软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007152] [CSTR: 32375.14.jos.007152]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于本地化差分隐私的多表星形连接查询*

张啸剑[1], 曹小杰[1], 王 宁[2], 孟小峰[3]

[1](河南财经政法大学 计算机与信息工程学院, 河南 郑州 450046)

[2](中国海洋大学 信息科学与工程学部, 山东 青岛 266100)

[3](中国人民大学 信息学院, 北京 100872)

通信作者: 张啸剑, E-mail: 20100739@huel.edu.cn

摘 要: 基于本地化差分隐私多关系表示上的 Star-JOIN 查询已得到研究者广泛关注. 现有基于 OLH 机制与层次树结构的 Star-JOIN 查询算法存在根节点泄露隐私风险、τ-截断机制没有给出如何选择合适 τ 值等问题. 针对现有算法存在的不足, 提出一种有效且满足本地化差分隐私的 Star-JOIN 查询算法 LPRR-JOIN (longitudinal path random response for join). 该算法充分利用层次树的纵向路径结构与 GRR 机制, 设计一种纵向本地扰动算法 LPRR, 该算法以所有属性纵向路径上的节点组合作为扰动值域. 每个用户把自身元组映射到相应节点组合中, 再利用 GRR 机制对映射后的元组进行本地扰动. 为了避免事实表上存在的频率攻击, LPRR-JOIN 算法允许每个用户利用阈值 τ 本地截断自身元组个数, 大于 τ 条元组删减、小于 τ 条元组补充. 为了寻找合适的 τ 值, LPRR-JOIN 算法利用 τ-截断带来的偏差与扰动方差构造总体误差函数, 通过优化误差目标函数获得 τ 值; 其次结合用户分组策略获得 τ 值的总体分布, 再利用中位数获得合适的 τ 值. LPRR-JOIN 算法与现有算法在 3 种多关系数据集上进行比较, 实验结果表明其响应查询算法优于同类算法.

关键词: 本地化差分隐私; 多表星形连接查询; 层次结构; 纵向节点组合; 随机应答机制

中图法分类号: TP311

中文引用格式: 张啸剑, 曹小杰, 王宁, 孟小峰. 基于本地化差分隐私的多表星形连接查询. 软件学报. http://www.jos.org.cn/1000-9825/7152.htm

英文引用格式: Zhang XJ, Cao XJ, Wang N, Meng XF. Multi-table Star-JOIN Queries Based on Local Differential Privacy. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7152.htm

Multi-table Star-JOIN Queries Based on Local Differential Privacy

ZHANG Xiao-Jian[1], CAO Xiao-Jie[1], WANG Ning[2], MENG Xiao-Feng[3]

[1](College of Computer and Information Engineering, Henan University of Economics and Law, Zhengzhou 450046, China)

[2](Faculty of Information Science and Engineering, Ocean University of China, Qingdao 266100, China)

[3](School of Information, Renmin University of China, Beijing 100872, China)

Abstract: Star-JOIN queries based on local differential privacy (LDP) have attracted a lot of attention from researchers in recent years. Existing Star-JOIN queries based on the OLH mechanism and hierarchical tree structures face issues such as privacy leakage risks at the root node and the lack of guidance on selecting an appropriate τ value for the τ-truncation mechanism. To remedy the shortcomings of the existing algorithms, this study proposes an effective Star-JOIN query algorithm, longitudinal path random response for join (LPRR-JOIN), to satisfy the requirements of LDP. In the LPRR-JOIN algorithm, full advantage is taken of the longitudinal path structure of the hierarchical tree and the GRR mechanism to propose an algorithm called LPRR to perturb users' tuples. This algorithm utilizes the combinations of nodes along the longitudinal paths of all attributes as the perturbation domain. In the LPRR-JOIN algorithm, tuples are mapped by each user to corresponding node combinations, followed by local perturbation of the mapped tuples using the GRR mechanism.

* 基金项目: 国家自然科学基金 (62072156)
收稿时间: 2023-08-06; 修改时间: 2023-11-08; 采用时间: 2023-12-23; jos 在线出版时间: 2024-11-01

To guard against frequency attacks on the fact table, the algorithm permits users to locally truncate the count of their tuples based on a threshold τ, where tuples are deleted if their count exceeds τ and supplemented if it falls below τ. Two solutions are proposed within LPRR-JOIN to compute the optimal τ value. The first is to solve the optimization equation over bias caused by τ-truncation and perturbation variance due to LPRR. The second is to obtain the distribution of τ under the constraints of LDP and compute the median value from the distribution. The LPRR-JOIN algorithm employs an overall error function constructed from the bias and perturbation variance resulting from τ-truncation to derive an optimal τ value through the optimization of the error objective function. Additionally, by integrating a user grouping strategy, the algorithm ascertains the overall distribution of τ values and identifies a suitable τ value using the median. When compared with current algorithms across three diverse multi-relational datasets, experimental outcomes demonstrate the superiority of the LPRR-JOIN algorithm in query response performance.

Key words: local differential privacy; multi-table Star-JOIN queries; hierarchical structure; longitudinal-path node combination; random response mechanism

不同的服务平台可以凭借信息技术肆意地收集用户的个人数据. 例如某电子商务系统包括 3 种 APP 收集用户不同的数据信息, 其中 APP1 收集用户个人数据 (Profile), 形成 User 表; APP2 收集用户购买的商品数据信息, 形成 Product 表; APP3 收集用户购买商品产生的交易数据信息, 形成 Transaction 表. 基于这 3 类 APP 上的数据构建多维数据模型能够为数据仓库分析提供有力支撑. 星形模型是常见的多维数据模型. 该模型通常由一个事实表 (fact table) 与多个维表构成, 事实表与维表之间通过主外键约束实现多表之间的关联, 如图 1 所示. 而多关系表上的星形连接 (Star-JOIN) 查询是数据仓库常见的数据分析工具之一. 例如, 图 1 中的 User 表、Product 表和 Transaction 表通过 UID 和 PID 进行主外键连接, 而基于星形模型常见的 Star-JOIN 查询包括 Q_1 (SUM)、Q_2 (COUNT) 与 Q_3 (AVG) 等聚集操作.

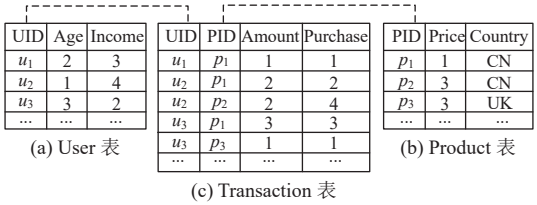


图 1　电子商务系统下的星形模型

Q_1: SELECT SUM(Amount) FROM Transaction JOIN User ON Transaction.UID = User.UID JOIN Product ON Product.PID = Transaction.PID WHERE Age∈[1,3] And Price∈[1,2].

Q_2: SELECT COUNT(*) FROM Transaction JOIN User ON Transaction.UID = User.UID JOIN Product ON Product.PID = Transaction.PID WHERE Age∈[1,2] And Price∈[1,3].

Q_3: SELECT AVG(Amount) FROM Transaction JOIN User ON Transaction.UID = User.UID JOIN Product ON Product.PID = Transaction.PID WHERE Age∈[1,2] And Price∈[2,4].

如果直接收集用户产生的多表数据并响应 Star-JOIN 查询, 与用户相关的敏感信息有可能会被泄露. 以图 1 响应 Q_2 查询为例, 可知查询结果为 COUNT(*)=3, 假设攻击者的先验知识为 u_2, 从查询结果中可推断出 u_1 只产生一条记录, u_1 的敏感信息被泄露. 此类情况下, 由于用户无法控制自己的原始数据, 进而造成敏感信息泄露. 本地化差分隐私 (local differential privacy, LDP)[1]将隐私数据的处理放在用户端, 确保每个用户采用本地编码与本地扰动机制来保护自身数据, 从而有效解决了数据收集过程中用户隐私泄露的问题. 目前, 基于 LDP 且支持多表间连接查询的算法主要包括 SC-JOIN[2]、HIO-JOIN[2]与 HIO-JOIN-RDC[2]等. SC-JOIN 算法结合隐私预算分割与 OLH[3]机制构建层次树 (如图 2 所示) 数据结构, 然而该方法仅能够响应一对一的两表关系查询, 无法支持一对多关系 (如图 1 中星形模型) 上的 Star-JOIN 查询. 由于该算法独立扰动表中的每个属性, 进而导致单表上的噪音过多, 连接查询结果可用性较低. HIO-JOIN 算法结合 HIO 机制[4]与两次用户分组策略对 SC-JOIN 算法进行了拓展, 使用 HIO 机制独立扰动每个元组, 并支持多表上的 Star-JOIN 查询. HIO-JOIN-RDC 算法对 HIO-JOIN 算法进行拓展,

其主要贡献是对多表查询谓词的分解方式进行了优化. 在响应一对多关系上的 Star-JOIN 查询时会存在频率攻击[2]问题. 为了避免出现该类问题, HIO-JOIN 与 HIO-JOIN-RDC 算法利用 τ-截断机制[2]对每个用户产生的元组条数进行限制, 大于 τ 条删减、小于 τ 条补充. 尽管 HIO-JOIN 与 HIO-JOIN-RDC 算法能够响应 Star-JOIN 查询, 仍然存在以下不足.

■ HIO-JOIN 与 HIO-JOIN-RDC 算法的本地扰动 HIO 机制会泄露根节点中的用户计数信息. HIO 机制核心方法包括 OLH 机制与用户分组策略. HIO 机制把每种节点组合作为一种哈希地址, 利用 OLH 机制对相应哈希地址进行本地扰动. 该机制要求层次树中每层节点组合不少于 $g=e^{\epsilon}+1$ 个, 然而层次树根节点组合只有一种情况. 例如, 假设需收集 45 个用户的 Amount 与 Income 数据, 其中 Amount 与 Income 属性对应的树结构如图 2 所示. 利用 HIO 机制的用户分组策略与层次树横向层次组合方式, 可以获得 9 种层次组合 (即: L^0L^0、L^0L^1、L^0L^2、L^1L^0、L^1L^1、L^1L^2、L^2L^0、L^2L^1、L^2L^2), 进而可以获得每一层次组合分到的用户数量为 5. 根节点所在的层次组合为 L^0L^0[1,4][1,4], 分配到该节点的用户数量为 5. 由于 OLH 机制把层次中所有节点组合作为哈希地址的扰动值域, 而根节点组合有且仅有 1 个, 进而 OLH 机制无法把根节点 L^0L^0[1,4][1,4] 组合扰动成其他哈希地址, 此种情况下 OLH 机制失效. 进而导致分配到 L^0L^0[1,4][1,4] 中 5 个用户的敏感属性泄露;

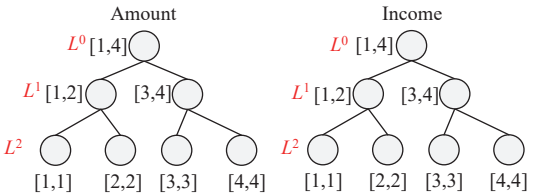


图 2 Amount 与 Income 属性对应的层次树

■ HIO-JOIN 与 HIO-JOIN-RDC 算法中的 τ-截断机制没有考虑截断操作引入的计数偏差. 仅利用本地扰动机制产生的均方误差度量算法的优劣. τ 的取值直接决定着偏差与扰动误差的大小. 过大的 τ 值会导致扰动误差过高、过小的 τ 值会导致偏差过大与数据损失严重. 如何选择合适的 τ 值是平衡偏差-均方误差的关键.

总而言之, 目前还没有一个有效且满足 LDP 的多表 Star-JOIN 查询算法能够弥补现有算法存在的不足. 基于 LDP 我们提出了一种有效算法来解决上述方法存在的问题. 主要贡献如下.

■ 为了避免 HIO-JOIN 与 HIO-JOIN-RDC 算法中根节点泄露隐私问题, 本文充分利用层次树的纵向结构设计一种有效响应 Star-JOIN 查询的算法 LPRR-JOIN. 其中 LPRR (longitudinal path random response) 算法结合两次用户分组策略与 GRR[5]机制实现纵向根-叶子路径上节点组合的本地扰动. LPRR 算法通过放弃叶子节点组合来防止叶子节点中元组信息泄露用户隐私.

■ 为了获得合适的 τ 值, LPRR-JOIN 算法分别采用目标函数优化方法与用户分组方法来求解 τ 值. 首先 LPRR-JOIN 结合 τ 值产生的偏差与本地扰动算法 LPRR 产生的误差构造总体误差作为目标函数, 通过优化总体误差获得最优 τ 值; 其次, LPRR-JOIN 算法把总体用户分成 βn 与 $(1-\beta)n$ 两组. βn 个用户以 GRR[5]的方式报告自己的元组个数, 然后以 βn 个报告值的中位数作为 τ 的估计值. $(1-\beta)n$ 个用户用来响应 Star-JOIN 查询.

■ 从 ε-uLDP 的角度理论分析了 LPRR-JOIN 算法的隐私性, 以及该算法响应 Star-JOIN 查询时的无偏性与误差. 通过实验分析, LPRR-JOIN 算法优于同类算法.

1 相关工作

目前, 基于差分隐私保护的多表连接查询得到了广泛关注. DJoin 算法[6]是中心化差分隐私 (central differential privacy, CDP) 下响应多表连接查询的早期代表. 该算法通过 MPC 技术[7]获得多方数据表, 并在多表连接操作结果上添加拉普拉斯噪音[8]. 然而该算法没有阐述如何控制多表连接查询的敏感度, 并且仅支持两张表的一对一连接. 为了限制多表等值连接查询的敏感度, FLEX 算法[9]基于 CDP 提出了一种局部敏感度[10]上界的弹性敏感度, 并利

用该敏感度控制多表连接查询的噪音量. 为了获得弹性敏感度, FLEX 需要借助平滑敏感度[10]来逼近本地敏感度. 然而, 平滑敏感度仅适应于固定大小的多表连接操作. 不同于 FLEX 算法的弹性敏感度, PrivateSQL 算法[11]利用 Lipschitz 拓展技术把多表连接查询转换成视图查询, 并利用截断操作控制视图查询的敏感度. 然而 PrivateSQL 算法仅支持多表上的 COUNT 查询. FLEX 与 PrivateSQL 算法计算弹性敏感度与视图敏感度的时间复杂度过高, 达到指数级. R_Lap 算法[12]结合多项式时间内获得的残差敏感度响应多表连接查询. 对比于弹性敏感度与平滑敏感度, 残差敏感度更逼近于局部敏感度. 然而该算法没有探讨外键约束情况下的多表连接查询. R2T 算法[13]结合力争上游策略与截断操作解决了外键约束单查询与自连接查询问题. 不同于 R2T 算法, PMSJA 算法[14]结合优化节点与二次规划技术响应多路查询. 尽管上述基于 CDP 的算法对多表连接查询进行系列研究, 然而这些算法无法直接应用于 LDP 下的 Star-JOIN 查询.

基于 LDP 与层次树结构的聚集查询同样得到研究者关注. TreeOUE 算法[15]利用 b-ary 层次树索引与编码用户数据, 并利用 OUE 机制[3]扰动每层的编码值. HI 算法[5]利用隐私预算分割与 OLH 机制[3]构建层次树, 并对单表上的聚集查询进行响应. 然而, 随着层次树高度的增加, HI 算法的聚集查询精度会降低. HIO 算法[4]结合用户分组策略与 OLH 机制响应单表上的聚集查询且精度高于 HI 算法. 文献 [2] 所提出的 AHIO 与 EHIO 算法都能支持敏感属性为聚集属性的查询, 弥补了 HI 与 HIO 算法的不足之处. 两种算法都采用了用户分组策略, 并对聚集属性进行随机凑整操作. 然而这两种算法的核心扰动机制 OLH 有可能泄露根节点隐私. LGRR 算法[16]为了避免根节点隐私泄露的风险, 利用层次树的纵向结构并结合 GRR 机制对用户的数据进行扰动. 然而, 上述这些方法仅支持单表上的聚集查询, 无法直接应用多表上的聚集查询. SC-JOIN[2]、HIO-JOIN[2]与 HIO-JOIN-RDC 算法[2]对多表上的聚集查询进行了探讨. SC-JOIN 算法利用隐私预算分割与 OLH 机制构建层次树, 然而该算法仅支持一对一的关系查询, 无法支持 Star-JOIN 查询. HIO-JOIN 算法结合 HIO 机制对 SC-JOIN 算法进行了拓展, 并支持多表上的 Star-JOIN 查询. HIO-JOIN-RDC 算法对 HIO-JOIN 算法进行拓展, 其主要贡献是对多表查询谓词的分解方式进行了优化. HIO-JOIN 与 HIO-JOIN-RDC 算法利用 τ-截断机制[2]避免 Star-JOIN 查询时存在的频率攻击[2]问题, 然而这两种方法没有阐述如何制定合适的 τ 值. 基于上述分析, 本文提出了有效的 Star-JOIN 查询算法 LPRR-JOIN, 该算法利用层次树的纵向结构与 GRR 机制扰动所有与单个用户关联的元组, 同时设计两种方法寻找 τ 值.

2 基础知识与问题描述

2.1 用户级别本地化差分隐私

LDP 保护技术允许用户在本地端编码与扰动后发送给数据收集者, 收集者聚集用户报告值响应查询. 本文关注星形模型下的 Star-JOIN 查询, 每个用户可能关联多条元组数据, 即是用户的数据分布在多个服务器上. 基于此, 结合用户级别的本地化差分隐私 (ε-uLDP)[17]保护所有与用户关联的数据. ε-uLDP 的形式化定义如下所示.

定义 1(ε-uLDP). 给定随机算法 R 及其定义域 $Dom(R)$ 和输出值域 $Range(R)$, 设 $T^u=(T_1^u,T_2^u,\ldots,T_K^u)$ 与 $T^{u'}=(T_1^{u'},T_2^{u'},\ldots,T_K^{u'})$ 表示用户 u 与 u'在 K 个服务器 $(s_1,s_2,\ldots,s_K)$ 上产生的元组集合. 若 R 符合不等式 (1), 则 R 满足 ε-uLDP.

$$\Pr[R(T^u)=y]\leqslant \exp(\varepsilon)\times \Pr\left[R\left(T^{u'}\right)=y\right] \tag{1}$$

其中, ε 为隐私预算. 该值越小表示算法 R 的保护程度越高.

ε-uLDP 通常具有序列组合性质.

性质 1[18]. 给定 K 个服务器上的元组集合和 K 个随机算法 $R_1,R_2,\ldots,R_K$, 且 R_i 满足 ε_i-uLDP, 则在 K 个服务器上的序列组合满足 ε-uLDP, 且 $\varepsilon=\sum\varepsilon_i$.

2.2 随机应答机制

LDP 下最常用的扰动机制为文献 [19] 所提出的随机应答机制, 用于收集敏感问题的统计信息, 报告真实值的概率为 p, 报告相反值的概率为 q. GRR 机制[5]将 RR 机制的二类别扩展到多类别, 公式 (2) 为 GRR 机制的扰动概率.

$$\forall_{y\in D_i}\Pr[\mathrm{GRR}(t)=y]=\begin{cases}p=\dfrac{\mathrm{e}^{\varepsilon}}{\mathrm{e}^{\varepsilon}+m-1}, & y=t\\ q=\dfrac{1-p}{m-1}, & y\neq t\end{cases} \tag{2}$$

其中, ε 为隐私预算, m 表示 t 的值域大小. 由 $p/q\leqslant \mathrm{e}^{\varepsilon}$ 可知 GRR 机制满足 ε-uLDP.

2.3 层次树结构

假设属性 a 有 m 个不同的取值, 按照顺序分别表示为 $v_1, v_2, \ldots, v_m$. 构建一个扇出为 b 的层次树, 层次树中的每个节点代表一个区间. 除叶子节点之外每个节点有 b 个孩子, 且每个区间的大小相同. 属性 a 的层次树树高为 $h=\log_b m$, 每层的区间个数为 b^j $(0\leqslant j\leqslant h)$. $L^0=\{[v_1, v_m]\}$为根节点所在层次, $[v_1, v_m]$ 为根节点对应区间, 根节点向下迭代划分成 b 个相同大小的子区间, 直到叶子节点层, 属性 a 的叶子节点层以及每个节点对应的区间为 $L^h=\{[v_1, v_1], [v_2, v_2],\ldots, [v_m, v_m]\}$.

用户将每个属性值映射到层次树上经过扰动后发送给数据收集者, 收集者重构每个属性对应的层次树结构. 收集者将给定的 Star-JOIN 查询 $Q(Agg(r\cdot a), C)$ 进行查询重写, 将谓词 C 中的条件范围按照每个属性的层次树结构对其进行区间划分. 若属性 $a_i\in[l, r]$ 为 C 中的条件谓词, $[l, r]$ 最多能分解成 $2(b-1)\log_b m$ 个子区间. 查询重写即将查询 Q 分解成层次树中不同的子区间查询, Star-JOIN 查询 $Q(Agg(r\cdot a), C)$ 可转化为子区间的计数查询. 因此, 收集者对查询 Q 进行重写后, 根据重构的层次树结构响应 Star-JOIN 查询 $Q(Agg(r\cdot a), C)$.

2.4 频率攻击与 τ-截断机制

根据文献 [2] 可知, 用户分布在多个服务器上的敏感数据经过 uLDP 保护后, 仍然会受到恶意频率攻击. 攻击者依据用户 ID 观察每个用户的交易元组个数 (即频率), 利用先验知识可以推断出目标用户的敏感信息 (如工资、年龄、购物情况等). 结合图 1 中的 3 张关系表来说明频率攻击. 图 3 中的两个用户 u_1 与 u_2 产生了 3 类数据: 个人数据 (UID, Age, Income)、购买商品数据 (PID, Price, Country)、交易数据 (UID, PID, Amount, Quantity). 这 3 类数据经过 uLDP 保护后分别被 s_1, s_2 和 s_3 服务器收集, 形成了 User 表、Product 表以及 Transaction 表. 假设攻击者的先验知识是“用户的收入 (Income) 与 Transaction 表中元组频率成 100 倍关系”. 凭借 Transaction 表中元组频率 (如 u_1 交易产生 1 条元组, u_2 产生 2 条元组) 和先验知识, 攻击者可以推理出用户 u_1 与 u_2 的收入 (Income) 分别是 100×1=100, 100×2=200.

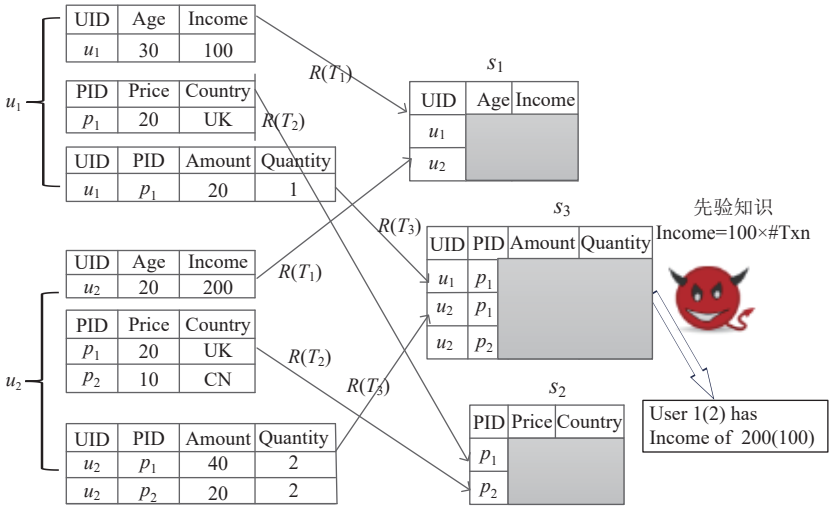


图 3 频率攻击例子

文献 [2] 提出了 τ-截断机制防止 Star-JOIN 查询下的频率攻击. 由图 3 可知, 恶意攻击者主要依赖 s_3 服务器上的 Transaction 表进行频率攻击, 主要原因是 Transaction 表中的元组经过 uLDP 保护后, 其频率仍然具有差异性.

τ-截断机制主要思想消除频率的差异性, 即是对于 Transaction 表中任意两个用户产生的元组条数都应该是 τ 条. 假设图 1 中 User 表包含 n 个用户、Product 表包含 n 个商品. 为了避免 Transaction 表中的频率攻击, 每个用户需要发送 $n\tau$ 条元组到 Transaction 表中, 最终 Transaction 表包含 $n^2\tau$ 条交易元组. 例如, 图 4 中用户 u_i 购买 p_1 商品产生了 1 条交易元组. 假设 Product 表中共包含 2 类商品, τ=1. 经过 τ-截断机制处理后, 在 Transaction 表中共产生 2 条元组, 如图 4 所示. 其中截断权重属性 r=1/2.

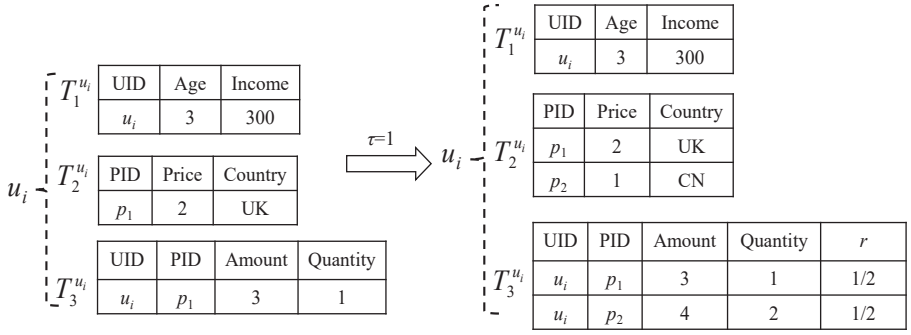


图 4 τ-截断机制例子

本文关注 K 个服务器 $(s_1, s_2, \ldots, s_K)$ 同时收集 n 个用户数据的情况, 经过 uLDP 保护后形成 $(T_1, T_2, \ldots, T_K)$ 数据表, 任意 $T_i = (T_i^{u_1}, T_i^{u_2}, \ldots, T_i^{u_n})$ $(1 \leqslant i \leqslant K)$ 且 $T_i^{u_j}$ $(1 \leqslant j \leqslant n)$ 表示用户 u_j 在第 i 个服务器上产生的元组. 假设 s_1 上的 T_1 表为 n 个用户的个人数据, s_K 上的 T_K 表为事实表 (或者 Transaction 表). 频率攻击通常发生在事实表 T_K 上. 如何把 τ-截断机制拓展到该表上是个大的挑战. 假设用户 u_i 在 T_K 中真实产生的元组条数为 $\left|T_K^{u_i}\right|$>0. 为了利用 τ-截断机制隐藏真实 $\left|T_K^{u_i}\right|$ 值, u_i 需要在事实表中添加 $(n^{K-2}\tau - \left|T_K^{u_i}\right|)$ 条元组才能达到 $n^{K-2}\tau$. 每个用户在 T_K 中产生 $n^{K-2}\tau$ 个元组, 则 n 个用户在 T_K 表中共有 $n^{K-1}\tau$ 条元组. 为了使得最终的 Star-JOIN 查询的无偏性, 需要为 T_K 表中每条元组添加一个截断权重 $r = \left|T_K^{u_i}\right| / n^{K-2}\tau$, 且 $r \in [r_{\min}, r_{\max}]$. 当 $\left|T_K^{u_i}\right| = 0$ 时, 用户 u_i 需要在 T_K 表的值域 D 中随机抽样 $n^{K-2}\tau$ 条元组来形成 $T_K^{u_i}$. τ-截断机制思想如公式 (3) 所示.

$$Trunc(\tau, T_K^{u_i}) = \begin{cases} \left\langle t_j \leftarrow D, r = \dfrac{\left|T_K^{u_i}\right|}{n^{K-2}\tau} \right\rangle_{j=1}^{n^{K-2}\tau}, & \text{if } \left|T_K^{u_i}\right| > 0 \\ \langle t_j \leftarrow D, r = 0 \rangle_{j=1}^{n^{K-2}\tau}, & \text{if } \left|T_K^{u_i}\right| = 0 \end{cases} \tag{3}$$

因此, 本文要解决的问题是考虑频率攻击与 τ-截断的前提下如何设计满足 uLDP 的 Star-JOIN 查询方法, 且尽可能获得较高精度的查询结果.

3 基于 uLDP 的 Star-JOIN 查询算法

在设计满足 uLDP 的 Star-JOIN 连接查询算法时需要考虑 3 条原则: 1) 在利用层次树结构设计本地扰动方法时如何避免根节点泄露隐私计数问题; 2) 在利用 τ-截断机制避免频率攻击时, 如何寻找合适的 τ 值来均衡噪音误差与截断误差; 3) 设计的 Star-JOIN 查询算法尽可能返回较高精度的查询结果. 针对上述 3 种原则, 本文设计了 LPRR-JOIN 算法来响应 Star-JOIN 查询. 其中 LPRR 是利用层次树的纵向结构与 GRR 机制设计的能够避免根节点泄露隐私的本地扰动算法. LPRR-JOIN 算法分别利用优化函数寻找最优 τ 值, 以及利用用户分组策略寻找可行的 τ 值. 首先介绍 LPRR-JOIN 算法, 细节如算法 1 所示.

算法 1. LPRR-JOIN 算法.

输入: n 个用户在 $(s_1, s_2, \ldots, s_K)$ 服务器上产生的数据表 $(T_1, T_2, \ldots, T_K)$, 数据表中每个属性的值域都为 m, 层次树的扇出为 b, 隐私预算 ε, Star-JOIN 查询 $Q(Agg(r \cdot a), C)$, 其中谓词 $C=\{a_1 \in [l_1, r_1] \wedge \ldots \wedge a_{Kd} \in [l_{Kd}, r_{Kd}]\}$;

输出: 满足 ε-uLDP 的 Star-JOIN 查询结果 $P(Q)$.

1. 收集者根据 $(T_1, T_2,\ldots, T_K)$ 表中每个敏感属性的扇出 b 与值域大小 m 构建 $K\times d$ 棵层次树;
2. 收集者把 $K\times d$ 棵层次树副本共享给 n 个用户;
3. 收集者计算合适的 $\tau\leftarrow$*Tao-Search*(τ_1, τ_2); //τ_1 与 τ_2 分别表示理论最优截断值与估计截断值
4. $M\leftarrow\varnothing$; //M 用来存储扰动后的元组, 以及元组所在节点组合对应的层次索引
5. **for each** user u_i **do**
6. 用户 u_i 利用公式 (3) 对 $T_K^{u_i}$ 的元组数进行截断后生成 $n^{K-2}\tau$ 条元组, u_i 共拥有 $1+(K-2)n+n^{K-2}\tau$ 条元组.
7. **for each** tuple t_i of $1+(K-2)n+n^{K-2}\tau$ tuples **do**
8. **if** $t_i\in T_1$ **or** $t_i\in T_K$ **then**
9. 用户 u_i 报告 $((j_1, j_2,\ldots, j_d), \mathrm{LPRR}(t_i))$ 与 $((j_{(K-1)d+1}, j_{(K-1)d+2},\ldots, j_{Kd}), \mathrm{LPRR}(t_i))$ 给服务器 s_1 和 s_K;
10. $M_{j_1,j_2,\ldots,j_d}\leftarrow M_{j_1,j_2,\ldots,j_d}\cup\{\mathrm{LPRR}(t_i)\}$;
11. $M_{j_{(K-1)d+1},j_{(K-1)d+2},\ldots,j_{Kd}}\leftarrow M_{j_{(K-1)d+1},j_{(K-1)d+2},\ldots,j_{Kd}}\cup\{\mathrm{LPRR}(t_i)\}$;
12. **else if** $t_i\in T_l$ $(2\leqslant l\leqslant K-1)$ **then**
13. 用户 u_i 报告 $((j_{(l-1)d+1}, j_{(l-1)d+2},\ldots, j_{ld}), \mathrm{LPRR}(t_i))$ 给服务器 s_l;
14. $M_{j_{(l-1)d+1},j_{(l-1)d+2},\ldots,j_{ld}}\leftarrow M_{j_{(l-1)d+1},j_{(l-1)d+2},\ldots,j_{ld}}\cup\{\mathrm{LPRR}(t_i)\}$;
15. **for each** $a_i\in C$ **do**
16. K 个收集者 $(s_1, s_2,\ldots, s_K)$ 利用属性 a_i 的层次树对 a_i 的谓词区间 $[l_i, r_i]$ 分解成 k_i 节点; //查询重写 Q
17. **for each** combination $g_i\in\{1, 2,\ldots, k_1\}\times\ldots\times\{1, 2,\ldots, k_{Kd}\}$ **do**
18. K 个收集者 $(s_1, s_2,\ldots, s_K)$ 利用 Star-JOIN 操作计算 $\tilde{c}(g_i)$;
19. **return** $P(Q)=\sum_{1\leqslant i_1\leqslant k_1,\ldots,1\leqslant i_{Kd}\leqslant k_{Kd}}\tilde{c}(g_i)$.

收集者借助于 $(T_1, T_2,\ldots, T_K)$ 每个敏感属性的扇出 b 和值域 m 构建 $K\times d$ 棵层次树, 并把 $K\times d$ 棵树共享给每个用户 (步骤 1, 2). 收集者采用两种方法计算合适裁剪阈值 τ 来均衡噪音误差与截断误差. 一种为理论最优截断值, 通过优化截断偏差与扰动误差之和获得最优裁剪阈值 τ_1; 一种为估计截断值, 通过 βn 个用户报告元组个数的中位数作为裁剪阈值的估计值 τ_2, 并广播给每个用户 (步骤 3). 对于任意用户 u_i 首先结合 τ 值对 $T_K^{u_i}$ 的元组数进行截断, 并在 $(s_1, s_2,\ldots, s_K)$ 服务器上产生 $1+(K-2)n+n^{K-2}\tau$ 条元组 (步骤 6). 对于 $1+(K-2)n+n^{K-2}\tau$ 中每条元组 t_i, 若 t_i 在服务器 s_1 或者在服务器 s_K 中, 则用户 u_i 需要向 s_1 发送 1 条或者向 s_K 发送 $n^{K-2}\tau$ 条扰动元组; 若 t_i 在 s_l $(2\leqslant l\leqslant K-1)$ 中, 用户 u_i 需要向 s_l 发送 n 条扰动元组 (步骤 7–14). 结合 $K\times d$ 棵树, K 个收集者需要对 Star-JOIN 查询 $Q(Agg(a), C)$ 中的谓词属性集合 C 进行查询重写. 对于 C 中的任意属性 a_i, 按照对应层次树对其进行分割 (步骤 15, 16). 对所有分割节点进行排列组合, 而对于所有组合计算其噪音计数, 最后响应 Q 查询 (步骤 17–19). 由算法 1 可知, LPRR(·) 与 *Tao-Search*(·) 是 LPRR-JOIN 算法的核心步骤. 基于 HIO 机制的本地扰动算法通常会泄露层次树根节点组合中用户计数隐私, 因此本文利用层次树的纵向路径组合设计本地扰动算法 LPRR(·), 通过纵向扰动叶子-根节点组合防止根节点隐私泄露, 并通过摒弃叶子节点层防止叶子节点泄露用户隐私, 其实现细节如算法 2 所示.

算法 2. LPRR 算法.

输入: 用户 u_i 分布在 $s_1, s_2,\ldots, s_K$ 上的 $1+(K-2)n+n^{K-2}\tau$ 条元组, 隐私预算 ε;
输出: $1+(K-2)n+n^{K-2}\tau$ 条元组对应节点扰动结果与层次索引.

1. **for each** tuple t_i of T_K **do** //用户 u_i 对自己在 T_K 中产生的 $n^{K-2}\tau$ 元组进行变换与扰动
2. 用户 u_i 从 T_K 的 d 个属性中随机抽取一个属性 a;
3. **if** a 是 $Q(Agg(r\cdot a), C)$ 查询中的聚集属性 **then**

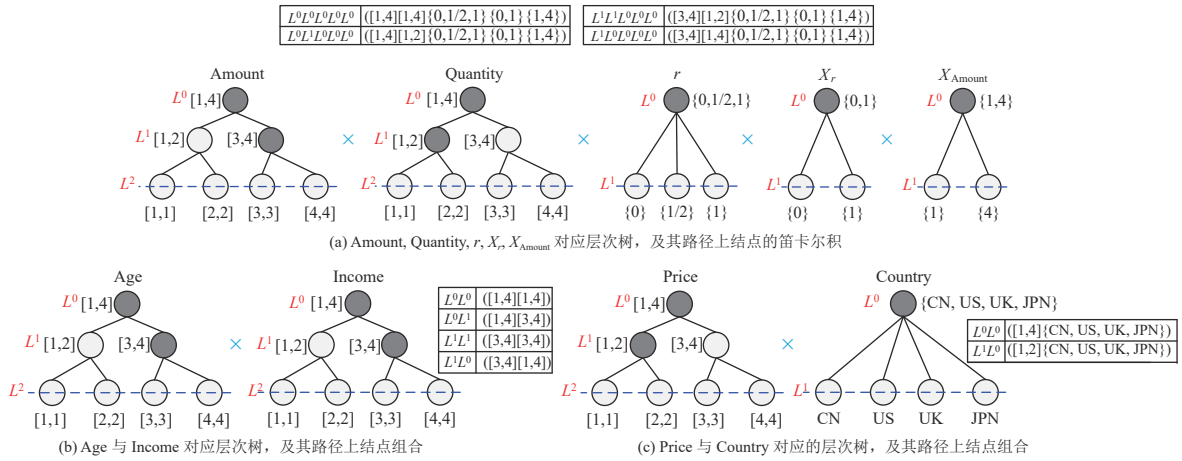
4. 用户 u_i 对 $t_i[a]$ 与 $t_i[r]$ 进行凑整, 形如: $t_i[a]=\begin{cases} a_{\max}, & w\cdot p\ \dfrac{t_i[a]-a_{\min}}{a_{\max}-a_{\min}} \\ a_{\min}, & w\cdot p\ \dfrac{a_{\max}-t_i[a]}{a_{\max}-a_{\min}} \end{cases}$, $t_i[r]=\begin{cases} r_{\max}, & w\cdot p\ \dfrac{t_i[r]-r_{min}}{r_{\max}-r_{\min}} \\ r_{\min}, & w\cdot p\ \dfrac{r_{\max}-t_i[r]}{r_{\max}-r_{\min}} \end{cases}$;
//$a_{\max}$ 与 $a_{\min}$ 为 a 的最大值与最小值; $r_{\max}$ 与 $r_{\min}$ 为 r 的最大值与最小值
5. 用户 u_i 为元组 t_i 分别添加凑整属性 X_a 与 X_r; //X_a 与 X_r 分别为 a 与 r 的凑整值;
6. 用户 u_i 把元组 t_i 的所有属性映射到 d+3 条无叶节点的纵向路径上;
7. 用户 u_i 计算 d+3 条纵向路径上所有节点的组合, 形成的集合记为 G, 且 $|G|=(\log_b m)^d$;
8. 用户 u_i 随机选择节点组合 g_i ($g_i\in G$), g_i 所在层次为 $(j_{(K-1)d+1}, j_{(K-1)d+2},\ldots, j_{Kd+3})\in\{\{0, 1,\ldots, \log_b m-1\}^d\cup\{0\}^3\}$;
9. 用户 u_i 随机本地扰动 g_i, 形如: $\forall_{y_i\in G}\Pr[\mathrm{LPRR}(g_i)=y_i]=\begin{cases} \dfrac{e^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}}{e^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+|G|-1}, & y_i=g_i \\ \dfrac{1}{e^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+|G|-1}, & y_i\neq g_i \end{cases}$;
10. **for each** tuple t_i of (1+(K–2)n) tuples **do** //u_i 对自己在 $T_1,\ldots,T_{K-1}$ 中产生的元组进行变换与扰动
11. 用户 u_i 把元组 t_i 的所有属性映射到 d 条无叶节点的纵向路径上;
12. 用户 u_i 计算 d 条纵向路径上所有节点的组合, 形成的集合记为 G, 且 $|G|=(\log_b m)^d$;
13. 用户 u_i 随机选择一个组合 g_i ($g_i\in G$), g_i 所在层次为 $(j_{(l-1)d+1}, j_{(l-1)d+1},\ldots, j_{ld})\in\{0, 1,\ldots, \log_b m-1\}^d$;
14. 用户 u_i 随机本地扰动 g_i, 形如: $\forall_{y_i\in G}\Pr[\mathrm{LPRR}(g_i)=y_i]=\begin{cases} \dfrac{e^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}}{e^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+|G|-1}, & y_i=g_i \\ \dfrac{1}{e^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+|G|-1}, & y_i\neq g_i \end{cases}$;
15. **return** $\{\{((j_1, j_2,\ldots, j_d), y_i)\}, \{((j_{(l-1)d+1}, j_{(l-1)d+1},\ldots, j_{ld}), y_i),\ldots, ((j_{(K-2)d+1}, j_{(K-2)d+2},\ldots, j_{(K-2)d}), y_i)\}, \{((j_{(K-1)d+1}, j_{(K-1)d+2},\ldots, j_{Kd+3}), y_i),\ldots, ((j_{(K-1)d+1}, j_{(K-1)d+2},\ldots, j_{Kd+3}), y_i)\}\}$; //1+($K$–2)$n$+$n^{K-2}\tau$ 条元组对应节点扰动结果与层次索引

每个用户 u_i 如何把与自己关联的 1+(K–2)n+$n^{K-2}\tau$ 条元组本地扰动后, 发送给 K 个服务器 $s_1, s_2,\ldots, s_K$, 并使扰动结果满足 uLDP 差分隐私是 LPRR 算法的关键. K 个服务器通过收集 n 个用户的元组数据来响应 Star-JOIN 查询. 服务器 s_K 上存储的是事实表 T_K, 该表中的元组度量属性通常作为 Star-JOIN 查询聚集函数中的聚集属性. 因此, 用户 u_i 对自己在 T_K 中产生的 $n^{K-2}\tau$ 元组进行变换与扰动. u_i 首先从 T_K 的 d 个属性中随机抽取一个属性 a(步骤 2), 若 a 是函数 $Agg(r\cdot a)$ 中的聚集属性, 则需要对其和 r 进行随机凑整并形成新的属性 X_a 与 X_r (步骤 3–5). 为了避免根节点与叶子节点泄露用户隐私, 用户 u_i 再分别对 T_K 与 (T_1, T_2,…, T_{K-1}) 中的所有元组数据进行本地纵向扰动, 即是分别把自身元组映射到 d+3 条或者 d 条无叶节点的纵向路径上, 并形成所有路径节点组合. 然后利用 GRR 机制对自身元组所在的节点组合进行扰动 (步骤 6–14). 以图 4 截断元组为例来说明 LPRR-JOIN 与 LPRR 算法原理. 给定 Star-JOIN 查询 Q_1: SELECT SUM(Amount) FROM Transaction JOIN User ON Transaction.UID = User.UID JOIN Product ON Product.PID = Transaction.PID WHERE Age∈[1,3] And Price∈[1,2], 其聚集属性为 Amount. 图 4 中用户 u_i 对 T_3 截断后拥有 5 条元组 t_1, t_2, t_3, t_4, t_5.

图 5 为属性 Age、Income、Price、Country、Quantity, 聚集属性 Amount 和 r 与凑整属性 X_r 和 X_{Amount} 对应的层次树及其节点组合. 设 Age、Income、Price、Amount 与 Quantity 的值域均为 [1,4], 截断权重 r 的取值为{0, 1/2, 1}. 利用 LPRR 算法步骤 4 对元组 t_4 与 t_5 的 Amount 与权重 r 进行凑整. Amount∈[1,4], t_4[Amount]=3 与 t_5[Amount]=4 被凑整为 4, 则在 t_4 与 t_5 中添加属性列 X_{Amount} 且 $t_4[X_{\mathrm{Amount}}]=4$, $t_5[X_{\mathrm{Amount}}]=4$, $X_{\mathrm{Amount}}\in\{1, 4\}$. $r\in\{0, 1/2, 1\}$, $t_4[r]=1/2$ 与 $t_5[r]=1/2$ 分别被凑整为 1 或者 0. t_4 与 t_5 中添加属性列 X_r 且 $t_4[X_r]=1$, $t_5[X_r]=0$, $X_r\in\{0, 1\}$. t_4 与 t_5 所对应的 Amount、Quantity、r、X_{Amount} 与 X_r 对应的层次树如图 5(a) 所示. 元组 t_4 与 t_5 分别被映射到层次树的纵向路径上, 如图 5(a) 中层次树灰色路径所示. 例如 $t_4[r]=1/2$, $t_5[r]=1/2$, t_4[Quantity]=1, t_5[Quantity]=2 分别被映射到{0, 1/2, 1}、[1,4]→[1,2] 所在路径上. 元组 t_1、t_2 与 t_3 的属性 Age、Income、Price 与 Country 对应的层次树如图 5(b) 与图 5(c) 所示. 同样利用 LPRR 算法把 t_1、t_2 与 t_3 映射到图 5(b) 与图 5(c) 中层次树灰色路径上. 如

t_1[Age]=3 与 t_1[Income]=3 被映射到 [1,4]→[3,4] 路径上, t_2[Country]=UK 与 t_3[Country]=CN 被映射到{CN, US, UK, IN, JPN}所在的路径上.

用户 u_i 的所有元组映射到对应的层次树之后, 需要根据层次树纵向路径上的节点组合对其本地扰动. 例如, 元组 t_1 中的属性 Age 与 Income 所对应的层次树结构如图 5(b) 所示, 其中灰色路径上节点的笛卡尔积组合共有 $|G|$=4 种. 用户 u_i 随机选择其中一种, 如 L^1L^0([3,4][1,4]). 根据 LPRR 算法步骤 14 可知, L^1L^0([3,4][1,4]) 依概率 $e^{\varepsilon/5}/e^{\varepsilon/5}$+3 扰动成自身, 依概率 $1/e^{\varepsilon/5}$+3 扰动成除自身之外的其他 3 种组合. 为了响应 Q_1 查询, 需要结合 LPRR-JOIN 算法的步骤 15–18 对其查询谓词进行重写. Q_1 查询的 WHERE 子句中包含 Age∈[1,3] 和 Price∈[1,2]. 结合 Age 与 Price 的层次树对区间 [1,3] 与 [1,2] 进行分割, 其中 [1,3] 被分割为 [1,2] 与 [3,3,1,2] 被分割为 [1,2]. 利用 Age 与 Price 的分割结果去过滤 t_1, t_2, t_3, t_4, t_5 中哪些元组符合谓词条件, 并求和 Amount 属性值.



(a) Amount, Quantity, r, X_r, X_{Amount} 对应层次树，及其路径上结点的笛卡尔积

(b) Age 与 Income 对应层次树，及其路径上结点组合

(c) Price 与 Country 对应的层次树，及其路径上结点组合

图 5　层次树及其节点组合

定理 1. LPRR 算法满足 ε-uLDP.

证明: 根据算法 1 与算法 2 可知, 用户 u_i 利用 τ-截断机制对 $T_K^{u_i}$ 截断后, $(s_1, s_2, \ldots, s_K)$ 服务器共收集该用户 $1+(K-2)n+n^{K-2}\tau$ 条元组. 设每个元组的维度均为 d, 每个维度的值域为 m, 树扇出为 b, 截断权重 r 的值域为 m_r. 两个凑整属性的值域大小为 2. LPRR 在扰动节点组合时摈弃了叶子节点, 这样致使截断属性和凑整属性的层次树只剩根节点. 因此 $1+(K-2)n+n^{K-2}\tau$ 条元组中的任意一条元组 t_i, 其所有层次树纵向路径节点组合仅有 $|G|=(\log_b m)^d$ 种. 利用公式 (2) 中的 GRR 机制对 G 中任何一种组合 g_i 进行本地扰动, 扰动概率如公式 (4) 所示.

$$\forall_{y\in G}\Pr[\mathrm{LPRR}(g_i)=y]=\begin{cases} p=\dfrac{e^{\varepsilon/(1+(K-2)n+n^{K-2}\tau)}}{e^{\varepsilon/(1+(K-2)n+n^{K-2}\tau)}+|G|-1}, & y=g_i \\ q=\dfrac{1}{e^{\varepsilon/(1+(K-2)n+n^{K-2}\tau)}+|G|-1}, & y\neq g_i \end{cases} \tag{4}$$

由 $p/q\leqslant\exp(\varepsilon/(1+(K-2)n+n^{K-2}\tau))$ 与性质 1 可知, LPRR 算法扰动 u_i 产生的 $1+(K-2)n+n^{K-2}\tau$ 条元组满足 ε-uLDP.

定理 2. 与每个用户关联的 $1+(K-2)n+n^{K-2}\tau$ 条元组经过 LPRR 扰动后, 被映射到 $(T_1, T_2, \ldots, T_K)$ 所有属性层次树纵向路径节点组合中. 任意表 $T_k\in(T_1,\ldots,T_K)$ 的属性层次树纵向路径节点组合有 $|G|=(\log_b m)^d$ 种. 给定任意 $g_i\in G$, $c'(g_i)$ 与 $\tilde{c}(g_i)$ 分别表示组合 g_i 中截断后真实的元组计数与估计计数, 则 $\mathrm{E}[\tilde{c}(g_i)]=c'(g_i)$ 成立.

证明: 令 $c''(g_i)$ 表示组合 g_i 中截断后元组计数的观测值, n_k 为 G 中所有节点组合中的元组个数. 观测值 $c''(g_i)$ 可以表示为 $c''(g_i)=c'(g_i)p+(n_k-c'(g_i))q$.

由极大似然定理可知 $c'(g_i)$ 的估计值 $\tilde{c}(g_i)$ 可以表示为: $\tilde{c}(g_i)=c''(g_i)-n_kq/p-q$. 则 $\mathrm{E}[\tilde{c}(g_i)]=\mathrm{E}[c''(g_i)-n_kq/p-q]$.

把公式 (4) 中的 p, q 与 $c''(g_i)=c'(g_i)p+(n_k-c'(g_i))q$ 带入 $\mathrm{E}[\tilde{c}(g_i)]=\mathrm{E}[c''(g_i)-n_kq/p-q]$, 可知 $\mathrm{E}[\tilde{c}(g_i)]=c'(g_i)$ 成立.

定理 3. 任意表 $T_k\in(T_1,\ldots,T_K)$ 的属性层次树纵向路径节点组合集合为 G, 给定任意节点组合 $g_i\in G$, g_i 中元组估计计数 $\tilde{c}(g_i)$ 由 LPRR 算法引起的方差为:

$$\mathrm{Var}[\tilde{c}(g_i)]=n_k\times\frac{\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}{\left(\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}-1\right)^2} \tag{5}$$

其中, n_k 为 G 中所有节点组合的元组个数.

证明: 由定理 2 中 $\tilde{c}(g_i)=c''(g_i)-n_kq/p-q$ 可知, $\mathrm{Var}[\tilde{c}(g_i)]=\mathrm{Var}[c''('g_i)-n_kq/p-q]$.

则

$$\begin{aligned}\mathrm{Var}[\tilde{c}(g_i)]&=\mathrm{Var}\left[\frac{c''(g_i)-n_kq}{p-q}\right]=\frac{\mathrm{Var}[c''(g_i)]}{(p-q)^2}\\&=\frac{c'(g_i)p(1-p)+(n_k-c'(g_i))q(1-q)}{(p-q)^2}=\frac{n_kq(1-q)}{(p-q)^2}+\frac{c'(g_i)(1-p-q)}{p-q}\end{aligned} \tag{6}$$

由于截断后的真实值 $c'(g_i)$ 在实际应用中很小, 在 $\mathrm{Var}[\tilde{c}(g_i)]$ 中忽略其影响. 根据公式 (4) 中的 p 与 q 值, 把公式 (6) 重写成公式 (7).

$$\mathrm{Var}[\tilde{c}(g_i)]\approx\frac{n_kq(1-q)}{(p-q)^2}=n_k\times\frac{\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}{\left(\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}-1\right)^2} \tag{7}$$

定理 1–定理 3 证明了 LPRR 算法的隐私性、无偏性与相应方差. 然而, 事实表 T_K 形成的任意节点组合中的元组计数误差由两部分构成: 截断误差与扰动误差. 任意用户 u_i 利用 τ-截断机制对自己的元组集合截断处理后, 在服务器 s_K 中产生 $n^{K-2}\tau$ 条元组. 本文利用截断偏差来度量由 τ-截断机制产生的截断误差.

定理 4. 服务器 s_K 中表 T_K 的所有属性层次树纵向路径组合有 $(\log_b m)^d$ 种. 对于任意组合 g_i, 由 τ-截断机制产生的元组计数截断偏差为:

$$|c'(g_i)-c(g_i)|=\left|\frac{n^{K-1}\tau}{d(\log_b m)^d b^{\sum_{i=1}^{d+3}j_i}}-c(g_i)\right| \tag{8}$$

证明: 令 $c(g_i)$ 为节点组合 g_i 截断前的真实元组计数, $c'(g_i)$ 为 τ-截断后 g_i 中的元组真实计数. 任意用户 u_i 发送 $n^{K-2}\tau$ 条元组到服务器 s_K, 则 $c'(g_i)=n^{K-1}\tau\big/d(\log_b m)^d b^{\sum_{j=1}^{d+3}j_i}$, $c(g_i)$ 与 $c'(g_i)$ 之间的偏差如公式 (8) 所示.

定理 5. 设 g_i 为表 T_K 产生的 $(\log_b m)^d$ 种纵向路径节点组合中任意一种, g_i 中的元组计数估计值 $\tilde{c}(g_i)$ 引起的方差为:

$$\mathrm{Var}[\tilde{c}(g_i)]=\frac{n}{db^{\sum_{i=1}^{d+3}j_i}}\times n^{K-2}\tau\times\frac{\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}{\left(\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}-1\right)^2} \tag{9}$$

证明: 在服务器 s_K 中, 第 $(j_1,j_2,\ldots,j_{d+3})$ 层的节点组合 g_i 中每个用户贡献 $n^{K-2}\tau$ 个元组, 因此:

$$\mathrm{Var}[\tilde{c}(g_i)]=n_K\times\frac{\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}{\left(\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}-1\right)^2}=\frac{n}{db^{\sum_{i=1}^{d+3}j_i}}\times n^{K-2}\tau\times\frac{\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}{\left(\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}-1\right)^2} \tag{10}$$

其中, $n_K=n^{K-1}\tau/db^{\sum_{i=1}^{d+3}j_i}$ 为表 T_K 中所有节点组合的元组个数.

上述定理 4 与定理 5 阐述了服务器 s_K 中任意节点组合 g_i 产生的元组计数偏差以及扰动所产生的方差. 以下定理从响应 Star-JOIN 查询 (COUNT, SUM) 的角度来度量 LPRR 算法的无偏性及可用性.

定理 6. 对于 $(T_1, T_2,\ldots, T_K)$ 的所有属性层次树纵向路径节点组合有 $|G|=(\log_b m)^{Kd}$ 种. 给定任意 $g_i\in G$, 设 $\tilde{c}_{(1,g_i)}$ 表示 g_i 中响应 COUNT 查询的元组计数估计值, 则 $\mathrm{E}[\tilde{c}_{(1,g_i)}]=c'_{(1,g_i)}$ 成立.

证明: 服务器 $s_1, s_2,\ldots, s_K$ 收集用户扰动后的数据形成 K 张表, K 张表进行连接操作响应查询, 谓词属性跨越 K 个服务器, 服务器 s_i $(1\leqslant i\leqslant K)$ 分别对元组是否满足该服务器上的谓词条件进行判断, 满足则 s_i 对元组的状态置为“1”, 否则置为“0”. 满足所有谓词条件的元组, $s_1,\ldots, s_K$ 对该元组的状态置为“11…1”, 对于 K 个服务器, 则该元组

的状态总共有 2^K 种. 令 $c'_{(1,g_i)}$ 为 g_i 中截断后响应任意一组谓词节点组合的元组真实计数, 即元组状态为“11…1”的情况.

$c''_{(1,g_i)}$ 为 g_i 中截断后用户元组个数的观测值. 则 $c''_{(1,g_i)} = Pc'_{(1,g_i)} + Qc'_{(2,g_i)} + \ldots + Q_{2^K-1}c'_{(2^K,g_i)}$. 其中, 扰动概率 P, Q_1, …, Q_{2^K-1} 的取值分别为:

$$P = \left(\frac{\mathrm{e}^{\varepsilon/\left(1+(K-1)+n^{K-1}\tau\right)}}{\mathrm{e}^{\varepsilon/\left(1+(K-1)+n^{K-1}\tau\right)}+(\log_b m)^d-1}\right)^K, Q_1 = \left(\frac{(\log_b m)^d-1}{\mathrm{e}^{\varepsilon/\left(1+(K-1)+n^{K-1}\tau\right)}+(\log_b m)^d-1}\right)^K,$$
$$Q_2 = \frac{\mathrm{e}^{\varepsilon/\left(1+(K-1)+n^{K-1}\tau\right)}\left((\log_b m)^d-1\right)^{K-1}}{\left(\mathrm{e}^{\varepsilon/\left(1+(K-1)+n^{K-1}\tau\right)}+(\log_b m)^d-1\right)^K}, \ldots, Q_{2^K-1} = \frac{\left((\log_b m)^d-1\right)\left(\mathrm{e}^{\varepsilon/\left(1+(K-1)+n^{K-1}\tau\right)}\right)^{K-1}}{\left(\mathrm{e}^{\varepsilon/\left(1+(K-1)+n^{K-1}\tau\right)}+(\log_b m)^d-1\right)^K} \tag{11}$$

截断后的真实值 $c'_{(1,g_i)}$ 无法获得, 则 $c'_{(1,g_i)}$ 的估计量可以表示为 $\tilde{c}_{(1,g_i)} = \dfrac{c''_{(1,g_i)} - \left(Qc'_{(2,g_i)} + \ldots + Q_{2^K-1}c'_{(2^K,g_i)}\right)}{P}$. 则 $\mathrm{E}[\tilde{c}_{(1,g_i)}] = \mathrm{E}\left[\dfrac{c''_{(1,g_i)} - \left(Qc'_{(2,g_i)} + \ldots + Q_{2^K-1}c'_{(2^K,g_i)}\right)}{P}\right]$. 带入 $c''_{(1,g_i)}$ 表达式, 可知 $\mathrm{E}[\tilde{c}_{(1,g_i)}] = c'_{(1,g_i)}$ 成立.

定理 7. 设 $\tilde{c}_{(1,g_i)}$ 表示 g_i 中响应 COUNT 查询的元组计数估计值, 则 $\tilde{c}_{(1,g_i)}$ 引起的方差为:

$$\mathrm{Var}[\tilde{c}_{(1,g_i)}] = \frac{n}{db^{\sum_{i=1}^{Kd+3} j_i}} \times n^{K-2}\tau \times \frac{\left((\log_b m)^d-1\right)^K \times \left(\left(\mathrm{e}^{\varepsilon/(1+(k-2)n+n^{K-2}\tau)}+(\log_b m)^d-1\right)^K - \left((\log_b m)^d-1\right)^K\right)}{\left(\mathrm{e}^{\varepsilon/(1+(K-2)n+n^{K-2}\tau)}\right)^{2K}} \tag{12}$$

证明: 估计值 $\tilde{c}_{(1,g_i)}$ 产生的方差为:

$$\mathrm{Var}[\tilde{c}_{(1,g_i)}] = \mathrm{Var}\left[\frac{c''_{(1,g_i)} - \left(c'_{(2,g_i)}Q_1 + \ldots + c'_{(2^K,g_i)}Q_{2^K-1}\right)}{P}\right] = \frac{1}{P^2}\mathrm{Var}\left[c''_{(1,g_i)}\right]$$
$$\leqslant \frac{c'_{(1,g_i)}P(1-P) + \left(\dfrac{n \times n^{K-2}\tau}{db^{\sum_{i=1}^{Kd+3} j_i}} - c'_{(1,g_i)}\right)Q_1(1-Q_1)}{P^2} \approx \frac{\dfrac{n^{K-1}\tau}{db^{\sum_{i=1}^{Kd+3} j_i}} \times Q_1(1-Q_1)}{P^2} \tag{13}$$

由公式 (11) 可知 Q_1,…, Q_{2^K-1} 中 Q_1 为最大值. 利用 Q_1 对公式 (13) 放缩后, 带入 P 和 Q_1 可知定理 7 成立.

定理 8. LPRR-JOIN 算法响应 COUNT 查询时, 最坏情况下的均方误差为:

$$\mathrm{Error}(\mathrm{COUNT}) = \frac{n^{K-1}\tau}{b^{\sum_{Kd_q+1}^{Kd+3} j_i}} \times \frac{\left((\log_b m)^d-1\right)^K\left(\mathrm{e}^{\varepsilon/(1+(K-2)n+n^{K-2}\tau)}+(\log_b m)^d-1\right)^K - \left((\log_b m)^d-1\right)^{2K}}{\left(\mathrm{e}^{\varepsilon/(1+(K-2)n+n^{K-2}\tau)}\right)^{2K}}$$
$$\times (\log_b m)^{Kd} \times (2(b-1)\log_b m)^{Kd_q} \times \frac{(r_{\min}^2 + r_{\max}^2)}{2} \tag{14}$$

证明: 当 $(j_1,\ldots,j_{Kd+3})=\{0\}^{Kd+3}$ 时, 响应 COUNT 查询的均方误差最大, 即 Kd_q 个谓词属性均选择第 L^0 层:

$$\mathrm{Error}(\mathrm{COUNT}) = \frac{n}{db^{\sum_{i=0}^{Kd_q} 0 \sum_{Kd_q+1}^{Kd+3} j_i}} \times n^{K-2}\tau \times \frac{\left((\log_b m)^d-1\right)^K \times \left(\left(\mathrm{e}^{\varepsilon/(1+(k-2)n+n^{K-2}\tau)}+(\log_b m)^d-1\right)^K - \left((\log_b m)^d-1\right)^K\right)}{\left(\mathrm{e}^{\varepsilon/(1+(K-2)n+n^{K-2}\tau)}\right)^{2K}}$$
$$\times d \times (\log_b m)^{Kd} \times (2(b-1)\log_b m)^{Kd_q} \times \frac{(r_{\min}^2 + r_{\max}^2)}{2} \tag{15}$$

其中, $r_{\min}$ 与 $r_{\max}$ 分别表示截断权重属性 r 的最小值与最大值. 由公式 (15) 可知定理 8 成立.

定理 9. 结合定理 8 可知 LPRR-JOIN 算法响应任何的 SUM 查询时, 最坏情况下的均方误差为:

$$\mathrm{Error}(\mathrm{SUM}) = \mathrm{Error}(\mathrm{COUNT}) \times \left(\frac{a_{\min}^2 + a_{\max}^2}{2}\right) \tag{16}$$

其中, $a_{\min}$ 与 $a_{\max}$ 分别表示聚集属性 a 的最小值与最大值.

证明: 为了估计 LPRR-JOIN 算法响应 SUM 查询的最坏均方误差, 可由 COUNT 查询的两个极端情况来进行估算. 已知 SUM 查询是用于计算满足谓词条件的聚集属性值的总和, 则以下为 LPRR-JOIN 算法响应 SUM 查询的误差计算过程.

(1) 若满足 COUNT 查询的元组计数及其聚集属性 a 的取值均为 $a_{\min}$, 结合公式 (14) 可知 SUM($a_{\min}$) 的均方误差为: Error(COUNT)×$(a_{\min})^2$;

(2) 若满足 COUNT 查询的元组计数及其聚集属性 a 的取值均为 $a_{\max}$, 结合公式 (14) 可知 SUM($a_{\max}$) 的均方误差为: Error(COUNT)×$(a_{\max})^2$;

结合 Error(COUNT)×$(a_{\min})^2$ 与 Error(COUNT)×$(a_{\max})^2$ 的平均值可知定理 9 成立.

定理 8 与定理 9 从均方误差来度量响应 COUNT 与 SUM 查询的可用性.

定理 10. LPRR-JOIN 算法响应 COUNT 查询时, 最大偏差

$$|\tilde{c}-c| = O\left(\frac{(r_{\min}+r_{\max})(\log m)^{Kd}}{(\varepsilon/(1+(K-2)n+n^{K-2}\tau))^K}\sqrt{\frac{n^{K-1}\tau(\log m)^{Kd_q}\ln(1/\gamma)}{b^{\sum_{Kd_q+1}^{Kd+3} j_i}}}\right) \tag{17}$$

至少以概率 1–γ 成立.

证明: 令 $\tilde{c}$ 表示响应 COUNT 查询的估计元组计数, c 表示响应 COUNT 查询的真实元组计数, 则 c 与 $\tilde{c}$ 之间的偏差满足不等式 $|\tilde{c}-c| \leqslant |c'-c|+|\tilde{c}-c'|$, 其中 c'表示截断后响应 COUNT 查询的元组真实计数.

由于需要响应 COUNT 查询, 并期望响应结果无偏, 则:

$$|c'-c| = 0 \tag{18}$$

其中, $|\tilde{c}-c'|$ 为响应 COUNT 查询的扰动误差. 由定理 7 可知, 响应 COUNT 查询的估计元组计数 $\tilde{c}_{(1,g_i)}$ 引起的最大方差为:

$$\mathrm{Var}[\tilde{c}_{(1,g_i)}] = \mathrm{Var}\left[\tilde{c}_{(1,g_i)}-c'_{(1,g_i)}\right] \leqslant O\left(\frac{n^{K-1}\tau}{db^{\sum_{Kd_q+1}^{Kd+3} j_i}}\times\frac{(\log m)^{Kd}}{(\varepsilon/(1+(K-2)n+n^{K-2}\tau))^{2K}}\right) \tag{19}$$

响应 COUNT 查询截断后的元组计数 c'与估计元组计数 $\tilde{c}$ 的误差可以表示为:

$$|\tilde{c}-c'| = \frac{(r_{\min}+r_{\max})}{2}\sum_{\left(2(b-1)\log_b m\right)^{Kd_q}}\sum_{d\left(\log_b m\right)^{Kd}}\left|\tilde{c}_{(1,g_i)}-c'_{(1,g_i)}\right| \tag{20}$$

其中, $d(\log_b m)^{Kd}$ 为两次用户分组数, $(2(b-1)\log_b m)^{Kd_q}$ 为 Kd_q 个谓词分解的最大区间.

定义随机变量 $\tilde{c}_{(1,g_i)}-c'_{(1,g_i)}$. 由定理 6 可知该随机变量的均值为 0, 则下列不等式成立:

$$\left|\tilde{c}_{(1,g_i)}-c'_{(1,g_i)}\right| \leqslant \frac{n^{K-1}\tau}{db^{\sum_{Kd_q+1}^{Kd+3} j_i}}\times\frac{\left(\mathrm{e}^{\varepsilon/(1+(k-2)n+n^{K-2}\tau)}+(\log_b m)^d-1\right)^K}{\left(\mathrm{e}^{\varepsilon/(1+(k-2)n+n^{K-2}\tau)}\right)^K} \tag{21}$$

根据伯恩斯坦不等式可知:

$$\begin{aligned}
\Pr[|\tilde{c}-c'|\geqslant\lambda] &= \Pr\left[\frac{(r_{\min}+r_{\max})}{2}\sum_{\left(2(b-1)\log_b m\right)^{Kd_q}}\sum_{d\left(\log_b m\right)^{Kd}}\left|\tilde{c}_{(1,g_i)}-c'_{(1,g_i)}\right|\geqslant\lambda\right]\\
&= \Pr\left[\sum_{\left(2(b-1)\log_b m\right)^{Kd_q}}\sum_{d\left(\log_b m\right)^{Kd}}\left|\tilde{c}_{(1,g_i)}-c'_{(1,g_i)}\right|\geqslant\frac{2}{r_{\min}+r_{\max}}\lambda\right]\\
&\leqslant 2\exp\left(-\frac{\left(\frac{2}{r_{\min}+r_{\max}}\lambda\right)^2}{\sum_{\left(2(b-1)\log_b m\right)^{Kd_q}}\sum_{d\left(\log_b m\right)^{Kd}}\mathrm{Var}\left[\tilde{c}_{(1,g_i)}-c'_{(1,g_i)}\right]+\frac{4}{3(r_{\min}+r_{\max})}M\lambda}\right)\\
&\leqslant 2\exp\left(-O\left(\frac{\left(\frac{\lambda}{r_{\min}+r_{\max}}\right)^2}{\frac{n^{K-1}\tau}{b^{\sum_{Kd_q+1}^{Kd+3} j_i}}\times\frac{(\log m)^{2Kd+Kd_q}}{(\varepsilon/(1+(K-2)n+n^{K-2}\tau))^{2K}}}\right)\right)
\end{aligned} \tag{22}$$

令 $\gamma=\exp\left(-O\left(\dfrac{\left(\dfrac{\lambda}{r_{\min}+r_{\max}}\right)^2}{\dfrac{n^{K-1}\tau}{b^{\sum_{Kd_q+1}^{Kd+3}j_i}}\times\dfrac{(\log m)^{2Kd+Kd_q}}{(\varepsilon/(1+(K-2)n+n^{K-2}\tau))^{2K}}}\right)\right)$,

则存在 $\lambda=O\left(\dfrac{(r_{\min}+r_{\max})(\log m)^{Kd}}{(\varepsilon/(1+(K-2)n+n^{K-2}\tau))^{K}}\sqrt{\dfrac{n^{K-1}\tau(\log m)^{Kd_q}\ln(1/\gamma)}{b^{\sum_{Kd_q+1}^{Kd+3}j_i}}}\right)$, 使得 $|\tilde{c}-c'|\leqslant\lambda$ 至少以概率 1–γ 成立. 结合公式 (18) 可知定理 (10) 成立.

3.1 截断阈值 τ 的选择

截断阈值 τ 是否合适直接制约着 LPRR-JOIN 算法响应 Star-JOIN 查询结果的方差与偏差. 阈值 τ 过大, 尽管偏差小, 然而会导致本地扰动误差过大; 阈值 τ 过小, 本地扰动误差小, 然而会导致偏差过大, 数据丢失严重. 根据定理 2 和定理 4 可知, 对于服务器 s_K 中任意一组节点组合 g_i 中元组计数的误差由截断偏差与扰动误差两部分组成. 本文通过构建截断偏差与扰动误差之间的优化函数来求解合适的裁剪阈值 τ, 并实现 LPRR-JOIN 算法中的 *Tao-Search*(τ_1, τ_2) 方法.

3.1.1 基于优化理论求解 τ 值

定理 11. 对于 T_K 中所有属性纵向路径上节点组合 G, 其中任意一组节点组合 g_i, 均可获得合适的截断阈值 $\tau=\dfrac{1}{4}d(\log_b m)^{2d}\ln(1/\gamma)n^{1-K}$, 使得扰动误差与截断产生的偏差均衡.

证明: 根据定理 4 和定理 5 的结论可知, 对于一组节点组合 g_i 中的真实元组计数 $c(g_i)$ 与估计元组计数 $\tilde{c}(g_i)$ 之间的误差可以表示为:

$$|\tilde{c}(g_i)-c(g_i)|\leqslant|c'(g_i)-c(g_i)|+|\tilde{c}(g_i)-c'(g_i)|=\left|\frac{n^{K-1}\tau}{d(\log_b m)^d b^{\sum_{i=1}^{d+3}j_i}}-c(g_i)\right|+|\tilde{c}(g_i)-c'(g_i)| \tag{23}$$

其中, $c'(g_i)$ 表示为截断后节点组合 g_i 中的元组真实计数.

节点组合 g_i 中的估计元组计数 $\tilde{c}(g_i)$ 与截断后元组计数 $c'(g_i)$ 之间的偏差可以表示为:

$$|\tilde{c}(g_i)-c'(g_i)|=\left|\sum_{j=1}^{n_K}Z_j-\sum_{j=1}^{n_K}w_j\right|=\left|\sum_{j=1}^{n_K}(Z_j-w_j)\right| \tag{24}$$

其中, w_j 与 Z_j 分别表示第 j 个元组的真实状态和估计状态.

根据定理 3 可知 $|\tilde{c}(g_i)-c'(g_i)|$ 的均值为 0. 设 Z_j–w_j 为一个随机变量. 由于 Z_j 与 w_j 的取值均为{0, 1}, 则 Z_j-w_j 的取值范围为{−1, 0, 1}. Z_j-w_j 的方差可以表示为:

$$\begin{aligned}\mathrm{Var}[Z_j-w_j]&=\mathrm{E}\left[(Z_j-w_j)^2\right]-[\mathrm{E}(Z_j-w_j)]^2\\&=(-1)^2\times\frac{1}{2}\times\frac{(\log_b m)^d-1}{\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}+1^2\times\frac{1}{2}\times\frac{(\log_b m)^d-1}{\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}\\&=\frac{(\log_b m)^d-1}{\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}\end{aligned} \tag{25}$$

由定理 2 可知: $|Z_j-w_j|\leqslant\dfrac{1}{p-q}$. 由公式 (4) 可知 p 与 q 的取值, 进而可得:

$$|Z_j-w_j|\leqslant\frac{\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}{\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}-1}.$$

根据伯恩斯坦不等式可知:

$$\begin{aligned}\Pr[|\tilde{c}(g_i)-c'(g_i)|\geqslant\lambda]&=\Pr\left[\sum_{j=1}^{n_K}|Z_j-w_j|\geqslant\lambda\right]\\&\leqslant 2\exp\left(-\frac{\lambda^2}{2\sum_{i=1}^{n_K}\mathrm{Var}[Z_j-w_j]+\frac{2}{3}M\lambda}\right)\leqslant 2\exp\left(-\frac{\lambda^2}{2n_K\dfrac{(\log_b m)^d-1}{\mathrm{e}^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}}\right)\end{aligned} \tag{26}$$

令 $\gamma=\exp\left(-\dfrac{\lambda^2}{2n_K\dfrac{e^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}{e^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}}\right)$, 则 $\lambda=\sqrt{\dfrac{\left((\log_b m)^d-1\right)2n_K\ln(1/\gamma)}{e^{\varepsilon/\left(1+(K-2)n+n^{K-2}\tau\right)}+(\log_b m)^d-1}}\leqslant O\left(\sqrt{n_K\ln(1/\gamma)}\right)$ 时使得 $|\tilde{c}(g_i)-c'(g_i)|\leqslant\lambda$ 至少以概率 $1-\gamma$ 成立.

结合以上叙述, $|\tilde{c}(g_i)-c(g_i)|$ 的误差可以表示为:

$$|\tilde{c}(g_i)-c(g_i)|\leqslant\left|\frac{n^{K-1}\tau}{d(\log_b m)^d b^{\sum_{i=1}^{d+3}j_i}}-c(g_i)\right|+O\left(\sqrt{n_K\ln(1/\gamma)}\right)\tag{27}$$

为了计算方便, 定义如下优化函数:

$$F(\tau)=\left|\frac{n^{K-1}\tau}{d(\log_b m)^d b^{\sum_{i=1}^{d+3}j_i}}-c(g_i)\right|+O\left(\sqrt{n_K\ln(1/\gamma)}\right)\tag{28}$$

其中, $n_K=\dfrac{n^{K-1}\tau}{db^{\sum_{i=1}^{d+3}j_i}}$. 当 $(j_1,\ldots,j_{Kd+3})=\{0\}^{d+3}$ 时, 在所有层次组合中, 当且仅当 d+3 个属性都选择 L^0 层且 $n_K=n^{K-1}\tau/d$ 时, 误差最大. 对公式 (28) 两侧同时对 τ 求导可得:

$$F'(\tau)=\frac{n^{K-1}}{d(\log_b m)^d}+\frac{\sqrt{n^{K-1}\ln(1/\gamma)}}{2\sqrt{\tau d}}\tag{29}$$

当 $F'(\tau)=0$ 时, 函数 $F(\tau)$ 在 $\tau=\dfrac{1}{4}d(\log_b m)^{2d}\ln(1/\gamma)n^{1-K}$ 取得最小值.

3.1.2 基于用户分组策略求解 τ 值

若能够获得用户所拥有元组个数的总体分布, 则从总体分布中计算出合适的 τ 值. 例如, 以真实 IPUMS-P 数据集为例, 根据该数据集中 1 428 037 个用户的元组个数总体分布 (如图 6 所示) 可知, 97% 的用户的元组个数小于 6. 因此, 截断阈值 τ 的理想选择区间在 [1,6] 之间.

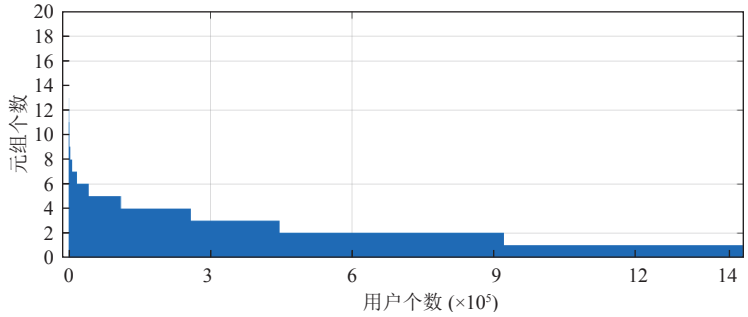


图 6 IPUMS-P 用户元组个数总体分布

为了获得所有用户元组个数的总体分布, 把 n 个用户分为两组: βn, $(1-\beta)n$. βn 个用户寻找合适的 τ 值, $(1-\beta)n$ 个用户响应 Star-JOIN 查询. βn 个用户中每个用户利用 GRR 机制对自身的元组个数进行扰动, 然后将其噪音值发送给收集者. 收集者汇聚 βn 个用户的元组个数, 利用中位数计算合适的 τ 值. 算法 3 是 *Tao-Search*() 的实现细节.

算法 3. *Tao-Search* 算法.

输入: n, β, ε, 元组个数的值域 d';

输出: τ.

1. **for each** user u_i in [βn] **do**
2. 用户 u_i 本地扰动自己拥有的元组个数 $f'_i\leftarrow$GRR(f_i, ε);
3. 用户 u_i 报告 f'_i 给收集端;
4. **for each** f'_i in [d'] **do**
5. 收集者计算元组个数为 f'_i 的计数 $c(f'_i)$;

6. 收集者无偏修正 $c(f_i')$ 后获得 $c'(f_i') \leftarrow \dfrac{(\mathrm{e}^{\varepsilon}+d'-1)c(f_i')-\beta n}{\mathrm{e}^{\varepsilon}-1}$;

7. 收集者利用中位数估计合适的 τ, 使其满足: $\tau \leftarrow \sum_{f_i'=1}^{\tau} c'(f_i') > 0.5 \times \sum_{i=1}^{d'} c'(f_i')$;

8. **return** τ.

βn 个用户中每个用户利用 GRR 机制本地扰动自己的元组个数 f_i, 并发送给收集者 (步骤 1–3); 收集者计算并修正每类元组个数的计数 (步骤 4–6); 利用中位数进行求解 τ (步骤 7).

3.2 基于层次树响应 JOIN 查询算法的误差对比

目前, 能够响应 Star-JOIN 查询且满足 ε-uLDP 的算法仅有 HIO-JOIN 与 HIO-JOIN-RDC. 本小节以响应 SUM 查询产生的误差为基础来分析 HIO-JOIN、HIO-JOIN-RDC 以及 LPRR-JOIN 算法的优劣性. 首先对 3 种响应 Star-JOIN 查询算法的方差进行 O 放缩. 例如, 根据定理 9, 可以把 LPRR-JOIN 算法响应 SUM 查询的方差进行 O 放缩为 $O\left(\dfrac{n^{K-1}\tau(1+(K-2)n+n^{K-2}\tau)^{2K}(\log m)^{2Kd}(r_{\min}^2+r_{\max}^2)(a_{\min}^2+a_{\max}^2)}{\varepsilon^{2K}}\right)$. 同理 HIO-JOIN 与 HIO-JOIN-RDC 响应 SUM 查询产生的方差也可以进行放缩, 如表 1 所示. 由表 1 可知, LPRR-JOIN 算法响应 SUM 查询的均方误差是 HIO-JOIN 与 HIO-JOIN-RDC 算法的 $1/4d$ 倍, 由于倍数小于 1, 则 LPRR-JOIN 算法 SUM 查询的均方误差低于 HIO-JOIN 与 HIO-JOIN-RDC 算法的查询误差.

表 1 响应 SUM 查询误差描述

算法名称	响应SUM查询误差 (设置$dq=d$为最坏情况)
HIO-JOIN	$O\left(\dfrac{n^{K-1}\tau\left(1+(K-2)n+n^{K-2}\tau\right)^{2K}4d(\log m)^{2Kd}\left(r_{\min}^2+r_{\max}^2\right)\left(a_{\min}^2+a_{\max}^2\right)}{\varepsilon^{2K}}\right)$
HIO-JOIN-RDC	$O\left(\dfrac{n^{K-1}\tau\left(1+(K-2)n+n^{K-2}\tau\right)^{2K}4d(\log m)^{2Kd}\left(r_{\min}^2+r_{\max}^2\right)\left(a_{\min}^2+a_{\max}^2\right)}{\varepsilon^{2K}}\right)$
LPRR-JOIN	$O\left(\dfrac{n^{K-1}\tau\left(1+(K-2)n+n^{K-2}\tau\right)^{2K}(\log m)^{2Kd}\left(r_{\min}^2+r_{\max}^2\right)\left(a_{\min}^2+a_{\max}^2\right)}{\varepsilon^{2K}}\right)$

4 实验结果与分析

实验平台采用 Windows 7 系统, 4 核 Intel CPU (4 GHz), 8 GB 内存的配置, 实验代码采用 Python 实现. 采用合成数据集 SYN-1 和 SYN-2, IPUMS-P 与 ADULT 数据集模拟星形模型并响应 Star-JOIN 查询. SYN-1 和 SYN-2 包含 6 种顺序属性, 每个维度的属性值都服从 ($\mu=m/2, \sigma=m/4$) 正态分布. SYN-1 包含 100 万个用户、SYN-2 包含 300 万个用户; IPUMS-P 是美国人口普查数据集, 约包含 330 万个用户, 3 张表包含 Age、Wkswork、Inctot、Presgl、Trantime 和 Uhrswork 这 6 种顺序属性; Adult 来源于 UCL ML 库, 为美国人口普查收入信息, 共包含 32 561 条记录, 3 张表包含 Age、Fnlwgt、Education-num、Capital、Capital-loss、Hours-per-week 这 6 种顺序属性. 每个用户匹配的元组数在 [1,10] 范围内. 4 种数据集具体细节如表 1 所示.

结合上述 4 种数据集, HIO-JOIN、HIO-JOIN-RDC、LPRR-JOIN-T、LPRR-JOIN-E 算法在归一化均方误差 (normalized mean squared error, *NMSE*) 与平均相对误差 (mean relative error, *MRE*) 的标准下比较响应聚集函数为 COUNT、SUM 与 AVG 的 Star-JOIN 查询精度. 其中 LPRR-JOIN-T 表示利用优化函数求解 τ 值的 LPRR-JOIN 算法; LPRR-JOIN-E 利用用户分组求解 τ 值的 LPRR-JOIN 算法. 归一化均方误差如公式 (30) 所示:

$$NMSE(\mathrm{P}(\mathcal{Q})) = \frac{1}{|\mathcal{Q}|}\sum_{Q\in\mathcal{Q}}\left(\frac{\mathrm{P}(Q,\mathcal{R}(T))-\mathrm{P}(Q,T)}{\Sigma_T}\right)^2 \tag{30}$$

其中, $T = T_1 \bowtie T_2 \bowtie \ldots \bowtie T_K$, 表示 K 张表之间星形连接; $\mathcal{Q}$ 表示所有基于 T 的 Star-JOIN 查询的集合; 对于 Star-JOIN

查询下的 COUNT 查询 $\Sigma_T = |T|$, 而对于 SUM 查询有 $\Sigma_T = \sum |t[a]|$.

平均相对误差如公式 (31) 所示:

$$MRE(\mathrm{P}(Q)) = \frac{1}{|Q|}\sum_{Q\in Q}\left|\frac{\mathrm{P}(Q,\mathcal{R}(T)) - \mathrm{P}(Q,T)}{\mathrm{P}(Q,T)}\right| \tag{31}$$

其中, $\mathrm{P}(Q, T)$ 和 $\mathrm{P}(Q,\mathcal{R}(T))$ 分别为 Q 查询的真实与噪音结果.

实验参数的选取分别为 ε, vol, d_q, τ. 其中, ε 为隐私预算, 取值分别为 1.0, 2.0, 3.0, 4.0, 5.0. vol 为谓词查询范围在整个值域的占比, 取值分别为 0.07, 0.15, 0.22, 0.3. d_q 为查询谓词的属性个数, 取值分别为 1, 2, 3. τ 为截断阈值, 取值分别为 1, 2, 3, 4.

4.1 基于 ε 变化的 4 种算法性能比较

为了与 HIO-JOIN、HIO-JOIN-RDC 算法的参数设置保持一致, 并且在相同标准对比算法的优劣, 本文结合 4 种数据集, 设置参数每张关系属性个数 d=2, 每个属性值域大小 m=25 或 125, 谓词查询范围与属性值域的比率 vol=0.15, 谓词查询个数 d_q=1, 截断阈值 τ=1, 层次树扇出 b=5, 用户采样率参数 β=0.2. 首先变化隐私预算 ε, 对 4 种算法进行性能对比分析. 图 7–图 10 中 (a)–(c) 描述了 HIO-JOIN、HIO-JOIN-RDC、LPRR-JOIN-T、LPRR-JOIN-E 算法在真实与合成数据集上的 *NMSE* 与 *MRE* 的比较结果. 随着 ε 从 1.0 变化到 5.0 时, ε 越大添加的噪音越少, 查询结果更接近真实值, 从而导致所有算法响应查询的 *NMSE* 值与 *MRE* 值均呈现下降趋势. 在合成数据集 SYN-2 上, LPRR-JOIN-T 和 LPRR-JOIN-E 算法比其他 2 种算法的查询精度更高, 其原因是纵向路径扰动算法造成的误差比较低. 在 ε=1.0 时, HIO-JOIN 算法的 NMSE 值约为 LPRR-JOIN-T 算法的 10 倍, 约为 LPRR-JOIN-E 算法的 5 倍; 而 HIO-JOIN 算法的 *MRE* 值则约为 LPRR-JOIN-T 算法的 12 倍, 约为 LPRR-JOIN-E 算法的 8 倍. 在相同参数的比较下, LPRR-JOIN-T 算法的查询精度要高于 LPRR-JOIN-E 算法, 其原因是 LPRR-JOIN-E 算法利用 βn 个用户去寻找截断阈值 τ, $(1-\beta)n$ 个用户响应查询, 因此分配到每个层次的用户数减少, *NMSE* 和 *MRE* 值增加.

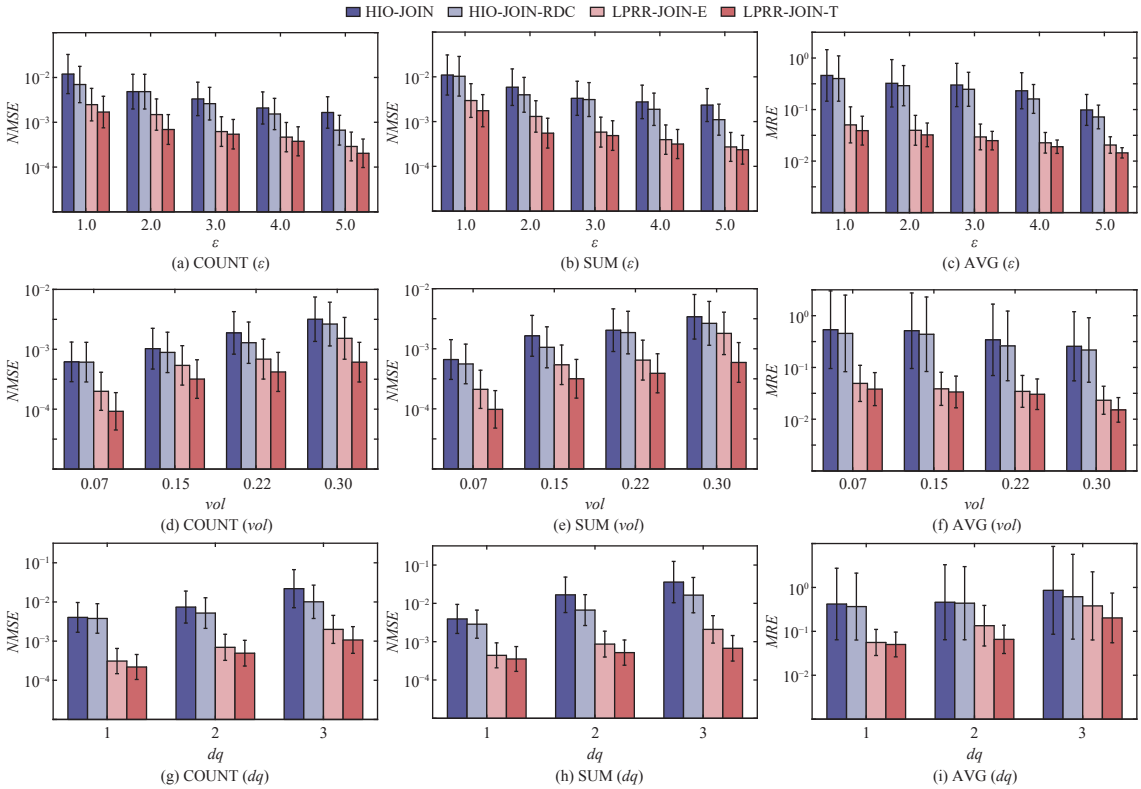


图 7 基于 SYN-1 数据集的算法对比

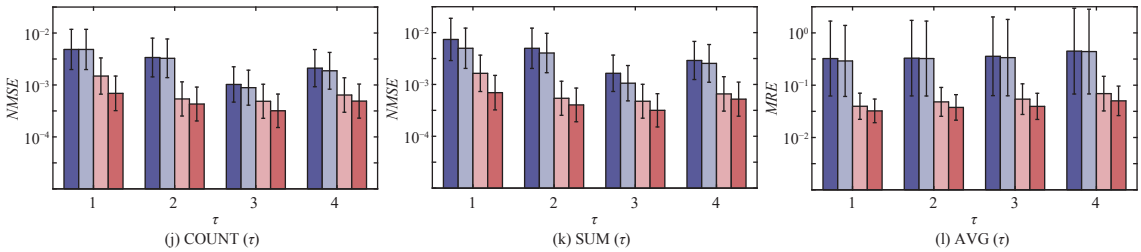


图 7　基于 SYN-1 数据集的算法对比 (续)

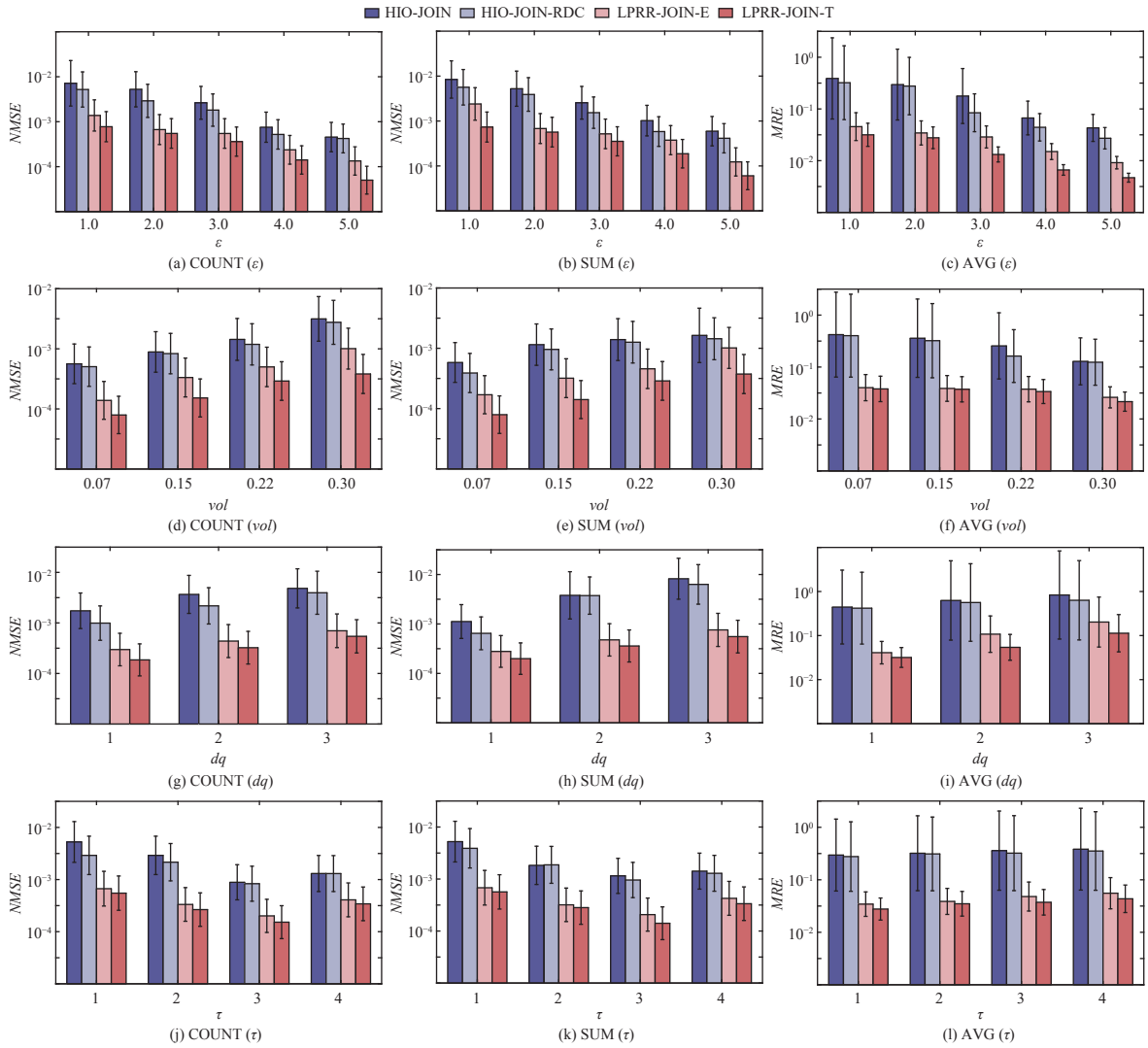


图 8　基于 SYN-2 数据集的算法对比

4.2　基于 *vol* 变化的 4 种算法性能比较

结合 4 种数据集, 设定 d=2, m=25 或 125, b=5, d_q=1, ε=2.0, β=0.2, 优化函数获得理论值 τ=3, 用户分组获得经验值 τ=2. 图 7–图 10 中 (d)–(f) 描述了算法 HIO-JOIN、HIO-JOIN-RDC、LPRR-JOIN-T、LPRR-JOIN-E 的 *NMSE*

与 *MRE* 的对比结果. 由实验结果可以发现, 随着谓词查询范围占比 *vol* 的增加响应 COUNT、SUM 查询的 *NMSE* 值上升, 响应 AVG 查询的 *MRE* 下降, 其原始是查询所覆盖的节点组合个数增加, 从而导致噪音累加. HIO-JOIN 算法的查询精度略低于 HIO-JOIN-RDC 算法, 其原因是 HIO-JOIN-RDC 算法实现了谓词范围的最优分解, 覆盖节点数减少, 进而噪音减少. 在 4 种数据集上, LPRR-JOIN-E 算法优于 HIO-JOIN 和 HIO-JOIN-RDC. 而 LPRR-JOIN-T 算法最优, 其原因是该算法能够通过优化函数寻找到合适的截断阈值.

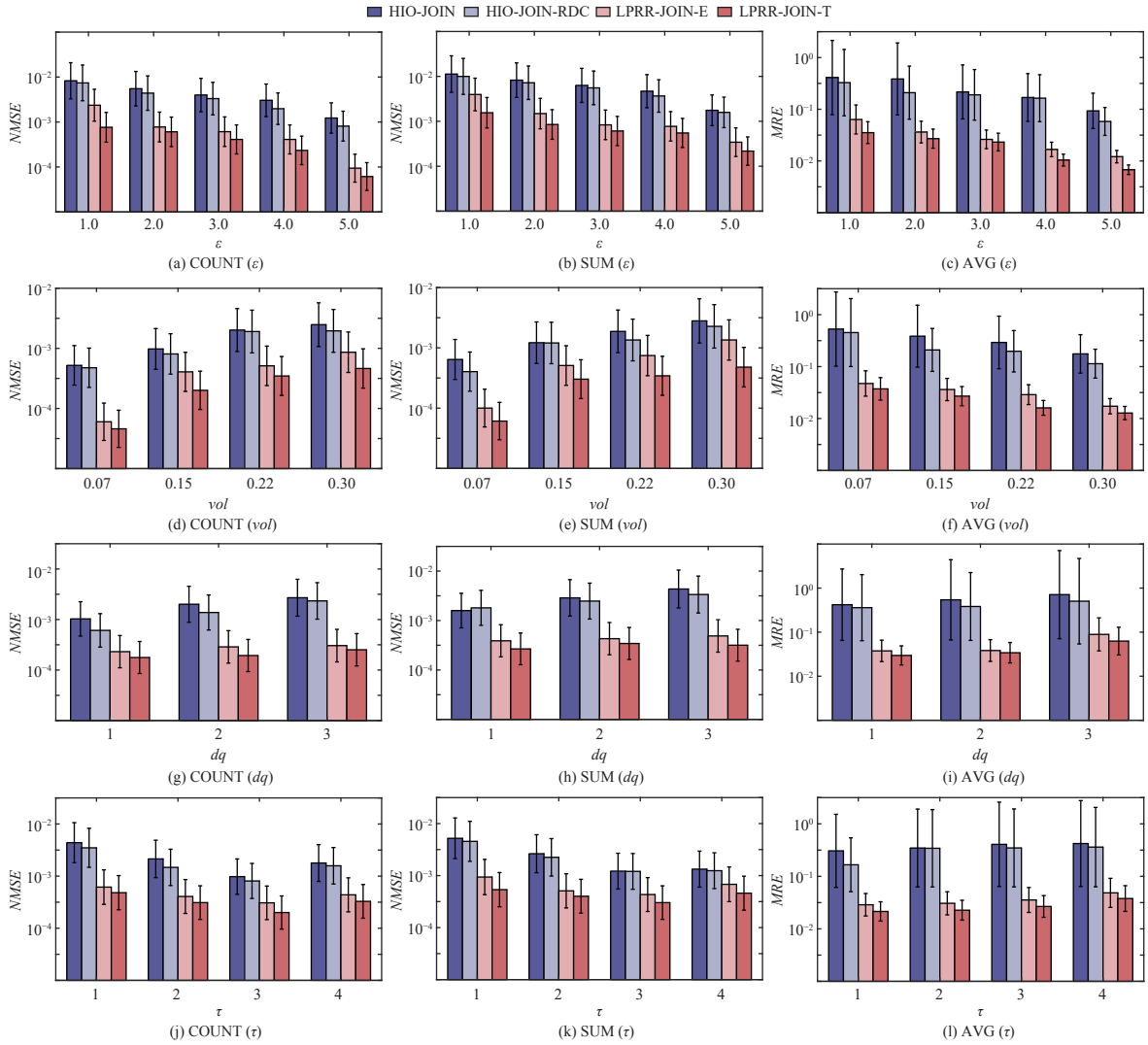


图 9　基于 IPUMS-P 数据集的算法对比

4.3　基于 d_q 变化的 4 种算法性能比较

结合 4 种数据集, 设定 d=2, m=25 或 125, b=5, vol=0.07, ε=2.0, τ=1, β=0.2. 图 7–图 10 中 (g)–(i) 描述了算法 HIO-JOIN、HIO-JOIN-RDC、LPRR-JOIN-T、LPRR-JOIN-E 的 *NMSE* 和 *MRE* 的比较结果. 从实验结果可以发现随着 *dq* 的增加, 响应 COUNT, SUM 与 AVG 查询算法的误差值增加, 原因是查询导致覆盖的节点组合个数增加, 从而导致噪音累加. 从图 7 和图 8 的实验结果可知, 在数据集 SYN-1 上所有算法的查询精度均略低于 SYN-2, 其原因是 SYN-2 的用户数大于 SYN-1 的用户数, 因此分配到每个层次的用户数增加. 每层用户数增加会直接减少

NMSE 和 *MRE*. 由表 2 可知数据集 SYN-2 与 IPUMS-P 响应查询的用户数近似相同, 因此所有算法在 SYN-2 和 IPUMS-P 上的查询精度近似, 实验结果如图 8 和图 9 所示. 从图 10 的实验结果可知, 所在算法在 ADULT 数据集上的查询精度低于 SYN-1、SYN-2 与 IPUMS-P 数据集, 这是因为 ADULT 数据集的用户数较少, 因此分配到每层的用户数减少, 导致查询精度低.

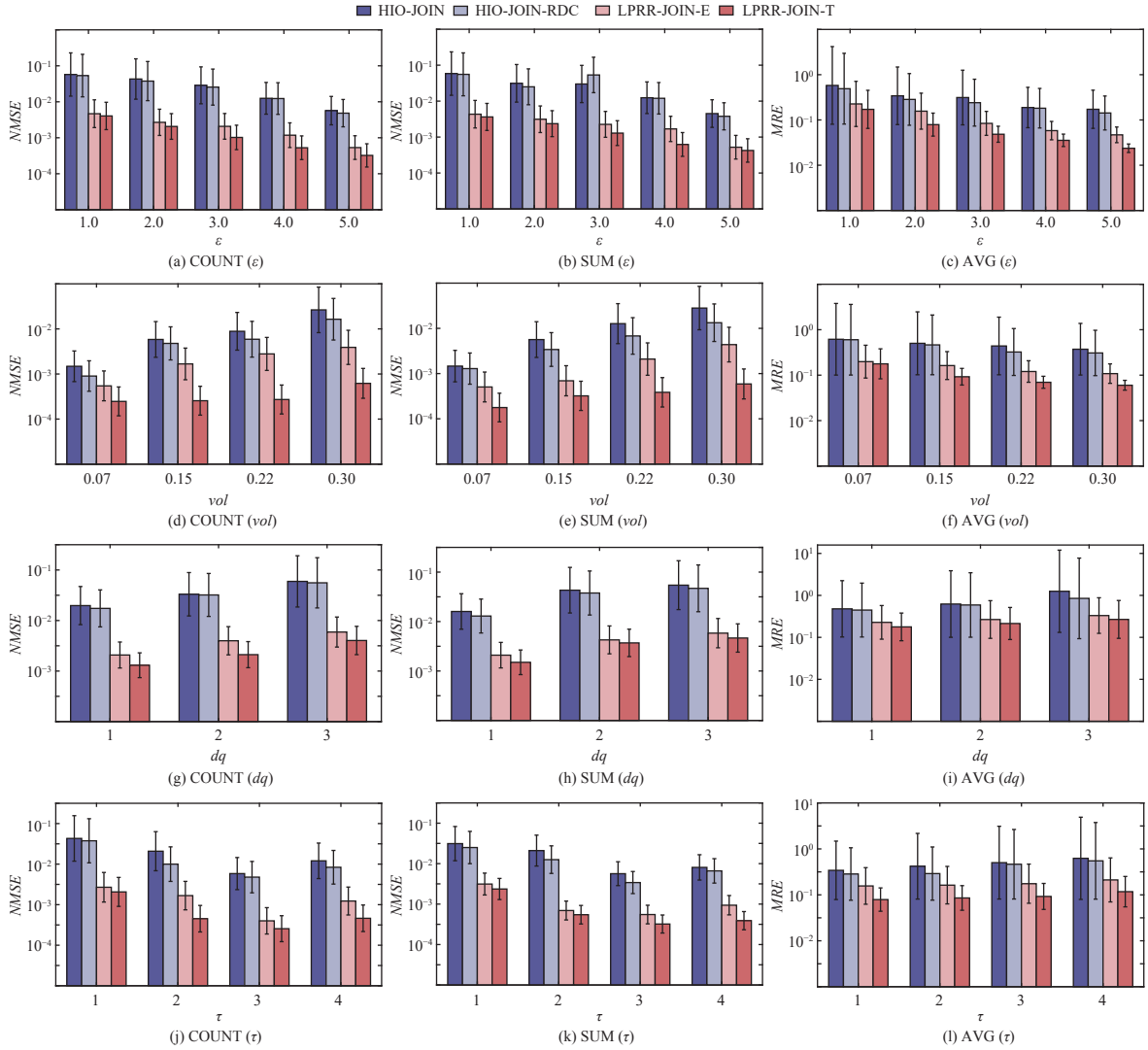


图 10　基于 ADULT 数据集的算法对比

表 2　实验数据集描述

名称	分布	用户数	维度	值域大小
SYN-1	Normal	1 000 000	6	125
SYN-2	Normal	3 000 000	6	125
IPUMS-P	—	3 210 231	10	125
ADULT	—	32 561	6	25 125

4.4　基于 τ 变化的 4 种算法性能比较

结合 4 种数据集, 设定参数 d=2, m=25 或 125, b=5, vol=0.15, ε=2.0, β=0.2. 图 7–图 10 中 (j)–(l) 描述了算法

HIO-JOIN、HIO-JOIN-RDC、LPRR-JOIN-T、LPRR-JOIN-E 的 NMSE 和 MRE 的比较结果. 从实验结果可以发现, 随着 τ 从 1 增加到 4, 所有算法响应 COUNT, SUM 查询的 NMSE 值先降低后增加, 在 τ=3 时最优. 响应 AVG 查询的 MRE 值增加, 在 τ=1 时最优. 从总体来看, LPRR-JOIN-T 与 LPRR-JOIN-E 算法利用了层次树的纵向结构与优化函数求解值 τ 等技术, 因此 LPRR-JOIN-T 与 LPRR-JOIN-E 算法在 3 个数据集上的查询精度都优于 HIO-JOIN 与 HIO-JOIN-RDC 算法,

5 结束语

本文研究了在本地化差分隐私下响应多表星形连接查询问题, 提出了响应 Star-JOIN 查询的本地扰动算法 LPRR-JOIN 来解决现有方法存在的不足. 该算法采用两次用户分组策略, 并结合层次树的纵向特征扰动与用户关联的所有元组. 为了均衡截断误差与扰动误差, LPRR-JOIN 算法分别通过构造总体误差优化函数与用户分组策略求解合适的 τ 值. 从 ε-uLDP 的角度证明了 LPRR-JOIN 算法的隐私性, 并通过在数据集上的实验结果验证了 LPRR-JOIN 算法在响应 Star-JOIN 查询时具有较好的可用性.

References:

[1] Duchi JC, Jordan MI, Wainwright MJ. Local privacy and statistical minimax rates. In: Proc. of the 54th Annual Symp. on Foundations of Computer Science. Berkeley: IEEE, 2013. 429–438. [doi: 10.1109/FOCS.2013.53]

[2] Xu M, Ding BL, Wang TH, Zhou JR. Collecting and analyzing data jointly from multiple services under local differential privacy. Proc. of the VLDB Endowment, 2020, 13(12): 2760–2772. [doi: 10.14778/3407790.3407859]

[3] Wang TH, Blocki J, Li NH, Jha S. Locally differentially private protocols for frequency estimation. In: Proc. of the 26th USENIX Conf. on Security Symp. Vancouver: USENIX Association, 2017. 729–745.

[4] Wang TH, Ding BL, Zhou JR, Hong C, Huang ZC, Li NH, Jha S. Answering multi-dimensional analytical queries under local differential privacy. In: Proc. of the 2019 Int'l Conf. on Management of Data. Amsterdam: ACM, 2019. 159–176. [doi: 10.1145/3299869.3319891]

[5] Kairouz P, Bonawitz K, Ramage D. Discrete distribution estimation under local privacy. In: Proc. of the 33rd Int'l Conf. on Machine Learning. New York: JMLR.org, 2016. 2436–2444.

[6] Narayan A, Haeberlen A. DJoin: Differentially private join queries over distributed databases. In: Proc. of the 10th USENIX Conf. on Operating Systems Design and Implementation. Hollywood: USENIX Association, 2012. 149–162.

[7] Yao AC. Protocols for secure computations. In: Proc. of the 23rd Annual Symp. on Foundations of Computer Science. Chicago: IEEE, 1982. 160–164. [doi: 10.1109/SFCS.1982.38]

[8] Dwork C, McSherry F, Nissim K, Smith A. Calibrating noise to sensitivity in private data analysis. In: Proc. of the 3rd Theory of Cryptography Conf. on Theory of Cryptography. New York: Springer, 2006. 265–284. [doi: 10.1007/11681878_14]

[9] Johnson N, Near JP, Song D. Towards practical differential privacy for SQL queries. Proc. of the VLDB Endowment, 2018, 11(5): 526–539. [doi: 10.1145/3187009.3177733]

[10] Nissim K, Raskhodnikova S, Smith A. Smooth sensitivity and sampling in private data analysis. In: Proc. of the 39th Annual ACM Symp. on Theory of Computing. New York: ACM, 2007. 75–84. [doi: 10.1145/1250790.1250803]

[11] Kotsogiannis I, Tao YC, He X, Fanaeepour M, Machanavajjhala A, Hay M, Miklau G. PrivateSQL: A differentially private SQL query engine. Proc. of the VLDB Endowment, 2019, 12(11): 1371–1384. [doi: 10.14778/3342263.3342274]

[12] Dong W, Yi K. Residual sensitivity for differentially private multi-way joins. In: Proc. of the 2021 Int'l Conf. on Management of Data. New York: ACM, 2021. 432–444. [doi: 10.1145/3448016.3452813]

[13] Dong W, Fang JR, Yi K, Tao YC, Machanavajjhala A. R2T: Instance-optimal truncation for differentially private query evaluation with foreign keys. In: Proc. of the 2022 Int'l Conf. on Management of Data. New York: ACM, 2022. 759–772. [doi: 10.1145/3514221.3517844]

[14] Dong W, Sun DJ, Yi K. Better than composition: How to answer multiple relational queries under differential privacy. Proc. of the ACM on Management of Data, 2023, 1(2): 123. [doi: 10.1145/3589268]

[15] Cormode G, Kulkarni T, Srivastava D. Answering range queries under local differential privacy. Proc. of the VLDB Endowment, 2019, 12(10): 1126–1138. [doi: 10.14778/3339490.3339496]

[16] Zhang XJ, Zhou D, Xu YX, Lin DD, Ji SL, Meng XF. Answering private multidimensional analytical queries with hierarchical structure. SCIENTIA SINICA Informationis, 2023, 53(6): 1111–1131 (in Chinese with English abstract). [doi: 10.1360/SSI-2022-0310]

[17] Joseph M, Roth A, Ullman J, Waggoner B. Local differential privacy for evolving data. In: Proc. of the 32nd Int'l Conf. on Neural Information Processing Systems. Red Hook: Curran Associates Inc., 2018. 2381–2390.

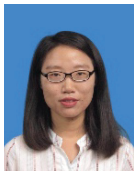
[18] McSherry F. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. Communications of the ACM, 2010, 53(9): 89–97. [doi: 10.1145/1810891.1810916]

[19] Warner SL. Randomized response: A survey technique for eliminating evasive answer bias. Journal of the American Statistical Association, 1965, 60(309): 63–69. [doi: 10.1080/01621459.1965.10480775]

附中文参考文献:

[16] 张啸剑, 周丹, 徐雅鑫, 林东岱, 纪守领, 孟小峰. 基于层次结构的隐私多维分析查询算法. 中国科学: 信息科学, 2023, 53(6): 1111–1131. [doi: 10.1360/SSI-2022-0310]

张啸剑(1980－), 男, 博士, 教授, 主要研究领域为差分隐私, 差分隐私与机器学习, AI 增强型数据管理, 图数据管理.



王宁(1988－), 女, 博士, 副教授, CCF 专业会员, 主要研究领域为数据隐私保护, 数据管理.

曹小杰(1999－), 女, 硕士生, 主要研究领域为差分隐私.

孟小峰(1964－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为云数据管理, 网络数据管理, 隐私保护.