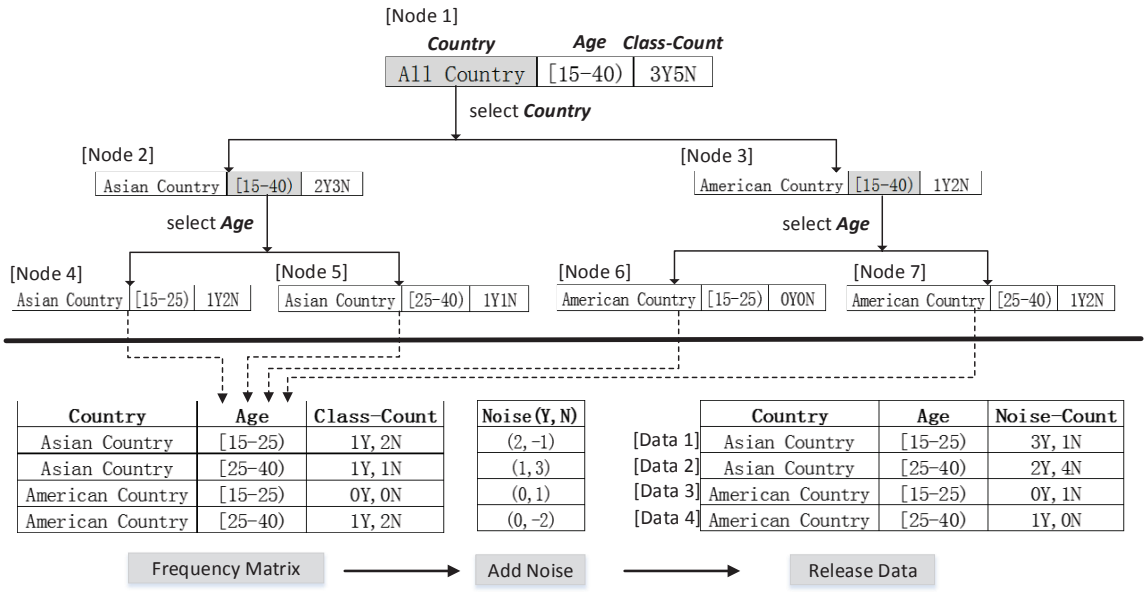


图 4: DiffGen Tree and Partitioning records

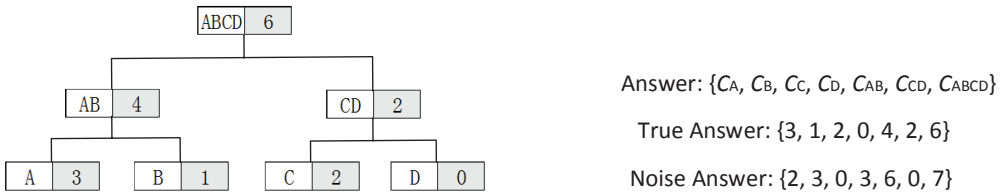
分裂过程用指数机制来选择待分裂属性。如第一次分裂时选取了属性“国家”，则依据其属性树的层级关系分裂为亚洲国家和美洲国家，并且数据集也按此分裂到下一层节点 Node2, Node3 中，以此类推。假设分裂层数足够多，那么最终所有数据集落在了叶节点上，得到了重匿名数据集。最后对它们进行加噪，噪音量为 $\mathbf{Lap}(2/\varepsilon)$ 并作负值处理，根据每行数据项的数目发布出新数据集给用户进行各种数据挖掘和分析工作。

对比与前文论述的频率矩阵加噪处理方式来看，发布出的 4 个数据节点 Node4, Node5, Node6, Node7，每个节点都对应着一条从根节点开始的路径，路径展开之后就是 Data[1-4] 的 4 条行项，组成了一个频率矩阵，如图 2。对这 4 个树节点的加噪过程其实就是对于 4 个行项组成的频率矩阵进行独立加噪 ($\mathbf{Lap}(2/\varepsilon)$) 的过程，最终得到数据集 {Data1，Data2, Data3，Data4}。可以看到，二者的处理过程是一样的。显然，DiffGen 在面对范围计数问题时也会遇到同样的噪音叠加问题。

2.3 Range-Count Queries with DiffGen

这一节详细揭示 DiffGen 在范围查询类应用中，属性维度、查询范围所带来的噪音叠加问题，产生了多余的噪音影响了发布数据的准确性。

5. 发布数据集 {Data1，Data2，Data3，Data4} 后，用户需要对这个数据集进行基于 ID3 算法 [17] 的决策树构建，这就需要计算属性的信息增益以决定分裂点，也就是统计每个属性在某个范围内的计数值。当计算第一个分裂点 (根节点) 时，需要统计整个数据集的 Y 和 N 分布，把 Data_{1-4} 的 Y、N 计数值累加起来计算熵值。而当属性值选定 Asian 后，需要计算剩余属性所包含的数据集的 Y/N 分布，即选择 Data1,Data2 的相应计数叠



(a) Tree Structure of String

(b) Query Answer

图 5: Example for Consistency

加。

通过例 5 可以看到，当计算属性 Asian 的 Y/N 分布时，选择 Data1 和 Data2 进行 2 次累加操作；计算全数据的 Y/N 分布时，对 Data_{1-4} 进行了 4 次累加操作。从 DiffGen Tree 结构可以看到，虽然发布的是叶节点数据，但其实每个叶节点数据项对应着一条从根节点开始的路径，这条路径展开就是图 2 的 Data[1-4] 的内容。因此，面对如例 5 的决策树算法应用的需求，对于 DiffGen 来说其实是找相关路径公共汇点，由于汇点节点没有进行加噪处理，因此把对应叶节点计数进行叠加得到结果。如计算属性 Asian 的 Y/N 分布，对应的汇点就是图 2 的 [Node 2]。因此，对 DiffGen Tree 的路径进行展开之后，这种叶节点进行加噪的方式与 Barak et al.'s[8] 采用的列联表加噪方式是一致的。

若这个数据集有更多的属性, 每个属性有着更大的维度呢？根据之前的讨论, 最多的叠加操作次数涉及到的数据集的维度空间大小, 由这个数据集的 domain size 决定, 即 $\prod_{i=1}^{d} |A_i|$, 计算全数据的 Y/N 分布就是如此。由之前章节的讨论，随着 domain size 的增加，列联表产生的问题在 DiffGen 中也会遇到，需要对其进行改进。

2.4 Consistency

一致性是数据集中属性之间的内在联系, 是用叠加来作为范围查询返回结果的前提。Dwork et al.'s[8] 的列联表方法, 之所以能够用单位长度的节点（叶节点）的叠加作为返回结果, 就是由于数据集数据关系间能保障严格的一致性，子节点的叠加能够覆盖父节点的属性范围。

一致性关系属于数据集的背景信息。DiffGen 模型在对数据集进行重匿名处理之后, 得到了新的数据组合关系, 对于每个节点来说, 总存在着对应的父节点或对应的子节点集。这些节点间的联系所暴露的背景信息虽然有可能泄露隐私信息，但是这种联系对噪音的分布也存在着约束，可以利用这种约束关系来减少叠加运算后产生的多余噪音，提高发布数据可用性。

6. 图 3 的数据库中有 3 个 A 字符，1 个 B 字符，2 个 C 字符，0 个 D 字符，面临图 3(b) 的查询需求，可能是单个字符的查询也有可能是字符串的查询，并返回加噪结果。字符或字符串 X 的计数值表示为 C_X，就字符 A,B,C,D 的统计情况总结如图 3(a) 的树状图，每个节点左框表示字符串（字符），右框表示计数，如字符串“AB”一共包含的字符计数

为每个字符的计数之和，即为 $C_{AB} = C_A + C_B = 3+1=4$。

考虑图 3(b) 的发布格式, 对所有节点进行加噪之后, 不仅发布叶节点数据, 内部非叶子节点数据也一同发布, 即根据路径需求发布汇点信息。如此的发布格式暴露出了树形结构存在的一致性约束关系：$C_{AB} = C_A + C_B$, $C_{CD} = C_C + C_D$, $C_{ABCD} = C_{AB} + C_{CD}$，但它可以优化范围计数时噪音的分布。例如, 对于加噪后父节点（字符串）的计数值的选取, 在返回字符串 AB 的加噪计数值时, 可以选择更优的方差表现组合, 即 $\min(\Theta(\tilde{C}_{AB}), \Theta(\tilde{C}_A + \tilde{C}_B))$；同理对于字符串 ABCD，选择的组合更多。因此，改变加噪方式，这种遍布每个内部节点的处理方式，在回答某些查询时可能会提供更好的噪音组合选择，提升返回结果质量。

现在有两种加噪方案。第一种方案采用上文所讨论的列联表式的加噪方式, 即 DiffGen 采用的方法，仅对叶节点进行加噪处理，发布格式为 $\{\tilde{C}_A, \tilde{C}_B, \tilde{C}_C, \tilde{C}_D\}$ 的数据集，需要进行后续的叠加计算才能满足范围计数需求。因此对于单个字符的查询，直接由发布数据集就可获取；对于字符串 AB 的计数查询，用户通过 $(\tilde{C}_A + \tilde{C}_B)$ 的一次运算得到结果；同理, 对于 ABCD 的计数查询，通过 3 次运算得到结果。显然，与前文的讨论一样存在着维度问题和粗糙的噪音叠加模式。在噪音敏感性方面，由于叶子节点间是相互独立的，因此敏感性为 1 [8]。

现在采用另一种处理方案, 给树上的每个节点都加噪, 发布所有的加噪节点给用户, 即发布格式为 $\{\tilde{C}_A, \tilde{C}_B, \tilde{C}_C, \tilde{C}_D, \tilde{C}_{AB}, \tilde{C}_{CD}, \tilde{C}_{ABCD}\}$。此时，无论是单个字符的查询还是字符串的查询，都不需要额外的运算处理（针对图 3(a) 中的树内节点而言，不讨论 C_{BC} 的这种情况），直接从发布数据中获取即可。

在噪音敏感性方面，显然需要重新定义。原生数据集建立在树形结构上，父子节点间的一致性关系使得发布出的数据之间不再是独立的关系了，此时敏感性等于树高 [14]。这里树高用 ℓ 表示，如图 3(a) 的树高 $\ell = 3$, 因此它的敏感性为 3。显然随着敏感性增加，噪音量级会变大，因此单个字符的返回值可能会更不准确，但对于范围查询来说能够最大化地减少噪音的叠加，也就是对于给定查询范围，尽量返回近根节点端的节点组合替代近叶节点端的节点组合。因为更靠近根节点更高层的树节点覆盖了更大的叶节点集合，能够尽可能地减少做叶节点范围覆盖而产生的叠加操作。其实，这相当于树中的最小顶点覆盖问题。

7. 查询图 3(a) 中的字符串 ABCD，可以返回多种组合方式：$\tilde{C}_{ABCD}$，$\tilde{C}_{AB} + \tilde{C}_{CD}$，$\tilde{C}_A + \tilde{C}_B + \tilde{C}_C + \tilde{C}_D$，为了选择最小的团集来覆盖节点集合 A,B,C,D，显然选择 $\{\tilde{C}_{ABCD}\}$。如果是字符串 BCD 呢？$\{\tilde{C}_B + \tilde{C}_{CD}\}$ 的选择是最优的。

（这里可以考虑写个小算法？如何做树中的最小顶点覆盖——先做一遍先序遍历记录父子节点间的关系，从待覆盖子节点集合的最左端节点开始，试探性检测其父节点能否覆盖到最右端待覆盖节点；若未达到，则继续向上找父节点检测，若超出范围，记下最顶端节点，更新左侧边界继续开始这个过程；若正好到达最右侧，完成退出。但有点太简单，，，）

但第二种方案在加噪后的数值表现来看产生了新的问题，同时也是新的机遇。由于噪

音的随机性，从每个树节点的含噪计数值来看，父子节点间的等号约束关系会被打破，原生数据的一致性关系在加噪之后无法维持。如 $C_{AB} = C_A + C_B$，但 $\tilde{C}_{AB} \neq \tilde{C}_A + \tilde{C}_B$。一致性是树形结构下原生数据的固有属性，加噪的影响使得它失去了这个特性。那么我们是否可以利用这个特性作为调整噪音分布的指导规则，通过这个先天的约束条件来调整整棵树的噪音分布情况，在调整的过程中裁剪或平衡一些多余的噪音使得它重新满足一致性，同时满足差分隐私呢？也就是优化整个数据集的加噪空间，让噪音的分布更加合理。不仅如此，方案二的发布格式会暴露会数据间的约束关系，能否像方案一一样，仅仅发布叶节点数据来保护这种约束关系，使得它依然安全可靠呢？两个问题的答案都是肯定的。

据 Hay[18] 的讨论，对于加噪后的数据集而言，一致性约束关系可以作为调整规则，在不影响隐私保障力度的前提下裁减掉一些不必要的噪音或调整各节点的噪音分布，即进行不违背 Differential Privacy 的后置处理以提升加噪数据的准确性。同时，由于处理之后的加噪数据重新满足了一致性约束关系，因此在发布格式上采取传统的发布方案即可，通过发布数据间的叠加来作为范围查询的应答结果，避免了内部节点关系的暴露问题。

3 DiffGen With Consistency

在这一节中，展示了利用一致性原理提升 DiffGen 算法效率的关键步骤。首先根据新的加噪方式重定义噪音敏感性，然后对树的内部节点进行拉普拉斯加噪，利用一致性原理调整噪音量，优化整棵树的噪音分布，最后发布叶节点数据集。

3.1 Sensitivity

在加噪之前，DiffGen 经过节点筛选后形成的具有一致性关系的树形结构称为 DiffGen Tree，如图 2。

1. *The Sensitivity of DiffGen Tree is ℓ , which is the height of the Tree.*

3.2 Add Laplace Noise

根据之前的噪音敏感性定义，接下来往树中的每个结点加噪。第 i 个树节点计数表示为 Tree(i)，m 为待加噪节点的维度，Lap 表示 Laplace 算子，加噪后变为

2. $\widetilde{Tree(i)} = Tree(i) + Lap(\ell/\varepsilon)^m,\ i \in DiffGen\ Tree$

如此，每个节点经过加噪后整棵树的一致性将被打破，最小单位长度的节点的发布在范围查询下无法起到很好的效果，因此我们需要做恢复一致性的处理。

3.3 Constrained Inference

由 Hay[18] 的调整算法为基础，对非一致性的树结构中的节点计数进行两次迭代处理，削减多余噪音的同时重新维护数据集的一致性。

在 Hay 的算法中，仅支持完全 K 叉树的情况，我们对此进行了改进使其可支持任意树结构。对于 DiffGen Tree 中的节点 i，其加噪后的计数表示为 noise(i)，做平衡削减后的

噪音量表示为 rest(i)；节点 i 的叶子数为 leafnum(i)，如叶节点的叶子数为 1，根节点叶子数为所有叶节点数目之和；节点 i 所覆盖的所有节点总数表示为 totalnum(i)，也就是由节点 i 开始的子树做遍历计算节点之和；节点 i 的父节点表示为 parent(i)；节点 i 的子节集表示 children(i)，若节点 j∈children(i)，表示节点 i 为节点 j 的父节点。

算法的迭代是逐层进行，第 j 层所有节点计算迭代完之后对 j-1 层开始迭代计算，以此类推。首先，从叶节点层开始到根节点，自下而上地遍历所有节点进行第一轮计算

if node i is leaf node, then rest(i) = noise(i)

else

$$rest(i) = \frac{leafnum(i) * rest(i) + (totalnum(i) - leafnum(i)) * \sum_{j \in children(i)} rest(j)}{totalnum(i)}$$

上面公式的等号表示赋值运算，具有从右向左的结核性，因此自下而上的运算过程右边的 rest(i) 之前已经得到，经过计算后更新了 rest(i) 的值。接着，进行依次自上而下的遍历，也是逐层迭代

if node i is root node, then rest(i) = rest(i)

else

$$rest(i) = rest(i) + \frac{rest(parent(i)) - \sum_{j \in children(i)} rest(j)}{\sum_{j \in children(i)} 1}$$

3.4 Release Dataset

由此过程, DiffGen Tree 获得了优化的加噪结构, 最后根据叶节点的加噪计数, 按照 Noman Mohammed et al.[7] 方法发布数据集即可。

Çëœ«ÖÂÐ»·ÅÔÚ×îº ó¡£

参考文献

[1] ÖÐ¹úÔË³ïÑ§»á. ÍøÖ·: http://www.orsc.org.cn

[2] ÕÂÏéÝ¥, Íõœš·œ, Áõ±ŠíÖ, ÁõµÂžÕ. *ÖÐ¹úÔË³ïÑ§»áµÚÆßœìÑ§Êõœ»Á÷»áÂÛÎÄŒ¯*. Global-Link Pulishing Company, 2004.

[3] ÎºÈšÁä. *ÊýŸÝ°üÂç·ÖÎö*, ¿ÆÑ§³ö°æÉç, 2004.

[4] Richard C. Larson. Operations Research: The Science of Better. *ÖÐ¹úÔË³ïÑ§»áµÚÆßœìÑ§Êõœ»Á÷»áÂÛÎÄŒ¯*, 1–11, Global-Link Pulishing Company, 2004.

[5] Ganta S R, Kasiviswanathan S P, Smith A. Composition attacks and auxiliary information in data privacy. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(SIGKDD).* New York,USA,2008:265-273

[6] Wong R C W,Fu A,Wang K,et al. Can the utility of anonymized data be used for privacy breaches. *ACM Transactions on Knowledge Discovery from Data* ,2011,5(3):16

[7] N. Mohammed, R. Chen, B. C. M. Fung, and P. S. Yu. Differentially private data release for data mining. In *SIGKDD* , 2011.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith, Calibrating noise to sensitivity in private data analysis, in *TCC* , 2006, pp. 265šC284.

[9] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Privbayes: Private data release via bayesian networks. In *SIGMOD*, pages 1423šC1434, 2014.

[10] G. Cormode, C. M. Procopiuc, D. Srivastava, and T. T. L. Tran. Differentially private publication of sparse data. *InICDT* , 2012.

[11] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111šC125, 2008.

[12] HAY, M., RASTOGI, V., MIKLAU, G., AND SUCIU, D. 2010. Boosting the accuracy of differentially-private queries through consistency. In Proceedings of *VLDB*.

[13] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar, Privacy, accuracy and consistency too: A holistic solution to contingency table release, in *PODS*, 2007.

[14] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, 2010.

[15] B. Ding, M. Winslett, J. Han, and Z. Li. Differentially private data cubes: optimizing noise sources and consistency. In *SIGMOD*, pages 217šC228, 2011.

[16] G. Cormode, M. Procopiuc, D. Srivastava, and T. Tran, ¡°Differentially private publication of sparse data,¡± in *ICDT*, 2012.

[17] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81šC106, 1986

[18] A. Charnes, W. W. Cooper and E. Rhodes. Measuring the efficiency of decision making units. *European Journal of Operaional Research*, 2(6), 429–444, 1978.

[19] º«ËÉ, ÎºÈšÁä. ×ÊÔŽÅäÖÃµÄ·Ç²ÎÊý DEA Ä£ÐÍ. *ÏµÍ³¹€³ÌÀíÂÛÓëÊµŒù*, 22, 59–64, 2002.