南开大学

Nankai University

南　　开　　大　　学

计 算 机 学 院

数据安全论文阅读

AHEAD: Adaptive Hierarchical Decomposition for Range Query under Local Differential Privacy

---

穆禹宸　　2012026

年级：2020 级

专业：信息安全、法学双学位班

指导教师：刘哲理

2023 年 6 月 30 日

目录

一、 论文信息 **1**

二、 论文动机 **1**

三、 论文贡献 **1**

四、 论文方法 **2**

（一） 简单对比 . . . 2

（二） 具体方法 . . . 3

1. 第一步：用户分区（UP——User Partition） . . . 3
2. 第二步：噪声频率构建（NFC——Noisy Frequency Construction） . . . 3
3. 第三步：新的分解生成（NDG——New Decomposition Generation） . . . 3
4. 第四步：后处理（PP——Post-processing） . . . 4

（三） 参数选择 . . . 4

（四） 扩展到高维查询 . . . 5

1. 扩展到二维 . . . 5
2. 扩展到更高维 . . . 5
3. 具体步骤 . . . 6

五、 我的思考 **7**

（一） 论文优点 . . . 7

（二） 缺陷不足 . . . 7

（三） 可能的改进 . . . 7

一、 论文信息

论文题目：《AHEAD: Adaptive Hierarchical Decomposition for Range Query under Local Differential Privacy》[1]

发表于：Computer and Communications Security (CCS)2021。

作者为：Linkang Du (Zhejiang University); Zhikun Zhang (CISPA Helmholtz Center for Information Security); Shaojie Bai (Zhejiang University); Changchang Liu (IBM Research); Shouling Ji (Zhejiang University); Peng Cheng (Zhejiang University); Jiming Chen (Zhejiang University)

一作单位为：浙江大学。

二、 论文动机

隐私保护是当前互联网时代面临的一个重要问题。随着个人数据的广泛收集和应用，用户的隐私面临着越来越大的风险。为了保护用户的隐私，研究人员提出了很多隐私保护机制，其中差分隐私是一种重要的方法。**这篇论文的动机在于解决现有基于局部差分隐私（LDP）的范围查询方法的限制。这些方法依据预定义的结构收集用户数据，这种静态的框架在低隐私预算设置下可能导致聚合数据添加过多的噪音**。具体来说，这篇论文的动机可以从几个方面来解释：

首先，差分隐私的应用通常会引入噪声，现有方法基本都存在一些限制。主流的范围查询可以根据维度分为低维（1、2 维）方法——树方法和高维（ 2 维度）方法——网格方法。然而，大多数真实世界的数据集中存在稀疏区域。例如，50-60 岁的人在一家足球俱乐部的成员中所占的比例很小。**因此，在完整树（网格）中值较小的节点（单元格）很可能被注入的噪音淹没。此外，现有技术主要设计用于特定的维度查询，即低维（1、2 维）查询和高（ 2）维度查询。尽管此前的一些方法在技术上不受查询维度的限制，但对于非目标维度的情况，它们的效果较差**。由于实际数据集的维度各异，组合不同情景的算法成为聚合器的需求，从而限制了这些算法的适应性和应用性。因此，此前的研究还有很大的发展空间。从而使得查询结果的准确性下降。因此，研究人员希望能够开发一种能够在保护隐私的前提下，尽可能减小噪声影响，提高查询结果的准确性的方法。

其次，现有的差分隐私查询处理方法存在一些问题。一方面，现有的方法通常采用静态的框架，对所有的查询都采用相同的隐私机制，无法根据数据的分布情况和查询的特点进行灵活调整。另一方面，现有的方法在处理低维数据时表现良好，但在处理高维数据时通常存在一定的局限性。因此，研究人员希望能够开发一种更加灵活和适应不同数据特点的差分隐私查询处理方法，可以结合不同参数的优点进行查询。

在这样的背景下，论文之中所提出的 AHEAD 方法可以在一定程度上解决这些问题。

三、 论文贡献

结合之前的研究动机，AHEAD 方法采用了自适应的子域划分策略和动态的隐私机制，能够根据数据的分布和查询的特点来调整隐私保护的力度。同时，AHEAD 方法通过合并数据域中粒度较小的子域来减小噪声的影响，提高查询结果的准确性。因此，AHEAD 方法的动机就是解决在保护隐私的前提下，实现高效查询处理的问题，弥补现有方法的不足之处。具体体现在以下方面：

首先，论文提出了一种基于局部差分隐私（LDP）的查询处理方法，即 AHEAD（Adaptive DEcomposition for LDP Range Queries）。**AHEAD 采用一种动态算法，可以自适应地确定域**

分解的粒度，以减少插入噪音对范围查询的影响，从而保持出色的实用性能表现。与现有的差分隐私查询处理方法相比，AHEAD 方法具有更高的灵活性和适应性。它采用了自适应的子域划分策略和动态的隐私机制，能够根据数据的分布和查询的特点来调整隐私保护的力度。这种灵活性使得 AHEAD 方法能够在不同的数据情况下提供更好的查询结果。

其次，在保证严格的 LDP 隐私保护的前提下，**AHEAD 理论上证明了分解阈值和树度等参数设置**，以保持高实用性。此外，AHEAD 将这种策略扩展到多维数据情景中，使得这种算法更加通用、灵活。

最后，论文对 AHEAD 方法进行了广泛的实验评估。AHEAD 的实验结果表明了算法的有效性和应用场景的广泛适用性。**通过在多个真实数据集上的大量实验，AHEAD 相对于之前的方法在平衡实用性与隐私之间取得了优势**。论文证明了 AHEAD 方法在保证查询准确性的同时，能够显著减小插入噪声对查询结果的影响，具有较好的实用性能。实验结果表明，AHEAD 方法在低维和高维数据场景下都能取得良好的效果，并且优于其他方法。

此外，**文章还提出了六个针对实际应用的有用观察。这些观察为使用 AHEAD 方法进行实际部署提供了指导**。以指导在实际应用中使用 AHEAD 算法。例如，在处理具有高偏斜性的私有数据时，AHEAD 方法能够优于其他方法。此外，AHEAD 方法还能够处理不同大小的数据域，并且对于较大的数据域，它可以选择更大的子域大小来获得更高的粒度。

因此，这篇论文的贡献包括引入了具有灵活性和适应性的 AHEAD 方法，AHEAD 算法提供了一种全新的解决方案，并在实现隐私保护的同时，兼顾实用性，具有非常实际的意义，具有广阔的应用前景。在通过实验评估证明了 AHEAD 方法的性能优越性后，提供了有用的观察用于实际部署。这些贡献推动了差分隐私查询处理领域的进一步发展，并为隐私保护和高效查询处理提供了新的解决方案。

四、 论文方法

本篇论文提出了一种基于局部差分隐私（LDP）的查询处理方法，称为 AHEAD（Adaptive DEcomposition for LDP Range Queries）。AHEAD 方法旨在解决在保护用户隐私的前提下，实现高效查询处理的问题。

(一) 简单对比

基准策略和自适应策略在处理“频率查询”时的不同。基准策略选择发布每个区间的估计频率值，并通过 FO 机制集成噪声误差以满足 LDP 保证。当回答一个范围查询时，如 frequency([0,5])，问询者想知道区间 [0,5] 的频率值。基准策略的答案是 $0.5+3\sigma^2$。值得注意的是，区间 [0,1] 不对查询答案做出贡献，但却会带来同样程度的噪声，这降低了现有算法的查询精度。另一方面，对于自适应策略，当回答相同的范围查询 frequency([0,5]) 时，自适应策略的答案为 $0.5+2\sigma^2$，相比于基准方法减少了 30%的噪声误差，而且在频率值较小的区间中，自适应策略的噪声误差也会减弱，表现更好。

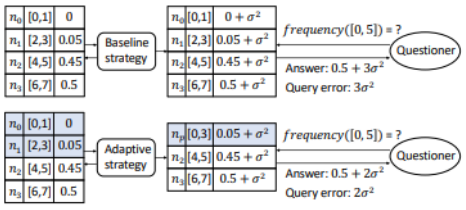

图 1: 论文之中简单的举例对比

上图是论文之中一例。

(二) 具体方法

在具有隐私约束条件下寻找二维数据集的最优划分是困难的。因此，文章提出了一种多阶段层次递归分区策略，以平衡误差并解决现有解决方案的局限性。该策略旨在找到最优域分解，通过合并间隔来减少噪声误差，同时通过假设均匀性引入非均匀误差。

通过下图中的示例展示 AHEAD 的工作流程。在这个例子中，聚合器想要基于 AHEAD 完成有关用户薪资的范围查询任务。薪资数据被分成了 8 个序数级别，即领域大小 $|D| = 8$。树分支因子 $B = 2$，这意味着 AHEAD 树的每个节点最多有两个子节点。在 AHEAD 原型树的每个节点中，n_i 表示节点索引，$[a, b]$ 表示节点间隔，$\hat{f}_i$ 表示估计的频率值。值得注意的是，AHEAD 采用了特殊的采样原则，即将用户划分为组，每个组使用完整的隐私预算。在本地设置中，这个采样原则可以显著降低整体误差。

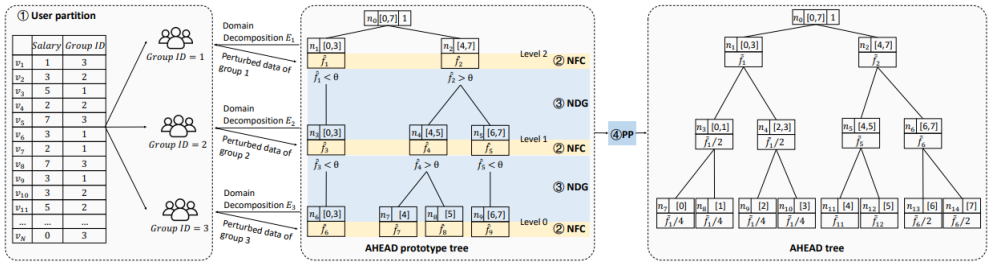

图 2: AHEAD 方法工作流

以下是一维查询的工作流。

1. 第一步：用户分区（UP——User Partition）

如图片左边的虚线框所示，聚合器确定分区的数量 c，其中 $c = \log_B |D|$ 被设置为确保用户被分配到 AHEAD 树的每一层。用户随机选择在范围 $[1, 2, 3, \cdots, c]$ 内的组号。此外，用户也可以利用其公共信息来选择组，例如账户注册时间、用户 ID 等等。分区过程应确保每个组代表总体人口，且用户数量相似。

2. 第二步：噪声频率构建（NFC——Noisy Frequency Construction）

在中间的虚线框中，聚合器首先建立一个代表整个领域的根节点 n_0。之后，聚合器对领域进行初始分解，即将整个领域分成 B 个大小相等的间隔，然后将这些间隔节点附加到根节点 n_0 上。根节点的子节点表示将整个领域划分的一种方式，被称为领域分解 E_1。聚合器选择第一组用户并向他们发送分解 E_1 和隐私预算 ϵ。第一组中的每个用户将私有值 v 投影到 E_1 的间隔上，并通过 OUE 方法 [2] 上传 v 的投影值。在接收到用户的报告后，服务器使用聚合算法获得估计的频率分布 $\hat{F}_1$，表示落在每个节点间隔内的用户比例。

3. 第三步：新的分解生成（NDG——New Decomposition Generation）

聚合器将 $\hat{F}_1 = \left\{\hat{f}_1, \hat{f}_2\right\}$ 中的每个频率值与阈值 θ 进行比较，决定是否进一步划分 E_1 中相应的间隔。具体而言，由于 $\hat{f}_2$ 大于设置的阈值 θ，因此应将 E_1 中的相应间隔 $[4, 7]$ 划分为 B 个相等的子区间 $[4, 5]$ 和 $[6, 7]$。因为节点 n_1 的频率值 $\hat{f}_1$ 不大于 θ，所以区间 $[0, 3]$ 不需要进一步

划分。对于新的间隔节点，我们将它们附加到相应的父间隔节点上。当完全遍历 $\hat{F}_1$ 中的所有元素后，我们可以获得作为分解 E_2 的新一组间隔。

然后，聚合器将分解 E_2 发送给第二组用户，并获得估计的频率分布 F_2。聚合器重复上述步骤，直到所有用户组均应用过，并得到 AHEAD 原型树，如??中间的虚线框所示。由于估计频率小于阈值 θ，在后续交互中，AHEAD 不会将间隔 $[0,3]$ 和 $[6,7]$ 进行分解。为了保证 LDP，$[0,3]$ 和 $[6,7]$ 应被所有用户组估计。

构建原型树时，AHEAD 分别估计每一层，未考虑树中频率值的约束，即子节点频率值之和等于其父节点频率值。因此，在第四步中，我们通过在节点的估计值之间进行**非负性**控制和**加权平均**来进一步提高 AHEAD 的精度。

4. 第四步：后处理（PP——Post-processing）

后处理模块包含两个步骤：**非负性**控制和**加权平均**。首先，AHEAD 通过 Norm-Sub [3] 对同一层中的节点进行处理，以确保节点的估计频率是非负的，且频率之和等于 1。AHEAD 将负值转换为 0，并计算正值之和与 1 之间的差异。然后，每个正值减去平均差异，这个平均差异是通过将总差异除以正估计值的数量得到的。**非负性**处理重复执行，直到所有值都变为非负值。

然后，从底部到顶部，AHEAD 计算非叶节点 n 和其子节点之间的加权平均，更新 n 的估计频率，即通过融合 n 的多个估计值来减少添加的噪声。对于非根节点 n：

$$\tilde{f}(n)=\begin{cases}\lambda_1\hat{f}(n)+\lambda_2\sum_{u\in child(n)}\hat{f}(u), & \text{if u is a leaf node}\\ \lambda_1\hat{f}(n)+\lambda_2\sum_{u\in child(n)}\tilde{f}(u), & \text{o.w.}\end{cases} \tag{1}$$

权重 λ_1 和 λ_2 与估计的差异的反比成正比,即 $\lambda_1=\frac{VAR_{child(n)}}{VAR_{child(n)}+VAR_{(n)}}$ 和 $\lambda_2=\frac{VAR_{(n)}}{VAR_{child(n)}+VAR_{(n)}}$，其中 $VAR_{child(n)}$ 表示节点 n 的子节点方差之和，VAR_n 表示节点 n 的方差。$\tilde{f}$ 表示 $\hat{f}$ 的后处理版本，并将用于回答查询。然后，合并子节点的频率，节点 n 可以达到最小更新方差。

最后，从顶部到底部，AHEAD 在节点区间的一致分布假设下，递归地划分频率值，以获得完整的树（如上图右侧的子图所示），这将用于回答范围查询。

(三) 参数选择

AHEAD 两个重要参数：树度 B 和阈值 θ。由于 AHEAD 已经严格满足 LDP 保证，因此文章旨在探索 B 和 θ 的设置，以使 AHEAD 的整体效用性能最大化。鉴于上述分区策略和实际场景中用户规模较大（$N>10^5$），我们假设每个组具有相同数量的用户。回顾隐私与效用分析中的误差分析，文章关注噪声误差和非均匀误差，这些误差主导了总体估计误差。

选择 θ 直观地说，在选择 AHEAD 的参数时，目标是平衡两种误差，以便 AHEAD 可以实现卓越的性能。

经过文章之中证明，对于一组参数，即树的分支数 B、隐私预算 ϵ、用户规模 N 和组数 c，分解阈值 θ 的设置遵循以下公式：

$$\theta=\sqrt{(B+1)VAR}, \tag{2}$$

其中，VAR 等于 $\frac{4e^{\epsilon}c}{N(e^{\epsilon}-1)^2}$，即每个估计频率值的方差。

选择 B 一般来说，B 用于平衡树高度和回答查询所需的节点数量。与此前的很多系列文章不同，AHEAD 在合并间隔的过程中引入了非均匀误差。与以前的研究所选择的 B 不同，考虑到非均匀性，文章将 B 设置为 2。

（四） 扩展到高维查询

1. 扩展到二维

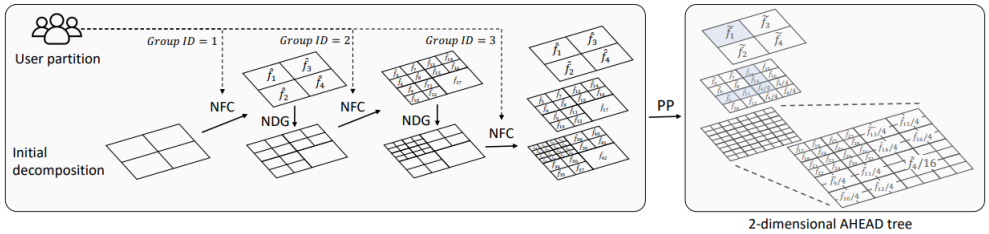

图 3: 扩展到高维查询

2 维区间查询的情况，与 1 维不同之处在于分解过程。使用 2 维网格来分解整个域，并同样包含四个步骤来捕捉用户的数据分布。

- 步骤 1：用户分组

  用户在范围 $[1, 2, 3, \cdots, c]$ 内随机选择组编号，其中 $c = \log_B |D|^2$。

- 步骤 2：带噪声频率构建

  然后，聚合器将整个域分成 B 个相等大小的正方形区域，并将初步分解结果发送给第一组用户。用户将他们的私有数据投影到初步分解中，并通过 OUE 报告他们的数据。服务器使用聚合算法获得估计频率分布，表示在每个子域中的用户比例。

- 步骤 3：新分解生成

  之后，聚合器将每个频率值与阈值 θ 进行比较，并决定是否进一步分解相应的子域。重复 NFC 和 NDG 过程，AHEAD 递归分解域并构建 AHEAD 原型树。

- 步骤 4：后处理

  最后，AHEAD 在每个层内进行非负性处理，用加权平均处理两个相邻层之间的误差，以进一步减小估计误差。基于均匀分布假设，AHEAD 获得一个完整的树来回答查询。

为了减少查询误差，AHEAD 更喜欢使用粗粒度节点来回答查询。例如，对于一个 2 维的查询 $[0, 5] \times [0, 5]$，AHEAD 从树的顶层到底层搜索并计算完全覆盖子域频率之和。该查询完全覆盖了顶层区域 $[0, 3] \times [0, 3]$ 和中间层的五个区域。这个例子在上图之中展示。

2. 扩展到更高维

AHEAD 可以通过两种方式扩展到更高维度：

- 直接估计（de——Direct Estimation）

  基于具有度 $B = 2^m$ 的树，AHEAD 同时分解 m 个维度。例如，AHEAD 将 3 维域处理为一个立方体，然后通过不同粒度的子立方体聚合频率来回答查询。使用前面提到的阈值设置，AHEAD 可以很好地控制子域的整体估计误差。但是，随着维度的增加，叶节点的数量呈指数级增长，这使得高维数据集的查询回答过程非常耗时。

- 利用低维估计（lle——Leveraging Low-dimensional Estimation）

为了解决数据维数增加引起的子域爆炸问题，lle 将属性成对组合，然后为每个属性对构建一个 2 维的 AHEAD 树。当回答一个 m 维查询时，lle 构造一个关联的 2^m 个查询集。然后，利用 2 维频率作为约束，lle 通过最大熵优化估计所有 2^m 个查询的频率值。

3. 具体步骤

- 步骤 1：构建块构造

  AHEAD 将所有属性成对分组以形成 C_m^2 个 2 维属性对。然后，AHEAD 分别估计这些 2 维属性对的频率分布，即总共有 C_m^2 个 2 维树。

- 步骤 2：属性一致性

  AHEAD 在相关的 m 个 2 维树中实现了所有 m 个属性的一致性。例如，属性 a 参与了 $(m-1)$ 个属性对的配对。假设这 $(m-1)$ 个 2 维树是 $\{T_1\ T_2\ T_3\ \cdots\ T_{m-1}\}$，且每棵树除了根节点以外都有 l 层。对于整数 $k \in [1, B^{l/2}]$，定义 $f_{T_i}(a,l,k)$ 为层 l 中 T_i 的节点频率总和，其对应于 a 的指定子域在 $[(k-1)\cdot\frac{|D|}{B^{l/2}}+1, k\cdot\frac{|D|}{B^{l/2}}]$。为使所有 $f_{T_i}(a,l,k)$ 一致，AHEAD 将它们的加权平均值计算为 $f(a,l,k)=\sum_{i=1}^{m-1}\lambda_i\cdot f_{T_i}(a,l,k)$，其中 λ_i 是 $f_{T_i}(a,l,k)$ 的权重，$\sum_{i=1}^{m-1}\lambda_i=1$。然后，我们应该选择权重 λ 的值来使 $f(a,l,k)$ 的方差最小化，即 $VAR[f(a,l,k)]=\sum_{i=1}^{m-1}\lambda_i^2\cdot VAR[f_{T_i}(a,l,k)]$，其中 $VAR[f_{T_i}(a,l,k)]$ 是用于计算 $f_{T_i}(a,l,k)$ 的节点的总方差。当权重与估计的方差成反比时，$VAR[f(a,l,k)]$ 实现了最小值。因此，最优权重为 $\lambda_i=\frac{1}{VAR[f_{T_i}(a,l,k)]}/\sum_{i=1}^{m-1}\frac{1}{VAR[f_{T_i}(a,l,k)]}$。我们应该通过添加改变量 $(f(a,l,k)-f_{T_i}(a,l,k))/B^{l/2}$ 来更新 T_i 中的每个相关节点，以使每个 $f_{T_i}(a,l,k)$ 等于 $f(a,l,k)$。一致性过程使 $\{T_1,T_2,T_3,\cdots,T_{m-1}\}$ 中的节点对属性 a 达成一致，而不会更改其他属性的频率分布。因此，在这些属性的任何顺序下，AHEAD 都可以实现所有 m 个属性的一致性。值得注意的是，每个 2 维树都有 l 层，因此，AHEAD 需要为所有 l 层进行上述过程。

- 步骤 3：最大熵优化

  我们面临的问题是在具有部分信息的 2 维查询情况下估计 m-dim 查询的频率。采用了最大熵原理。具体来说，对于一个 m-dim 范围查询 q，我们可以定义一组查询范围 $Q(q)$，其中

  $$Q(q)=\left\{\wedge\left(a_j,[\alpha_j,\beta_j]\ \text{ or }\ \overline{[\alpha_j,\beta_j]}\right)\mid a_j\in A\right\},$$

  其中 $[\alpha_t,\beta_t]$ 表示查询区间，$\overline{[\alpha_t,\beta_t]}$ 是它的补集。对于 2^m 个查询 $g\in Q(q)$，我们将 $f_q(g)$ 定义为查询 $Q(q)$ 中的查询的答案集。类似地，对于 2 维场景，我们可以获得查询集 $Q(q^{(j,k)})$ 和答案集 $f_{q^{(j,k)}}$：

  $$Q(q^{(j,k)})=\left\{\left(a_j,[\alpha_j,\beta_j]\ \text{ or }\ \overline{[\alpha_j,\beta_j]}\right)\wedge\left(a_k,[\alpha_k,\beta_k]\ \text{ or }\ \overline{[\alpha_k,\beta_k]}\right)\right\},$$

  并且对于任何查询 $g^{(j,k)}\in Q(q^{(j,k)})$，我们使用 $f_{q^{(j,k)}}(g^{(j,k)})$ 来表示其答案。特别地，对于 $g^{(j,k)}\in Q(q^{(j,k)})$，$f_q(g^{(j,k)})$ 表示从 f_q 构造 $g^{(j,k)}$ 答案的方式，即答案为 $Q(q)$ 中关联查询的答案求和。然后，我们可以制定以下优化问题：

  $$\begin{aligned}&\text{maximize} && -\textstyle\sum_{g\in Q(q)} f_q(g)\cdot\log\left(f_q(g)\right)\\ &\text{subject to} && \forall_{g\in Q(q)} f_q(g)\geq 0\\ & && \forall_{a_j,a_k\in A}\forall_{g^{(j,k)}\in Q\left(q^{(j,k)}\right)} f_{q^{(j,k)}}(g^{(j,k)})=f_q(g^{(j,k)})\end{aligned}$$

  上述优化问题可以通过现成的凸优化工具解决。为了更高效地解决频率估计问题，AHEAD 可以采用加权更新，其准确度几乎与最大熵相同。

五、 我的思考

(一) 论文优点

AHEAD 通过动态构建树结构，解决了现有 LDP 方法的局限性，显著提高了查询准确性，并可以促进未来基于 LDP 的隐私保护框架的发展。为克服在查找最优分区方面的障碍，AHEAD 利用理论上严格的 LDP 保证下推导出树扇出 B 和阈值 θ 来分解域。从实验结果来看，在低维和高维情况下这些参数设置都很好地工作。通过对各种隐私预算、域大小、用户规模、分布偏斜度、数据维度和属性相关性进行深入分析，得出了很多有用的结论。

(二) 缺陷不足

根据论文的内容，结合我个人的思考，论文提出的方法应该至少有以下一些潜在的问题：

第一，虽然 AHEAD 方法在低维和高维数据场景下都表现出色，但对于高维（>2）区间查询，AHEAD 对用户记录的规模敏感。AHEAD 需要将用户分组 2 次，即将其分成不同的属性组合和不同层数的 2 维树。因此，当用户组数较多时，**此时对于保证精度有更多开销**。AHEAD 需要充足的用户记录，以确保每个 2 维层的频率估计精度。

第二，AHEAD 生成具有各种粒度的 2 维区间来分解整个域。这带来了更大的内存和计算压力。也就是说，**AHEAD 方法可能面临组合爆炸的问题，即子域的数量可能会呈指数增长，导致计算和存储开销的增加**。因此，在实践中 AHEAD 需要更长的频率值搜索时间和更大的内存使用。

第三，AHEAD 是一个完全动态的框架，它使用阈值确定每个子域的划分。在域大小较大的情况下，每个子域都需要与阈值进行比较，这不可避免地增加了计算开销。**阈值确认计算开销都有更高的要求**。

第四，AHEAD 方法虽然采用了局部差分隐私机制来保护用户的隐私，但**其对于全局攻击的鲁棒性仍然有待改进**。全局敌手攻击是一种强大的攻击模型，攻击者可以利用多个查询结果来推断用户的隐私信息。由于 AHEAD 默认是 LDP 方法，因此有这种隐患。

第五，虽然 AHEAD 方法已经在多个数据集上进行了实验评估，但**仍然需要进一步验证其在实际应用中的可行性和效果**。比如，文中提出的一些高维查询的实验数量比较少。

(三) 可能的改进

首先，作者在文中提出可以为每个 2 维 AHEAD 树设计一种层融合策略以压缩树高度，同时潜在地采用“静态”和“动态”混合树结构（例如，前几层可以使用静态框架，而剩余层可以利用阈值按照动态框架分解子域）。这些方法都可以降低存储和计算消耗的问题。

我个人认为，其他未来可能的研究方向，可以考虑提高其在全局敌手攻击下的鲁棒性。同时，未来的研究可以探索更有效的子域划分策略，以降低计算和存储开销。例如，研究人员可以将 AHEAD 方法应用于真实的大规模数据集，并与现有的隐私保护和查询处理方法进行比较，评估其在实际应用中的性能和效果。

此外，可以探索将 AHEAD 方法与深度学习技术相结合，以进一步提高查询准确性和计算效率，比如通过深度学习或者机器学习的方法自动确定参数选择和子域划分的问题，也许会比文中所提出的方法有更好的表现。

参考文献

[1] Linkang Du, Zhikun Zhang, Shaojie Bai, Changchang Liu, Shouling Ji, Peng Cheng, and Jiming Chen. Ahead: adaptive hierarchical decomposition for range query under local differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1266–1288, 2021.

[2] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 729–745, 2017.

[3] Tianhao Wang, Milan Lopuhaä-Zwakenberg, Zitao Li, Boris Skoric, and Ninghui Li. Consistent and accurate frequency oracles under local differential privacy. *arXiv*, 2019.