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

Linkang Du
Zhejiang University
linkangd@gmail.com

Zhikun Zhang
CISPA Helmholtz Center for Information Security
zhikun.zhang@cispa.de

Shaojie Bai
Zhejiang University
white.shaojie@gmail.com

Changchang Liu
IBM Research
changchang.liu33@ibm.com

Shouling Ji*
Zhejiang University & Binjiang Institute of Zhejiang University
sji@zju.edu.cn

Peng Cheng*
Zhejiang University
saodiseng@gmail.com

Jiming Chen
Zhejiang University & Zhejiang University of Technology
cjm@zju.edu.cn

ABSTRACT

For protecting users' private data, local differential privacy (LDP) has been leveraged to provide the privacy-preserving range query, thus supporting further statistical analysis. However, existing LDP-based range query approaches are limited by their properties, *i.e.*, collecting user data according to a pre-defined structure. These static frameworks would incur excessive noise added to the aggregated data especially in the low privacy budget setting. In this work, we propose an Adaptive Hierarchical Decomposition (AHEAD) protocol, which adaptively and dynamically controls the built tree structure, so that the injected noise is well controlled for maintaining high utility. Furthermore, we derive a guideline for properly choosing parameters for AHEAD so that the overall utility can be consistently competitive while rigorously satisfying LDP. Leveraging multiple real and synthetic datasets, we extensively show the effectiveness of AHEAD in both low and high dimensional range query scenarios, as well as its advantages over the state-of-the-art methods. In addition, we provide a series of useful observations for deploying AHEAD in practice.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**.

KEYWORDS

Differential Privacy; Range Query; Adaptive Decomposition

*Peng Cheng and Shouling Ji are the co-corresponding authors.

CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3485668

ACM Reference Format:
Linkang Du, Zhikun Zhang, Shaojie Bai, Changchang Liu, Shouling Ji, Peng Cheng, and Jiming Chen. 2021. AHEAD: Adaptive Hierarchical Decomposition for Range Query under Local Differential Privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 23 pages. https://doi.org/10.1145/3460120.3485668

1 INTRODUCTION

With the increasing incidents of data breaches, such as Facebook [44], Marriott [47], Exactis [23], *etc.*, users' privacy has become a serious obstacle in many practical applications. As a promising countermeasure, differential privacy (DP) [16, 17] has been accepted as the *de facto* standard for protecting data privacy in academia and industry [18, 28, 32, 33, 41, 49, 74], due to its rigorous theoretical guarantees and independence of attacker's background knowledge. DP in the centralized setting requires a *trusted aggregator* that collects sensitive data from users and performs perturbation analysis, and then provides data services by answering queries or publishing synthetic data [22, 79].

When there is no trusted aggregator, DP in the centralized setting is no longer applicable and users are often reluctant to share their private data without protection. To address this obstacle, local differential privacy (LDP) [15, 55] is proposed, which allows individuals to encode and perturb their private data locally. In recent years, LDP has been deployed by many well-known leading companies, including Google [20, 21], Apple [13] and Microsoft [14]. For example, Google collects users' favorite homepages and Apple analyzes users' emoji preferences with LDP.

Previous studies [6, 7, 20, 59, 61] on LDP mainly focus on obtaining frequency distribution throughout the entire domain, *i.e.*, frequency oracle (FO) [65]. However, in practice, people may be more interested in a range query, *i.e.*, estimating the frequency in a certain range of a domain. For instance, supermarkets are interested to know the proportion of their high-income customers, e.g., earning between 100K to 120K dollars annually, to make commercial

policies. Furthermore, based on range query results, we can directly obtain other distribution features such as order statistics [48].

For range query, recent main-stream solutions can be divided into two categories by query dimension. For low(≤ 2)-dimensional query scenes, Wang *et al.* [62] proposed to hierarchically decompose the entire domain based on the complete B-ary tree structure and answer the range query by accumulating the frequency values, which was originally developed by Hay *et al.* [27] in the centralized setting. Cormode *et al.* [11] proposed to apply the discrete wavelet transformation (based on a full binary tree structure over the domain) to convert each user's private value to a Haar wavelet coefficient vector for perturbation and perform inverse transformation to get the query answer, as a generalization of [70] under the centralized DP setting. For high(≥ 2)-dimensional range query, Yang *et al.* [73] proposed to combine information from 1, 2-dimensional grids, which was originally proposed by Qardaji *et al.* [50] in the centralized setting, and leverage the weighted update strategy to estimate the high-dimensional range queries. However, the existing methods have several limitations. First, there exist sparse areas in the data domain of most real-world datasets. For instance, 50-60 years old people account for a small ratio among the members of a football club. Therefore, the nodes (cells) with small values in the complete tree (grid) are highly likely to be overwhelmed by the injected noises. In addition, existing techniques are mainly designed for specific dimensional queries, *i.e.*, [11, 62] for 1, 2-dim queries and [73] for high(≥ 2)-dimensional queries. Although [11, 62, 73] are not technically limited by query dimensions, they are less effective in the case of non-target dimensions. Since the dimensions of datasets are various in practice, the aggregator needs to combine the algorithms for different scenarios, thus limiting the adaptability and applicability of these algorithms.

To suppress the excessive injected noises, AHEAD provides a fine domain decomposition mechanism to accommodate the injected noise of nodes with various granularities in the tree. In order to enable AHEAD to find the proper domain decomposition, we carefully analyze the error source of the query answer obtained by AHEAD, and provide a guideline to obtain the decomposition. After AHEAD completes the interaction with all users, there exist certain constraints on nodes' values, *e.g.*, the sum of the children's values is equal to their parent's value. Thus, a post-processing method is designed for AHEAD to further boost the query accuracy. For high-dimensional queries, we compare two different expansion methods, *i.e.*, Direct Estimation (DE) and Leveraging Low-dimensional Estimation (LLE), and show the advantage of LLE based on experimental results.

To validate the effectiveness of AHEAD, we use multiple real and synthetic datasets to show the consistent advantage of AHEAD over the state-of-the-art methods. Specifically, on several real datasets, AHEAD can achieve significantly smaller estimation errors as compared to previous works by up to two orders of magnitude. For low dimensional scenarios, we evaluate various combinations of essential parameters, *e.g.*, privacy budget, domain size, user scale and distribution skewness, and then provide a comprehensive understanding of AHEAD by considering 757 parameter combinations in order to guide its adoption in practice. For high dimensional scenarios, we investigate the query accuracy of AHEAD in different data dimensions and attribute correlations, and show the characteristics of AHEAD and competitors based on experimental results.

Table 1: Summary of mathematical notations.

Notation	Description
N	The total number of users (user scale)
D	Private attribute domain
B	Tree fanout
m	The number of private attributes
ϵ	Privacy budget
θ	Threshold for intervals decomposition

In summary, the contributions of this paper are three-fold:

- We propose a dynamic algorithm for range query under LDP, which can adaptively determine the granularity of the domain composition. As compared to the state-of-the-art techniques, AHEAD can reduce the impact of the inserted noise to range queries for maintaining outstanding utility performance.
- We theoretically derive the parameter settings (decomposition threshold and tree fanout) for the consistent high utility under rigorous LDP guarantees. Furthermore, we extend our strategy to multi-dimensional scenarios.
- Through extensive experiments, we demonstrate the effectiveness of AHEAD on multiple real-world datasets as well as its advantages over previous approaches in balancing the utility and privacy tradeoff. In addition, we present six useful observations for deploying AHEAD in practical use.

2 BACKGROUND

2.1 Local Differential Privacy

In LDP, each user perturbs his/her private data v, through a perturbation mechanism Ψ, and then transmits $\Psi(v)$ to the aggregator while satisfying rigorous LDP guarantees defined as below.

Definition 2.1. ϵ–Local Differential Privacy (ϵ-LDP) [38]. A perturbation function $\Psi(\cdot)$ satisfies ϵ-LDP if and only if for $\epsilon > 0$ and all possible pairs of input $v_1, v_2 \in D$, we have

$$\forall T \in \text{Range}(\Psi) : \Pr\left[\Psi\left(v_1\right) \in T\right] \leq e^{\epsilon} \Pr\left[\Psi\left(v_2\right) \in T\right],$$

where $\text{Range}(\Psi)$ denotes the set of all possible outputs of Ψ.

2.2 Frequency Oracle

The frequency oracle (FO) protocol is used to estimate the frequency distribution F across a private attribute, serving as a basic building block for general LDP tasks such as marginal release [78] and range query [11, 62]. Most FO protocols consist of three steps: Encoding, Perturbation and Aggregation [61]. We introduce two state-of-the-art FO protocols in the following.

*2.2.1 Generalized Randomized Response (*GRR*).* The GRR algorithm is a generalized version of random response [68].

Encoding. GRR directly perturbs on private value v, thus the encoded value x_i equals to v_i for user i.

Perturbation. User i keeps v_i with probability $p = \frac{e^{\epsilon}}{e^{\epsilon}+|D|+1}$ and randomly chooses $v_i' \in D$ s.t. $v_i \neq v_i'$ with probability $q = \frac{1}{e^{\epsilon}+|D|+1}$, then uploads x_i' to the server, where $x_i' := \text{Perturb}(x_i)$.

Aggregation. The aggregator counts how many times v is reported, denoted by $\text{count}[v] = \sum_{i=1}^{N} \mathbb{I}_{\{x'=v\}}$. An unbiased estimation of the frequency of v is $\hat{f}_v = \frac{\text{count}[v]-Nq}{N(p-q)}$.

Estimation Error. $\hat{f}_v$ is an unbiased estimation of the true frequency f_v [62]. Therefore, the estimation error of GRR originates from the algorithm variance

$$\text{Var}_{\text{GRR}(\epsilon)} = \frac{|D| - 2 + e^{\epsilon}}{N\,(e^{\epsilon} - 1)^2} \tag{1}$$

*2.2.2 Optimized Unary Encoding (*OUE*).* OUE [61] is an optimization of the basic RAPPOR protocol in [20].

Encoding. User i encodes his/her private data into a one-hot binary vector, *i.e.*, $x_i = [0,\ 0\ \ldots,\ 1,\ \ldots,\ 0]$ of length $|D|$, where only the v_i-th position is 1.

Perturbation. User i flips each bit of x_i based on probabilities $p = \frac{1}{2}$ and $q = \frac{1}{e^{\epsilon}+1}$ as below, while transmitting 1's and 0's differently. 1's (resp., 0's) keeps with the probability of p (resp., $1-q$) and flips to the reverse with the probability of $1-p$ (resp., q). Then user i uploads x_i' to the server.

Aggregation. The aggregator collects $\{x_i'\}_{i=1}^{N}$ uploaded by the users and counts the number of occurrences of 1 in each bit, *e.g.*, for the v-th bit, $\text{count}[v] = \sum_{i=1}^{N} x_i'[v]$. The $\text{count}[v]$ needs to be corrected to obtain an unbiased estimation $\hat{f}[v] = \frac{\text{count}[v]-Nq}{N(p-q)}$.

Estimation Error. It is proved in [61] that OUE has variance

$$\text{Var}_{\text{OUE}(\epsilon)} = \frac{4e^{\epsilon}}{N\,(e^{\epsilon} - 1)^2} \tag{2}$$

Both GRR and OUE achieve unbiased estimation of frequency values. As shown in Equation 1 and Equation 2, OUE has a variance that is independent of $|D|$. For smaller $|D|$ (such that $|D| - 2 < 3e^{\epsilon}$), GRR is better; while OUE is superior for larger $|D|$.

3 PROBLEM DEFINITION AND EXISTING SOLUTIONS

3.1 Range Query Problem

	Age	Salary	Loan amount
v_1	18	150	0
v_2	42	5400	49192
v_3	27	2310	2194
...	...	...	...
v_N	69	3820	1982

Figure 1: An example database containing N users with three attributes: *age, salary* and *loan amount.*

Assume there are N users, where the i-th user has an m-dim ordinal record $\mathbf{v}_i^m = (v_i^{(1)}, v_i^{(2)}, \cdots, v_i^{(m)})$, with $v_i^{(j)}$ representing the j-th private attribute value owned by user i. Denote the domain for the j-th item as D_j. Given a series of ranges α_j, β_j ($j = 1, 2, \cdots, m$), an m-dim range query can be computed as

$$R_{\bigcap [\alpha_j,\beta_j]_{j=1}^m} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_{\bigcap \{\alpha_j \le v_i^{(j)} \le \beta_j\}_{j=1}^m}$$

where $\mathbb{I}_\gamma$ is an indicator function that takes 1 if the predicate γ is true and 0 otherwise. Figure 1 gives a running example of range query. For example, the proportion of people within 20 years to 40 years old constitutes a 1-dim range query, while the ratio of people within 20 years to 40 years old, with salary less than 5000, and with loan amount less than 20000 constitutes a 3-dim range query, where the three dimensions corresponding to age, salary and loan amount, respectively.

3.2 Hierarchical-Interval Optimized (HIO)

Based on a B-ary tree, HIO [62] hierarchically decomposes the entire domain into mutually disjoint subsets called *intervals*. The root node represents the entire domain, and the leaf nodes represent the individual values. Nodes on the same layer represent intervals of the same granularity. Then, HIO obtains the frequency estimations of nodes in each layer by the OUE [61] algorithm. When answering a range query, HIO completely covers the query range by using the minimum number of intervals from different layers.

For example, when the users' private attribute domain size $|D| = 8$ and tree fanout $B = 2$, the range query $[2, 7]$ can be decomposed into intervals $[2, 3] \cup [4, 7]$. Then, HIO adds the estimated frequency values of the two intervals above to get the answer of a range query. For general query with range length r, HIO can answer it with at most $2(B-1)\log_B |D|$ intervals. Compared with directly using FO mechanisms, HIO can effectively reduce the number of intervals used when answering queries, thus substantially reducing the cumulative error caused by adding noisy frequency values of intervals within the range.

However, HIO has two weaknesses that limit its applicability in practice. 1) HIO inserts the same level of noise into the estimated frequencies of all the intervals. For nodes with small intervals, the perturbation noise often overwhelms the true frequency values thus degrading the utility of the entire algorithm. 2) In the multi-dimensional scene, the number of tree layers increases exponentially with the number of dimensions. For high-dimensional scenarios, the query error increases extremely with the excessive small value nodes.

3.3 Discrete Haar Wavelet Transform (DHT)

DHT [11] imposes a full binary tree structure over the domain, and encodes the user private value v into a set of *Haar wavelet coefficients*. The motivation underlying DHT [11] is that the calculation of a length-r range query uses only a smaller number of estimated values in the Haar wavelet domain, comparing to apply FO directly.

DHT also faces several limitations. 1) Similar to HIO, DHT inserts the same level of noise into all estimated Haar wavelet coefficients. For some coefficients with low values, noise tends to skew the estimated coefficients causing query error to increase. 2) It is mainly designed for 1-dim scenario, thus limiting its application in practice.

3.4 Consistent Adaptive Local Marginal (CALM)

CALM [78] is a marginal release LDP protocol, which can construct the joint distribution of m attributes with privacy protection guarantee. Instead of directly estimating all marginal tables, CALM strategically chooses the size and the number of marginal tables, based on which all the marginal tables can be reconstructed. We

notice that CALM can be used to answer range queries. More specifically, to answer a multi-dimensional range query, CALM can sum up the reconstructed marginals included in the query.

However, when the domain size $|D|$ is large, CALM needs to sum up extensive noisy marginals to answer a query, which is likely to inject a large amount of noise to the true answer.

3.5 Hybird-Dimensional Grids (HDG)

HDG [73] is the state-of-the-art method to answer multi-dimensional range query under LDP. The main idea of HDG is to carefully bucketize the 2-dim domains of all attribute pairs into coarse 2-dim grids and then estimate the answer of a higher dimensional range query from the answers of the associated 2-dim range queries. To capture the fine-grained distribution information for users' data, HDG also introduces 1-dim grids to offer finer-grained distribution information on each attribute and combines information from 1-dim and 2-dim grids to answer range queries.

HDG also faces several limitations. 1) The equal granularity grids of HDG cannot handle various distributions of users' data. For skew-distributed datasets, whose data is concentrated in a small part of the whole domain, noise error or non-uniform error dominates in some grids thus degrading the utility. 2) Using 1-dim grids may destroy the correlation between attributes.

3.6 Remarks

To overcome the limitations of the state-of-the-art low-dimensional mechanisms (HIO, DHT) and high-dimensional mechanisms (CALM, HDG), we aim to achieve the following two design goals: 1) find a reasonable decomposition for the domain to avoid introducing excessive noise; 2) the designed mechanism can be extended to multi-dimensional scenarios, with better query accuracy than existing algorithms. Motivated by these goals, we propose AHEAD, which differs from the existing work in several major aspects: 1) AHEAD is an adaptive and dynamic algorithm, compared to the existing algorithms with static frameworks; 2) AHEAD reduces the impact of noise on small value nodes by merging intervals. 3) The designed mechanism can migrate from 1-dim to multi-dimensional scenarios. Next, we will illustrate the motivation and design of AHEAD in detail.

4 AHEAD: ADAPTIVE HIERARCHICAL DECOMPOSITION

4.1 Motivation and Overview

In this subsection, we use an example to illustrate the limitations of the existing algorithms and the rationality of AHEAD. As shown in Figure 2, the tables on the left show the intervals with corresponding real frequency values. For instance, the true frequency value of interval $[0, 1]$ is 0, meaning that there is no user data reside in this interval. Then, the middle tables display the frequency values of the intervals, separately estimated by different strategies. σ^2 represents the variance of the noise introduced in the perturbation process. The remaining part on the right shows the process of answering the query based on the estimated values.

Firstly, we focus on the process of the baseline strategy, such as HIO. The baseline strategy chooses to publish the estimated

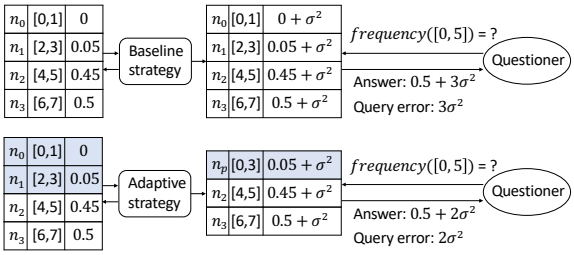


Figure 2: Baseline strategy vs. Adaptive strategy.

frequency of each interval. Each estimated value integrates a *noise error* by the FO mechanism to meet the LDP guarantees. For interval $[0, 1]$, its true value is 0, meaning that the estimated values for these intervals are completely filled with noise. When answering a range query, such as $frequency([0, 5])$, the questioner wants to know the frequency value of interval $[0, 5]$. The answer of the baseline strategy is $0.5 + 3\sigma^2$. It is worthy noting that interval $[0, 1]$ does not contribute to the query answer but bring the same degree of noise, which reduces the query accuracy of the existing algorithms.

On the other hand, for the adaptive strategy, as if we know the true frequency values of the intervals, we can combine the intervals n_0 and n_1 and estimate a single value for n_p. When answering the same range query $frequency([0, 5])$, the answer of the adaptive strategy is $0.5 + 2\sigma^2$, which reduces the noise error by 30% compared to the baseline method. We attenuate the noise error for the intervals with small frequency values, and the adaptive strategy works better in this case.

The combination of intervals will reduce the impact of noise on intervals with small frequency values. However, when a query falls within an interval, the answer has to be approximated by the assumption about the distribution within the interval. Making the *uniform distribution assumption* is a dominant strategy [29], where the value of each record in the interval is the same. When the assumption is not satisfied, it leads to a *non-uniform error*. For instance, if the questioner wants to know $frequency([2, 3])$, the answer of the adaptive strategy is calculated from the frequency value of interval $[0, 3]$, *i.e.*, the half of the frequency value of interval $[0, 3]$. Compared with the baseline strategy, the adaptive strategy reduces the noise error from σ^2 to $\frac{\sigma^2}{2}$, while it also brings non-uniform error $0.05 - \frac{0.05}{2} = 0.025$.

Adaptivity reduces the noise error by merging intervals, while introducing the non-uniform error by assuming uniformity. Therefore, to reduce the overall query error, we aim to find the optimal domain decomposition through balancing these two errors. However, finding the optimal partition for 2-dim datasets is diffcult [45], which is even worse with privacy constraint. Inspired by the above example, we propose a multi-phase hierarchy based recursive partitioning strategy (detailed in Section 4.2) that seeks to balance the errors and address the limitations of the existing solutions.

4.2 Workflow of AHEAD

In this subsection, we show the workflow of AHEAD with an example as shown in Figure 3. In this example, the aggregator wants to complete the range query task about the user's salary based on AHEAD. The salary data is bucketized into 8 ordinal levels, *i.e.*,

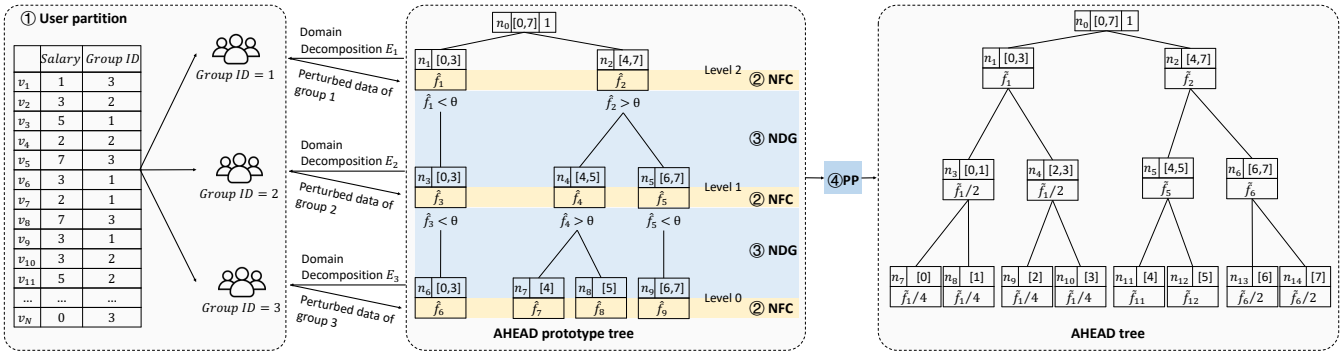


Figure 3: Workflow of AHEAD**. From left to right, the four steps in the** AHEAD **algorithm, *i.e.*, user partition, noisy frequency construction, new decomposition generation and post-processing, are shown respectively.** AHEAD **answers the range queries based on the tree in the rightmost sub-figure.**

domain size $|D| = 8$. The tree fanout $B = 2$, meaning that each node of the AHEAD tree has at most two child nodes. In each node of the AHEAD prototype tree, n_i represents the node index, $[a, b]$ represents the node's interval, and $\hat{f}_i$ represents the estimated frequency value. It is worth noting that AHEAD adopts the *sampling principle* [11], *i.e.*, partitioning users into groups with each group using the full privacy budget). *Sampling principle* can significantly reduce the overall error in local setting [46, 61, 64] (refer to more details in Appendix A). Below, we divide the workflow of AHEAD into four steps and describe the steps in detail.

Step 1: User Partition (UP). As shown in the left dashed box in Figure 3, the aggregator determines the number of partitions c, where $c = \log_B |D|$ is set to ensure that users are assigned to each layer of the AHEAD tree. The users randomly choose the group number in range $[1, 2, 3, \cdots, c]$. In addition, the users can also leverage their public information to select groups, such as the time of account registration, user ID, *etc.* The partition process should ensure that each group is representative of the overall population with a similar number of users.

Step 2: Noisy Frequency Construction (NFC). In the middle dashed box, the aggregator first establishes a root node n_0 representing the entire domain. After that, the aggregator performs the initial decomposition of the domain, *i.e.*, dividing the entire domain into B equal-sized intervals, then attaches the interval nodes to the root node n_0. The children of the root node represents a way to divide the whole domain, denoted as domain decomposition E_1. The aggregator selects the first group of users and sends the decomposition E_1 and privacy budget ϵ to them. Each user in the first group projects his/her private value v onto the intervals of E_1 and uploads the projected value of v via OUE. After receiving users' reports, the server uses the aggregation algorithm to obtain the estimated frequency distribution $\hat{F}_1$, which represents the ratio of users falling within each node's interval.

Step 3: New Decomposition Generation (NDG). The aggregator compares each frequency value of $\hat{F}_1 = \left\{\hat{f}_1, \hat{f}_2\right\}$ with a threshold θ and decides whether to divide the corresponding interval of $E_1 = \{[0, 3], [4, 7]\}$ further. To be specific, since $\hat{f}_2$ is greater than the setting θ, the corresponding interval $[4, 7]$ in E_1 should be divided into B equal-sized sub-intervals $[4, 5]$, $[6, 7]$. While the frequency value $\hat{f}_1$ of node n_1 is not greater than θ, interval $[0, 3]$ does not need further partition. For the new interval nodes, we attach them to the corresponding parent interval nodes. When all the elements in $\hat{F}_1$ are traversed completely, we can obtain a new set of intervals serving as decomposition E_2.

Then, the aggregator sends decomposition E_2 to the second group of users and obtains the estimated frequency distribution F_2. The aggregator repeats the above steps until all user groups are applied and gets an AHEAD prototype tree as shown in the middle dashed box of Figure 3. Since the estimated frequency is less than the threshold θ, AHEAD will not decompose the intervals $[0, 3]$ and $[6, 7]$ in subsequent interactions. To guarantee LDP, $[0, 3]$ and $[6, 7]$ should be estimated by all groups of users.

While constructing the prototype tree, AHEAD estimates each layer separately, which does not consider the constraint of frequency values in the tree, *i.e.*, the sum of the child nodes' frequency values is equal to that of their parent node. Therefore, in Step 4, we further boost the accuracy of AHEAD by conducting *non-negativity* and *weighted averaging* between the nodes' estimations.

Step 4: Post-processing (PP). The post-processing module contains two steps: *non-negativity* and *weighted averaging*.

Firstly, AHEAD processes the nodes in the same layer by Norm-Sub [65] to ensure that the estimated frequencies of nodes are non-negative and the sum of the frequencies is equal to 1. AHEAD converts the negative value into 0 and calculates the total difference between the sum of positive values and 1. Next, each positive value subtracts the average difference, which is obtained by dividing the total difference by the number of positive estimated values. The *non-negativity* process repeats until all values become non-negative.

Then, from bottom to top, AHEAD calculates the weighted average between non-leaf node n and its children to update the estimated frequencies of n, *i.e.*, reducing the added noise by fusing multiple estimations of n. For a non-root node n:

$$\tilde{f}(n) = \begin{cases} \lambda_1 \hat{f}(n) + \lambda_2 \sum_{u \in child(n)} \hat{f}(u), & \text{if u is a leaf node} \\ \lambda_1 \hat{f}(n) + \lambda_2 \sum_{u \in child(n)} \tilde{f}(u), & \text{o.w.} \end{cases} \tag{3}$$

The weights λ_1 and λ_2 are inversely proportional to the variance of the estimates, *i.e.*, $\lambda_1 = \frac{\mathrm{Var}_{child(n)}}{\mathrm{Var}_{child(n)} + \mathrm{Var}_{(n)}}$ and $\lambda_2 = \frac{\mathrm{Var}_{(n)}}{\mathrm{Var}_{child(n)} + \mathrm{Var}_{(n)}}$, where $\mathrm{Var}_{child(n)}$ represents the sum of node n's children variances,

and Var_n indicates the variance of node n. $\tilde{f}$ indicates the post-processed version of $\hat{f}$ and will be used to answer queries. The weighted average process can minimize the magnitude of noise as shown in the following theorem. The proof of Theorem 4.1 can be found in Appendix C.

Theorem 4.1. *Using Equation 3 to combine the frequencies of child nodes, the node n can achieve the minimal updated variance.*

Finally, from top to bottom, AHEAD decomposes the frequency value recursively under the uniform distribution assumption of the node's interval to obtain a complete tree (shown in the rightmost sub-figure of Figure 3), which will be used to answer range queries.

4.3 Privacy and Utility Analysis

Privacy Guarantee. AHEAD is sequentially interactive [2, 15, 35, 36, 38], *i.e.*, each user communicates once (Step 2 in Section 4.2) but the randomization depends on earlier user's messages (Step 3 in Section 4.2). Since the private data of each user is transmitted to the aggregator once via OUE with privacy budget ϵ (no other information of the users is leaked), we claim that AHEAD rigorously satisfies ϵ-LDP and the proof is deferred to Appendix B due to the space limitation.

Error Analysis. The overall error between the true query answer and the estimated answer originates from three sources of errors.

Noise and Sampling Errors originate from the OUE's perturbation and the user sampling processes. As shown in Section 2.2.2, although OUE can get an unbiased estimation of the frequency values, there is still an estimation variance caused by perturbation. In addition, AHEAD divides users into c groups and uses each user group to represent the frequency estimation from the entire population. Based on the analysis in [73], the sampling error is a constant which is much smaller than the inserted noise. Since each user randomly chooses one of the c groups to report private data, the population of each group approximates to $\frac{N}{c}$. By Equation 2, the variance of perturbed noise X is proportional to the number of groups c, *i.e.*, $\sigma^2 = c \cdot \frac{4e^\epsilon}{N(e^\epsilon-1)^2}$. Due to the threshold setting, some fine-grained intervals' frequency values may not be directly estimated. For the non-estimated intervals, their frequency values should be calculated from the larger intervals, *i.e.*, the higher-level intervals (such as parent nodes) in the AHEAD tree. If a non-estimated interval's frequency value is calculated from a larger interval whose size is k times of the non-estimated interval's size, AHEAD assigns $\frac{1}{k}$ of the large interval's value to the non-estimated interval. Thus, the noise error of the non-estimated interval can be viewed as originating from a random variable $\frac{X}{k}$.

Non-uniform Error arises from some intervals whose values are approximated by larger intervals' values in the AHEAD tree. For a non-estimated interval n whose true frequency value is f_n, the size of the larger interval is k times that of n (k is the same as that in the noise and sampling errors part). During the calculation, it is assumed that the private values in the larger interval satisfy a uniform distribution [50]. Thus, the assigned frequency value of the interval n is $\frac{f_p}{k}$, where f_p is the true frequency value of the larger interval. If the values in interval p satisfy the uniform distribution, there is no non-uniform error, *i.e.*, $f_n = \frac{f_p}{k}$. When the values in interval p do not meet the uniform distribution assumption, the deviation of the interval n's frequency value is $|f_n - \frac{f_p}{k}|$. Thus, the non-uniform error is influenced by the true distribution of the interval p. If the distribution is closed to the uniform distribution, the non-uniform error becomes small. Otherwise, the non-uniform error will increase, and the upper bound of the uniform error depends on the true frequency value f_p, *i.e.* $|f_p - \frac{f_p}{k}|$. For the entire domain, when the frequencies are more uniformly distributed across nodes, AHEAD behaves better owing to smaller non-uniform errors.

Algorithm 1 1-dim AHEAD Tree Construction

Input: All users' value set $V = \{v_1, v_2, \ldots, v_N\}$, attribute domain D, tree fanout B, privacy budget ϵ, threshold θ
Output: AHEAD Tree T

$c = \log_B|D|$
// Step 1: User partition
Randomly divide users into c parts $\{V_1, V_2, \ldots, V_c\}$
Create the root node of tree T with initial interval $e_0^0 = [1, |D|]$ and $T.\mathrm{node}(e_0^0).frequency = 1$
for i from 1 to c **do**
  // Step 2: New decomposition generation
  for j, node in enumerate($T.\mathrm{node}(level = i-1)$) **do**
    if node.frequency $> \theta$ **then**
      Divide interval e_j^{i-1} into B disjoint intervals $\{e_{j,k}^{i-1}\}$
      for k from 1 to B **do**
        node.add_child($e_{j,k}^{i-1}$)
      end for
    else
      node.add_child(e_j^{i-1})
    end if
  end for
  // Step 3: Noisy frequency construction
  $F = \mathrm{FO}(V_i, T.\mathrm{node}(level = i).\mathrm{interval}, \epsilon)$
  for k, node in enumerate $T.\mathrm{node}(level = i)$ **do**
    $node.\mathrm{frequency} = F[k]$
  end for
end for
// Step 4: Post-processing
Run Algorithm 2
Return T

4.4 Selection of B and θ

The most important parameters of AHEAD are tree fanout B and threshold θ. Since AHEAD has already rigorously satisfied LDP guarantees (recall Theorem B.1 in Appendix B), we aim to explore the settings of B and θ so that the overall utility performance of AHEAD can be maximized. Due to the partition strategy mentioned above and a large scale of users in actual scenarios ($N > 10^5$), we assume that each group has an equal number of users. Recalling the error analysis in Section 4.3, we focus on the noise error and non-uniform error, which dominate the overall estimation error.

Choosing θ. Intuitively, our goal in selecting the parameters of AHEAD is to balance the two errors, so that AHEAD can achieve an outstanding performance. For a set of parameters, *i.e.*, tree fanout

Algorithm 2 Post-processing

Input: AHEAD tree T, tree fanout B
Output: AHEAD tree T

```
for i from 1 to c do
    norm_sub(T. node(level = i) . frequency)
end for
for j from c − 1 to 1 do
    for _, node in enumerate T. node(level = j) do
        f1 = node.frequency, f2 = ∑ node. children(). frequency
        node.frequency = λ1 f1 + λ2 f2
    end for
end for
for k from 1 to c do
    for _, node in enumerate T. node(level = k) do
        if node.children() == None then:
            node . add_children()
            node . children() . frequency = node . frequency/B
        end if
    end for
end for
```

B, privacy budget ϵ, user scale N and the number of groups c, the decomposition threshold θ setting follows the formula below.

$$\theta = \sqrt{(B+1)\mathsf{Var}}, \tag{4}$$

where Var is equal to $\frac{4e^\epsilon c}{N(e^\epsilon-1)^2}$, *i.e.*, the variance of each estimated frequency value.

The analysis to support Equation 4 is as follows. Recalling the new decomposition generation step in Section 4.2, AHEAD divides each interval separately by comparing the estimated value with the threshold. Therefore, our analysis can focus on one of the interval nodes of the AHEAD tree. Suppose we have a node n with a true frequency value f and use $f_1, f_2, \cdots, f_B$ to denote the true frequency values of its children. Without loss of generality, one of n's children frequency value is ηf and the sum of others is $(1-\eta)f$, where $\eta \in [0,1]$. The parameter η is determined by the distribution of the users' data. When the distribution is far away from the uniform distribution, η closes to 0 or 1, which is the boundary of its value domain. Otherwise, η closes to $\frac{1}{B}$. Here, we consider two different strategies mentioned in Section 4.1 to estimate the frequency values of n's children. Firstly, we use the baseline strategy used in HIO to obtain their frequency values and the expected overall estimation error can be calculated below.

$$\begin{aligned}\mathbb{E}\left[\mathsf{Err}_1\right] &= \mathbb{E}\left[(\hat{f}_1 - f_1)^2 + (\hat{f}_2 - f_2)^2 + \cdots + (\hat{f}_B - f_B)^2\right] \\ &= \mathbb{E}\left[(f_1 + X_1 - f_1)^2 + \cdots + (f_B + X_B - f_B)^2\right] \\ &= \mathbb{E}\left[B \cdot X^2\right] = B\mathbb{E}\left[X^2\right] = B\mathsf{Var}\end{aligned} \tag{5}$$

In the derivation of Equation 5, OUE is conducted once to obtain users' data distribution over B intervals in each layer. X_i represents the perturbed noise added to the frequency of the i-th sub-interval. For each layer, the number of users in Equation 2 is the same for all intervals, *i.e.*, $\mathbb{E}[X_i^2] = O(c/N)$ for any i. Leveraging the adaptive strategy used in AHEAD, we estimate the frequency value of node n and assign the average as the child nodes' values. Then, we obtain the expected estimation error.

$$\begin{aligned}\mathbb{E}\left[\mathsf{Err}_2\right] &= \mathbb{E}\left[(\frac{\hat{f}}{B} - f_1)^2 + (\frac{\hat{f}}{B} - f_2)^2 + \cdots + (\frac{\hat{f}}{B} - f_B)^2\right] \\ &= \mathbb{E}\left[(\frac{f+X_1}{B} - \eta f)^2 + \cdots + (\frac{f+X_B}{B} - f_B)^2\right] \qquad (6) \\ &= \mathbb{E}\left[\eta^2 f^2 + \sum_{i=2}^{B} f_i^2 - \frac{f^2}{B}\right] + \mathbb{E}\left[\frac{X^2}{B}\right] \\ &\le \eta^2 f^2 + (1-\eta)^2 f^2 - \frac{f^2}{B} + \frac{1}{B}\mathsf{Var} \qquad (7) \\ &\le \frac{1}{B}((B-1)f^2 + \mathsf{Var}) \qquad (8)\end{aligned}$$

In the derivation of Equation 8, the error of each child node contains noise error and non-uniform error. For instance, the squared error of the first child in Equation 6 can be rewritten as $\left(\frac{X_1}{B} + \left(\frac{f}{B} - f_1\right)\right)^2$.

We let Equation 8 < Equation 5 to ensure that the adaptive strategy has a lower overall estimation error than that of the baseline strategy. Then we get the inequality as follows.

$$f < \sqrt{(B+1)\mathsf{Var}} \tag{9}$$

We further calculate the frequency values of the nodes on node n's subtree and the adaptive strategy always outranks the baseline strategy. Based on the analysis for Equation 9, if the frequency value of a node meets Equation 9, then it cannot be divided. Otherwise, it needs to be further divided. Therefore, we set the threshold $\theta = \sqrt{(B+1)\mathsf{Var}}$ to guarantee that AHEAD does not divide the nodes with small frequency values and reduces the estimation error than the baseline strategy.

It is worth noting that θ can be less optimal after the post-processing step (step 4 in Section 4.2). However, post-processing is correlated with the tree structure, which is unknown when choosing θ. Therefore, post-processing is hard to be incorporated in the theoretical analysis of θ. Our current analysis of θ is independent of the tree structure (and the input data as well), making the derived θ generally applicable to any input data. In the practical implementation of AHEAD, we only have access to the estimated frequency value $\hat{f}$, *i.e.*, the true frequency value f with a random noise variable X. Since OUE is an unbiased protocol, the expected value of $\hat{f}$ is equal to f. Thus, we still use θ as the threshold in AHEAD and provide a comprehensive validation of threshold choice in Appendix G.

Choosing B. In general, B is used to balance the tree height and the number of nodes required to answer the query. Previous studies [11, 62] select the optimal fanout B around 4 when only considering noise and sampling errors. Different from [11, 62], AHEAD introduces non-uniform error in the process of merging intervals. Compared to the B choice of previous studies, we set $B = 2$ considering non-uniformity and provide the analysis in the following.

For a node n with a true frequency f ($f < \theta$), AHEAD does not further decompose the interval of node n due to the threshold setting, where the children of node n can not directly get estimation frequency values in noisy frequency construction (Step 2 in Section 4.2). From Equation 2, the variance of the perturbed noise X on node n is proportional to the number of groups c, *i.e.*, $\sigma^2 = c \cdot \frac{4e^\epsilon}{N(e^\epsilon-1)^2}$. To obtain a complete tree for answering queries, AHEAD assigns $\frac{1}{B}$ of node n's estimation frequency value to its

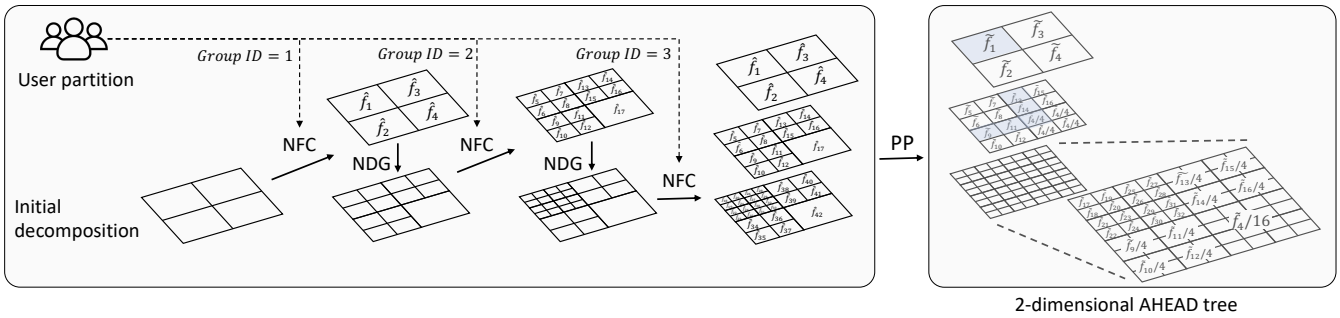


Figure 4: 2-dim AHEAD algorithm, where the decomposition is implemented by decomposing both dimensions simultaneously.

child nodes in post-processing (Step 4 in Section 4.2). Then, we can obtain the noise error of node n's child as $\frac{X}{B}$. Since we have no prior knowledge of the data distribution, for non-uniform error, we consider the worst case, *i.e.*, the non-uniform error is $f - \frac{f}{B}$. Considering f should be close to the threshold, the expected estimation error can be expressed as

$$\begin{aligned}\mathbb{E}\left[\mathrm{Err}_3\right] &= \mathbb{E}\left[(\frac{X}{B} + (f - \frac{f}{B}))^2\right] \\ &= \frac{c\sigma^2}{B} + (f - \frac{f}{B})^2 \\ &= \frac{c\sigma^2}{B} + (\frac{B-1}{B})^2(B+1)c\sigma^2 \\ &= (\sigma^2 \ln |D|)\frac{B + (B-1)^2(B+1)}{B^2 \ln B}, \end{aligned} \tag{10}$$

where $|D|$ is the domain size of attribute, ϵ is the privacy budget, N is the user scale and c is the number of user groups. Letting the derivative of Equation 10 to 0, we get $B = 0.6$ and $B = 2.2$. Since the tree fanout B is an integer greater than 1 and the value of Equation 10 is smaller at $B = 2$ than that at $B = 3$, we select $B = 2$ for AHEAD and compare the query error with $B = 4$ [11, 62]. We empirically validate the effectiveness of this parameter setting in Section 5.

4.5 Extension to Multi-dimensional Settings

2-dim Range Query. Let us first look at the 2-dim scenario. Without loss of generality, we assume that all attributes have the same domain $D = \{1, 2, \cdots, d\}$, where d is a power of the fanout B (if not in real setting, we can simply add some dummy values to achieve it). The main difference between 1-dim and 2-dim AHEAD is the decomposition process (lines 4, 9 of Algorithm 1). For 1-dim scenarios, AHEAD hierarchically divides the entire domain, which is an interval $[1, |D|]$, into different sub-intervals with varying granularity. While for 2-dim scenarios, the entire domain becomes a square area $[1, |D|] \times [1, |D|]$. AHEAD generates different granularity 2-dim grids to decompose the entire domain. The pseudo-code of 2-dim AHEAD can be found in Appendix D.

An example of 2-dim AHEAD is shown in Figure 4. The domain size of the user private attribute is 8×8 and the tree fanout $B = 4$. Similar to 1-dim scenarios, 2-dim AHEAD also contains four steps to capture the users' data distribution.

- **Step 1: User Partition (UP).** The users randomly choose the group number in range $[1, 2, 3, \cdots, c]$, where $c = \log_B |D|^2$.
- **Step 2: Noisy Frequency Construction (NFC).** Then, the aggregator divides the entire domain into B equal-size square areas and sends the initial decomposition of the domain to the first group of users. The users project their private data into the initial decomposition and reports their data through OUE. The server uses the aggregation algorithm to obtain the estimated frequency distribution, which represents the ratio of users falling within each sub-domain.
- **Step 3: New Decomposition Generation (NDG).** After that, the aggregator compares each frequency value with a threshold θ and decides whether to divide the corresponding sub-domain further. Repeating the NFC and the NDG processes, AHEAD recursively decomposes the domain and constructs the AHEAD prototype tree.
- **Step 4: Post-processing (PP).** Finally, AHEAD conducts the non-negativity process within each layer and the weighted averaging process between two adjacent layers to further reduce the estimated error. Based on uniform distribution assumption, AHEAD obtains a complete tree to answer queries.

As shown in the right part of Figure 4, In order to reduce the query error, which increases with the number of nodes used, AHEAD prefers to use coarse-grained nodes to answer a query. For example, AHEAD answers a 2-dim query $[0, 5] \times [0, 5]$. AHEAD searches from the top to the bottom layer of the tree and calculates the sum of the fully covered sub-domains' frequencies. The query completely covers the area of $[0, 3] \times [0, 3]$ of the top layer and five areas $[0, 1] \times [4, 5]$, $[2, 3] \times [4, 5]$, $[4, 5] \times [4, 5]$, $[4, 5] \times [0, 1]$, $[4, 5] \times [2, 3]$ of the middle layer (these areas are highlighted in blue), thus the query answer is $(\tilde{f}_1 + \tilde{f}_9 + \tilde{f}_{11} + \tilde{f}_{13} + \tilde{f}_{14} + \tilde{f}_4/4)$.

For the 2-dim range query, we set $B = 2^2$, *i.e.*, the B of each dimension is 2. As a comparison, we also provide the results of $B = 4^2$ in the experiment. Since the derivation of θ does not involve dimension changes, we still select θ according to Equation 4.

High-dimensional Range Query. AHEAD can be extended to higher dimensions in two ways.

- Direct Estimation (DE). Based on a tree with fanout $B = 2^m$, AHEAD decomposes m dimensions simultaneously. For instance, AHEAD treats the 3-dim domain as a cube, and then aggregates the frequencies of sub-cubes with different granularities to answer queries. With the threshold setting of Equation 4, AHEAD can well control the overall estimation errors of sub-domains. However, the number of leaf nodes increases exponentially with

dimension, which makes the query answer process extremely time-consuming on high-dimensional datasets.

- Leveraging Low-dimensional Estimation (LLE). To solve the sub-domain explosion caused by the rise of data dimension, LLE combines the attributes in pairs, and then constructs a 2-dim AHEAD tree for each attribute pair. When answering an m-dim query, LLE constructs a query set with the associated 2^m queries (recall Algorithm 6 in Appendix F). Then, taking the 2-dim frequency as constraints, LLE estimates the frequency values of all the 2^m queries by the maximum entropy optimization. For self-containment, we include its description and pseudo-code in Appendix F. Previous methods also used the maximum entropy optimization to effectively extend the low-dimensional mechanism to high-dimensional scenarios. PriView [52] proposes to construct low-dimensional views, *i.e.*, 2, 3-dim marginal tables, and apply maximum entropy optimization to reconstruct higher-way marginals from views. In this way, PriView constructs marginal tables for m-dim data with high data utility while satisfying DP. CALM [78] migrates the idea of PriView to LDP, which also achieves high accuracy for the problem of marginal release under LDP. HDG [73] groups all attributes in pairs and estimates the frequency distribution of user data on each attribute pair (2-dim grid). Then, HDG obtains the answer of a higher dimensional range query from the answers of the associated 2-dim range queries.

The implementation of the DE method is relatively simple, and the number of user partitions will not increase with the increase of the dimension. However, the AHEAD tree with DE might be very large in high-dimensional scenarios, making the tree construction and query answering process time-consuming. Compared to DE, the LLE method combines the attributes in pairs, and then constructs a 2-dim AHEAD tree for each attribute pair, which makes the scale of each tree not too large. We empirically show the performance of these two strategies and discuss how to choose between them in Section 5.4.

4.6 Discussion

AHEAD is similar to PrivTree [76], *i.e.*, a general approach for hierarchical decomposition on private data, where PrivTree also 1) generates tree T by recursively splitting a root node n_0 whose sub-domain covers the entire data domain D, and 2) decides whether a node n should be decomposed based on a noisy frequency of n. However, AHEAD differs from PrivTree in several important aspects. It is worth noting that these differences are mainly due to the fact that these two methods work under different privacy requirements. That is, PrivTree works in a centralized setting of differential privacy, while AHEAD works in a local setting.

- PrivTree can directly access all the information in the server, where PrivTree operates on the dataset, adds noise, and then derives the answers. AHEAD only has access to the noisy data uploaded by users, and then aggregates the reports to answer queries.
- In PrivTree, each frequency is estimated using the information of all users, with a split privacy budget. While in the local setting, partitioning users into groups and estimating the frequency with an entire privacy budget can obtain a higher data utility [11, 62, 78]. Therefore, in AHEAD, each frequency is estimated by only a subset of users, with an entire privacy budget.
- Based on the above two differences, besides *noise errors* in PrivTree, AHEAD further considers the *sampling* and *non-uniform errors*.
- In PrivTree, the tree fanout B is not considered in error analysis. In comparison, AHEAD derives the B setting through the analysis of *noise errors* and *non-uniform errors*.
- For high-dimensional scenes, PrivTree adopts direct estimation (DE) to extend low-dimensional strategies, while AHEAD leverages a more efficient way, *i.e.*, leveraging low-dimensional estimation (LLE).

Table 2: Summary of datasets.

Dataset	Distribution	Scale	Field	Type
Salaries	–	148,654	employee salary	real
BlackFriday	–	537,577	shopping	real
Loan	–	2,260,668	online loan	real
Financial	–	6,362,620	fraud detection	synthetic
Cauchy	Cauchy	–	–	synthetic
Zipf	Zipf (power-law)	–	–	synthetic
Gaussian	Gaussian	–	–	synthetic
Laplacian	Laplacian	–	–	synthetic

5 EVALUATION

To validate the effectiveness of AHEAD, we evaluate its performance on multiple real-world datasets and compare AHEAD with the state-of-the-art methods such as HIO [62] (1-dim query), DHT [11] (1-dim query), CALM [78] (1, 2-dim query) and HDG [73] (2-dim and high-dimensional query) in balancing utility and privacy. We also provide a detailed complexity analysis of the algorithms used in our evaluation in Appendix E.

5.1 Experimental Setup

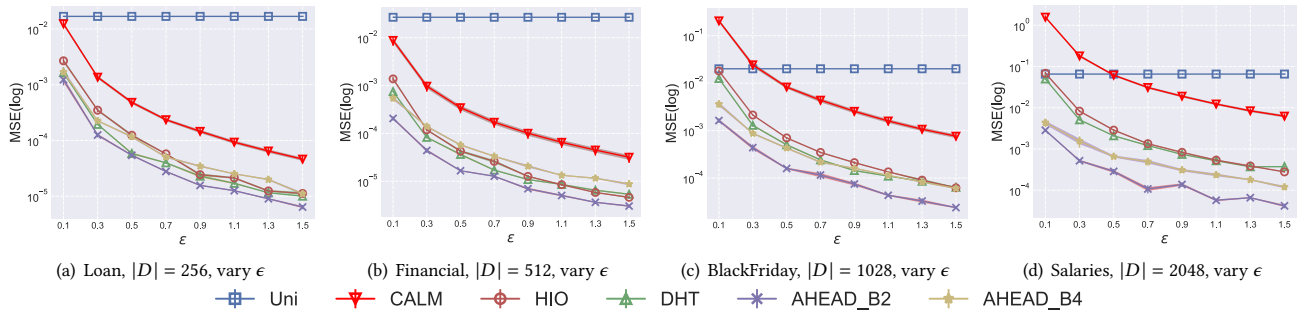
Environment. All our evaluations are conducted on a PC with Intel Xeon Platinum 8269@2.5GHz and 32GB memory.

Datasets. In our experiments, we use 3 real-world datasets and 5 synthetic datasets to evaluate the performance of AHEAD, and Table 2 provides an overview of all datasets. The detailed information about the four datasets, *i.e.*, Salaries, BlackFriday, Loan and Financial, is demonstrated in Appendix H.

We generate 1-dim datasets by sampling data from the *Cauchy* ($x_0 = 0, \gamma = 1$), *Zipf* ($\alpha = 1.1$), *Gaussian* ($\mu = 0, \sigma^2 = 1$) and *Laplacian* ($\mu = 0, \lambda^2 = 1/2$) distribution respectively, as prior works did [11, 73, 75]. The multi-dimensional datasets are synthesized from multivariate *Gaussian* and *Laplacian* distribution with mean 0, standard deviation 1 [73].

Metrics. To quantify the performance of AHEAD, we use the MSE (mean square error) widely used in literature [11, 65, 75] to measure the deviation between estimated and true values. For each experimental setting, we compute the MSE of 200 query results, and then compute the mean and std among 20 repetitions. In addition, we also provide a 95% confidence interval to reflect the deviations between MSEs.

Competitors. For a fair comparison, the HIO [62], DHT [11], CALM [78] and HDG [73] methods are applied with the same parameter settings as in the original papers. We also plot the MSE of



(a) Loan, $|D| = 256$, vary ϵ (b) Financial, $|D| = 512$, vary ϵ (c) BlackFriday, $|D| = 1028$, vary ϵ (d) Salaries, $|D| = 2048$, vary ϵ

Figure 5: The MSE of different methods with privacy budget varying from 0.1 to 1.5. The results are shown in log scale.

the Uniform method (denoted as Uni) in 1, 2-dim scenes, which always obtains the query answer from a uniform distribution. Clearly, if the performance of one method is worse than Uni, the query answer from that method is meaningless. For high-dimensional range query, we take HDG as the baseline method.

5.2 Evaluation for 1-dim Range Query

We evaluate the effectiveness of AHEAD under various values of privacy budgets. Specifically, we consider privacy budget ϵ varying from 0.1 (representing high privacy protection) to 1.5 (representing low privacy protection) [11, 78]. Figure 5 illustrates the MSE comparison between AHEAD and existing algorithms [11, 62, 78] for 1-dim queries. For each plot, we vary the privacy budget ϵ on the X-axis and show the corresponding MSE on the Y-axis. Each histogram in the plots shows the average MSE of 20 repeated experiments with the error bar representing the standard deviation.

From Figure 5, we have the following observations. 1) The MSE of AHEAD is smaller than its counterparts throughout the experiment datasets. Recalling Equation 9 in Section 4.4, for the nodes with frequency values less than the threshold, AHEAD with the adaptive strategy has a smaller overall estimation error than the counterparts that employ the baseline strategy. 2) AHEAD obtains different performance over various datasets. For the Salaries dataset, AHEAD achieves significant advantages over previous methods across the entire range of privacy budget, where the MSE of AHEAD is almost one order of magnitude smaller than state-of-the-art methods. For the Loan dataset, AHEAD is slightly better than DHT. We speculate that this is mainly because these two datasets have quite different distributions, such as more sub-domains with small frequency values (less than the threshold) on the Salaries dataset compared to the Loan dataset, which causes AHEAD to behave differently on these two datasets.

In addition, the performance of AHEAD varies under different datasets, which have various user scales, attribute domain sizes and data distributions. Therefore, we will conduct a comprehensive experiment in Appendix J to further analyze the performance of AHEAD under different experimental parameters and conclude important observations for its practical adoption.

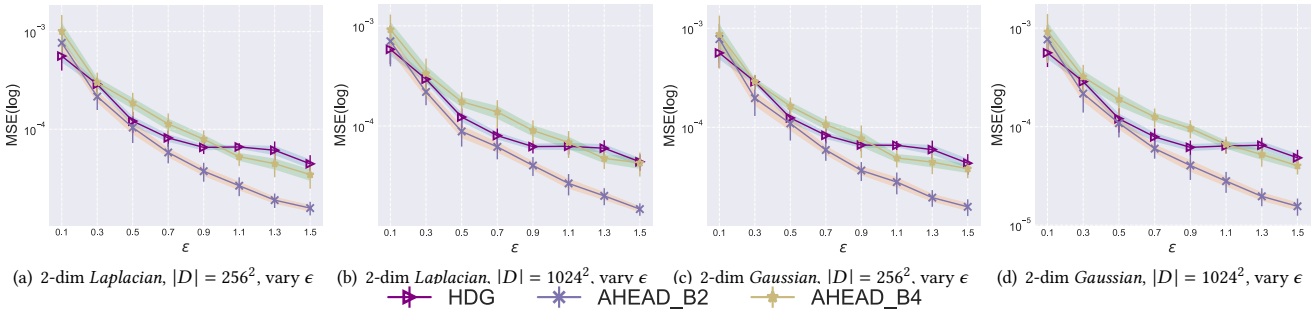
5.3 Evaluation for 2-dim Range Query

MSE under Various Privacy Budget. We evaluate the impact of varying privacy budget on the algorithms under 2-dim queries. Here, we focus on the synthetic datasets since they can reflect the algorithms' performance on the standard distribution and facilitate the adjustment of the dataset parameters. Each dataset contains 10^7 records, sampling from 2-dim *Laplacian* and *Gaussian* distributions respectively, with two domain sizes as 256×256 and 1024×1024.

Figure 6 shows the results over 2-dim *Laplacian* and *Gaussian* distributions respectively. We adopt correlation coefficient $r = 0.8$ the same with [73] between the two attributes. The MSEs of Uni and CALM are out of scale, thus their results are omitted here.

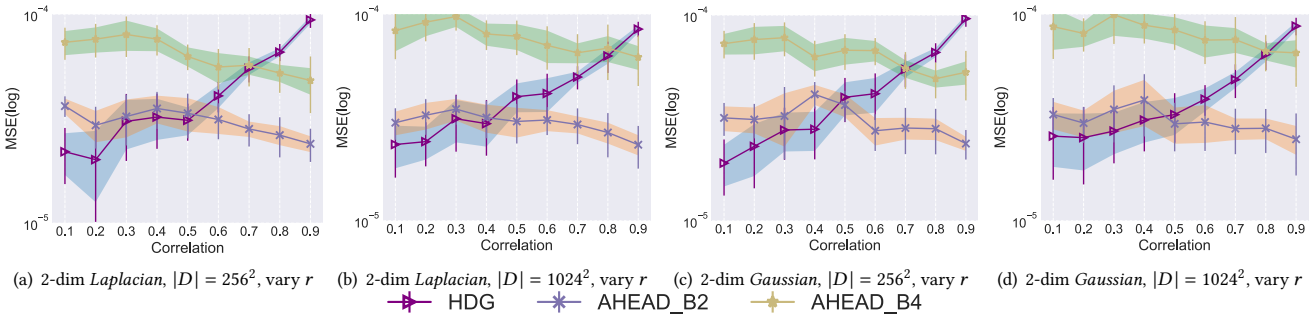
Based on the results, we have the following observations. 1) AHEAD outperforms HDG throughout the privacy budget settings. HDG leverages coarse 2-dim grids to bucketize the 2-dim domain, and captures the correlation information between two attributes. For a 2-dim 1024×1024 domain, the finest granularity of HDG is 8×8 in experiments. Using such coarse-grained grid results in the loss of some fine-grained data correlation. AHEAD establishes multiple granularity decompositions for sub-domains with different frequency values. For a sub-domain with high frequency value, which has a great impact on query answers, AHEAD uses fine-grained decomposition to estimate the frequency distribution within the sub-domain. Thus, the data correlation will be captured more accurately. 2) AHEAD is robust to the changes in domain size. On one hand, a larger domain size means that users need to be divided into more groups. For example, AHEAD partitions users into 8 groups for domain size 256×256, and 10 groups for domain size 1024×1024. There are fewer users in each group for a larger domain size, which will increase the added noise in the frequency values. On the other hand, the sub-domains whose frequencies are smaller than the threshold are not further divided from the layer where they first appear. These sub-domains will be estimated multiple times by user reports from different groups. After the weighted average process in post-processing, the noise errors of these sub-domains will become $\frac{1}{\beta}$ of the original noise errors, where β is the number of the estimation times. Comparing two adjacent subplots in Figure 6, the impact of the threshold setting is greater than that of domain size changes. Thus, the MSE of AHEAD almost does not vary across different domain sizes, and we select diverse domain sizes to further verify this fact in Section J.2.

MSE under Various Attribute Correlation. Then, we evaluate the impact of different attribute correlations on query errors as shown in Figure 7. For each subplot, we vary the correlation coefficient r from 0.1 (weakly correlated) to 0.9 (strongly correlated) with a fixed privacy budget $\epsilon = 1.1$. From the results, we have following findings. 1) The MSE of AHEAD almost does not change with different correlations. AHEAD decomposes both dimensions simultaneously, thus better protecting the correlation of the data.



(a) 2-dim *Laplacian*, $|D| = 256^2$, vary ϵ (b) 2-dim *Laplacian*, $|D| = 1024^2$, vary ϵ (c) 2-dim *Gaussian*, $|D| = 256^2$, vary ϵ (d) 2-dim *Gaussian*, $|D| = 1024^2$, vary ϵ

Figure 6: Comparison of different methods on 2-dim *Laplacian* and *Gaussian* datasets under various privacy budgets. We only plot the methods that are scalable in each setting. HDG is a baseline method. The results are shown in log scale.



(a) 2-dim *Laplacian*, $|D| = 256^2$, vary r (b) 2-dim *Laplacian*, $|D| = 1024^2$, vary r (c) 2-dim *Gaussian*, $|D| = 256^2$, vary r (d) 2-dim *Gaussian*, $|D| = 1024^2$, vary r

Figure 7: Comparison of different methods on 2-dim *Laplacian* and *Gaussian* datasets under various attribute correlation. The results are shown in log scale.

2) The data utility of HDG changes significantly with the correlation of attributes, and becomes worse with a stronger correlation. HDG combines finer-grained 1-dim grid to estimate the frequency distribution. If the correlation between two attributes is not very strong, the fine-grained 1-dim grid can complement the deficiencies of the coarse-grained 2-dim grid. Otherwise, the supplementary 1-dim information may be counterproductive. It is interesting to find that correlation $r = 0.5$ seems to be the intersection of HDG and AHEAD, which may guide the aggregator to select the better algorithm based on the correlation of attributes. For high-dimensional scenarios, we also evaluate the impact of different attribute correlations on query errors as shown in Section J.4, where AHEAD reacts similarly as 2-dim scenes.

5.4 Evaluation for High-dimensional Range Query

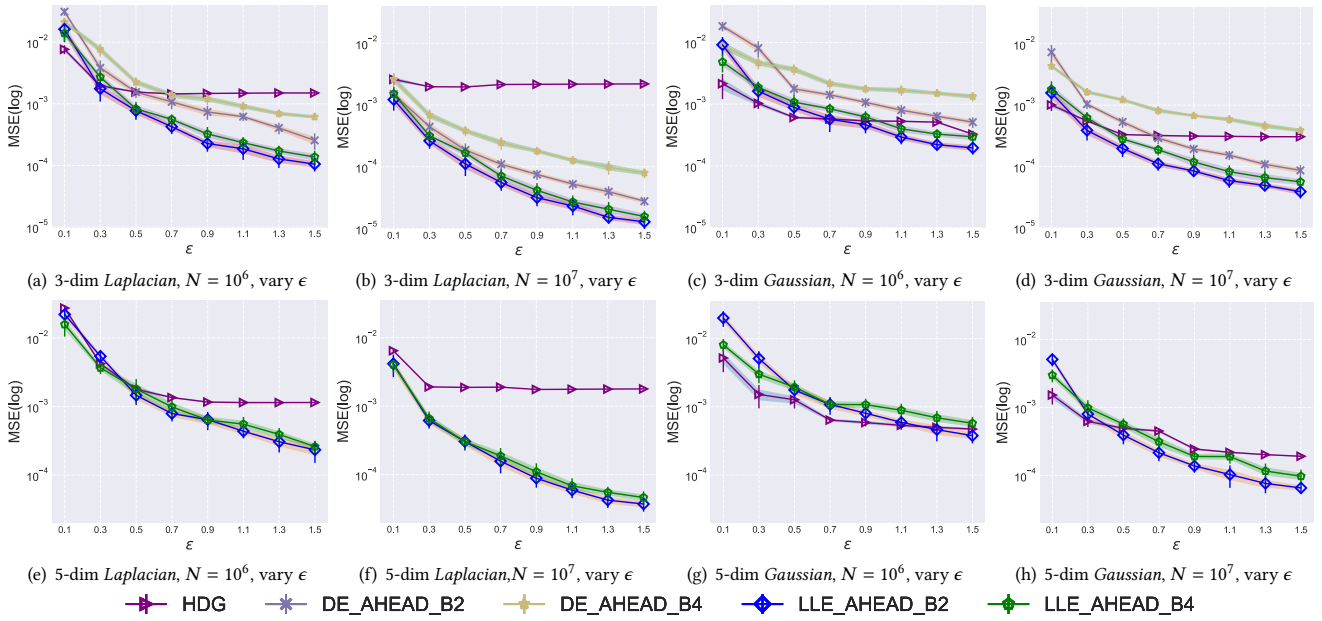
In this subsection, we evaluate AHEAD on high-dimensional private datasets. As observed in Figure 6 and Figure 7 (where the domain size is 256×256 and 1024×1024), MSEs of HDG and AHEAD are not sensitive to domain size. Therefore, we fix the domain size $|D| = 2^6$ and conduct the experiments on synthetic datasets with 10^6 and 10^7 records, which are sampled from high-dimensional *Laplacian* and *Gaussian* distributions, respectively. We refer the readers to Appendix I for the evaluation AHEAD on high-dimensional real-world datasets due to space limitation.

MSE under Two Expansion Methods. Recalling the two extension ways in Section 4.5, *i.e.*, DE and LLE, we compare the MSEs of the two expansion methods under the 3-dim range query. For queries higher than 3 dimensions, we only consider LLE since the DE method is too time-consuming.

From Figure 8, we observe that AHEAD with LLE obtains lower MSEs than DE. The sub-domains obtained by DE are equilateral high-dimensional cubes. AHEAD tends to use the underlying nodes, inducing more nodes used in the answering process. For example, when answering query $[1, 1] \times [1, 8] \times [1, 8]$ in a 3-dim dataset with domain $[1, 8] \times [1, 8] \times [1, 8]$, AHEAD selects the 64 leaf nodes at the bottom to answer the query instead of using higher-level nodes with larger sub-domains, which causes error accumulation in query answer. Although LLE needs to divide users and gather different attribute combination records from each user group, *i.e.*, holding fewer records to estimate the frequency for each layer of AHEAD trees compared to DE, the maximum entropy optimization step of LLE incorporates more information, thus resulting in smaller MSEs.

MSE under Various Privacy Budget. Figure 8 shows the results over 3-dim and 5-dim datasets, respectively. We adopt correlation coefficient $r = 0.8$ to quantify the correlation between attributes, following the same setting as [73].

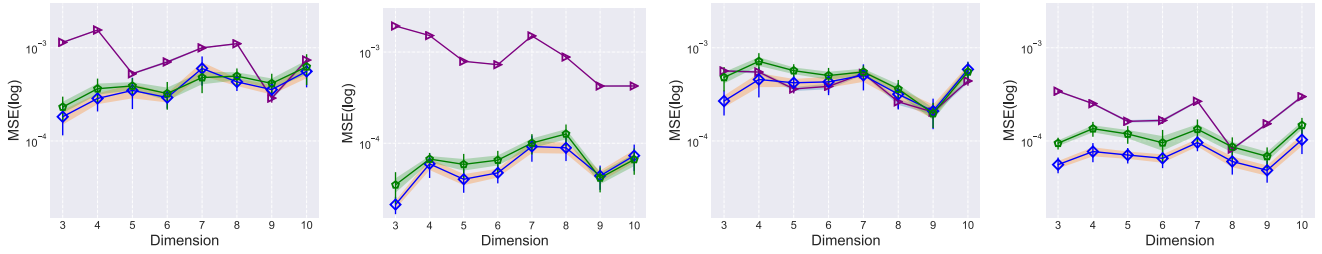
Based on the results, we have the following observations which are consistent with our analysis in Section 4.5. 1) AHEAD is robust to the changes in data distributions. Recalling the derivation of parameters θ and B in Section 4.4, AHEAD does not have specific requirements for the distribution of users' data. When the entire domain is adaptively divided, the same θ ensures that the frequencies of intervals have similar overall errors. Therefore, from Figure 8(b), Figure 8(d) (or Figure 8(f), Figure 8(h)), AHEAD behaves similarly on different datasets. Under the same dataset parameters, HDG uses the same granularity 1, 2-dim grids to aggregate user records, then answering queries based on the uniform assumption. Therefore, on the *Laplacian* distribution with more uneven frequency, the MSEs of HDG are larger than those on the *Gaussian* distribution. 2) For



(a) 3-dim *Laplacian*, $N = 10^6$, vary ϵ (b) 3-dim *Laplacian*, $N = 10^7$, vary ϵ (c) 3-dim *Gaussian*, $N = 10^6$, vary ϵ (d) 3-dim *Gaussian*, $N = 10^7$, vary ϵ

(e) 5-dim *Laplacian*, $N = 10^6$, vary ϵ (f) 5-dim *Laplacian*,$N = 10^7$, vary ϵ (g) 5-dim *Gaussian*, $N = 10^6$, vary ϵ (h) 5-dim *Gaussian*, $N = 10^7$, vary ϵ

Figure 8: Comparison of different methods on high-dimensional *Laplacian* and *Gaussian* datasets under various privacy budgets. DE and LLE respectively represent two high-dimensional expansion methods, *i.e.*, "direct estimation" and "leveraging low-dimensional estimation". HDG is a baseline method. The results are shown in log scale.



(a) *Laplacian*, $N = 10^6$, $|D| = 64^m$ (b) *Laplacian*, $N = 10^7$, $|D| = 64^m$ (c) *Gaussian*, $N = 10^6$, $|D| = 64^m$ (d) *Gaussian*, $N = 10^7$, $|D| = 64^m$

Figure 9: The MSE of different methods when varying data dimension m with $\epsilon = 1.1$. The results are shown in log scale.

high-dimensional queries, the performance of AHEAD is more dependent on the scale of user records. Compared with HDG which only needs to divide users into different attribute combinations, AHEAD needs to further part users into different layers in 2-dim trees (recall Algorithm 5 in Appendix F). For a 3-dim dataset with attribute domain size $|D| = 64$, AHEAD randomly divides the users into $C_3^2 \cdot 6$ groups, while HDG separates users into $C_3^2 + C_3^1$ groups, where the number of user records in each group of AHEAD is half of that of HDG, *i.e.*, doubling noise error of AHEAD. Therefore, from Figure 8(a) and 8(b), we know that the superiority of AHEAD will decrease with fewer user records.

MSE under Various Data Dimension. We evaluate the performance of AHEAD and HDG with varying dimensions of data (from 3-dim to 10-dim), which are sampled from multivariate *Laplacian* and *Gaussian* distribution with correlation coefficient $r = 0.8$.

From Figure 9, AHEAD is robust to the changes in data dimension. Recalling Section 4.5, with the data dimension m increasing, LLE needs to divide users into more groups due to more attribute combinations, *i.e.*, holding fewer user records to estimate the frequency for each 2-dim AHEAD tree. For instance, when $m = 3$, the number of attribute combination is $C_3^2 = 3$, and when $m = 10$, the number of attribute combination is $C_{10}^2 = 45$. Since LLE constructs a query set with associated 2^m queries (recall Algorithm 6 in Appendix F), the maximum entropy optimization step of LLE incorporates more query information with higher dimension, thus resulting in almost consistent MSEs with dimension increasing.

6 DISCUSSION

Highlights of AHEAD. 1) Through dynamically building the tree structure, AHEAD addresses the limitations of the state-of-the-art LDP methods, which significantly enhances the query accuracy and can motivate the development of future LDP based privacy-preserving frameworks. 2) To overcome the hindrance in finding the optimal partition [45], AHEAD decomposes the domain by leveraging the tree fanout B and the threshold θ, which are theoretically derived under rigorous LDP guarantees. From the experimental results, these parameter settings work well both in low and high

dimensional scenarios. 3) By conducting an in-depth analysis of AHEAD under various privacy budget, domain size, user scale, distribution skewness, data dimension and attribute correlation, we conclude some useful observations for adopting AHEAD (recall the observations in Appendix J).

Limitations and Future Work. Below, we discuss the limitations of AHEAD and promising directions for further improvements. 1) For high(>2)-dimensional range query, AHEAD is sensitive to the scale of user records. AHEAD needs to divide users 2 times, *i.e.*, partition into different attribute combinations and different layers in 2-dim trees. Thus, when the number of group user is large, AHEAD needs sufficient user records to ensure the accuracy of the frequency estimation of each 2-dim layer. 2) Compared to HDG using only two grids, *i.e.*, finer-grained 1-dim and sparser-grained 2-dim grids, AHEAD generates 2-dim intervals with various granularities to decompose the entire domain. Therefore, in practice, AHEAD requires longer frequency value searching time and larger memory usage compared to HDG. A layer fusion strategy can be designed for each 2-dim AHEAD tree to compress tree height. 3) AHEAD is a completely dynamic framework, which uses the threshold to determine the division of each sub-domain. In the cases of large domain size, each sub-domain needs to be compared with the threshold, which unavoidably increases the computational overhead. Since the higher-level node represents a relatively large sub-domain (whose frequency value is generally greater than the threshold), a 'static' and 'dynamic' hybrid tree structure can be potentially adopted. For instance, the top few layers can use the static framework, while the remaining layers can leverage the threshold to decompose the sub-domains in line with the dynamic framework.

7 RELATED WORK

Frequency Estimation under LDP. The notion of local differential privacy (LDP) was introduced in [38]. Duchi *et al.* [15] systematically analyzed the LDP algorithm and gave LDP's theoretical upper bound based on information theory. For LDP scenarios, one of the upmost basic tasks is frequency estimation of user values. Erlingsson *et al.* [20] proposed RAPPOR, which is the first practical example of frequency estimation. The algorithm uses the Bloom filter [7] to encode the private data, and then leverages a random response (RR) method [68] to perturb the encoded data. After that, several mechanisms [5, 6, 15, 68] were also proposed for frequency estimation under LDP. Wang *et al.* [61] compared the estimation variance of different algorithms and gave algorithm recommendations under different domain sizes. Wang *et al.* [60] proposed a wheel mechanism with a same variance as OUE [61].

Marginal Release under LDP. Marginal release are widely studied under LDP. Kulkarni *et al.* [10] proposed to apply the Fourier transformation method and Ren *et al.* [56] proposed to apply the expectation maximization method. The state-of-the-art method CALM [78], strategically choose sets of attributes and adaptively choose a randomization algorithm to reduce the noise effect. We notice that the methods for marginal release can be also used to answer range queries. Thus, we have also analysed CALM's performance in Section 3.4.

Range Query under DP. The range query problem has been studied extensively in the centralized setting including works based on hierarchy [12, 27, 40, 51], coarsened domain [39, 50, 71, 72, 77], Wavelet or Fourier transformation [3, 70], or publishing a synthetic dataset [25]. Hay *et al.* [26] proposed DPBench to evaluate algorithms for answering 1-dim and 2-dim range queries. McKenna *et al.* [43] presented HDMM to answer workloads of predicate counting queries. Both Ngram [8] and AHEAD adopt the adaptive tree strategies and leverage the decomposition threshold to balance utility and privacy. However, they are different in terms of the DP setting and targeted data type. 1) Ngram is designed and analyzed for DP scenarios, while AHEAD is orientated to the local setting. 2) Ngram publishes the sequential data with DP guarantee, while AHEAD aggregates one/multi-dimensional ordinal attributes satisfying LDP.

Range Query under LDP. For 1-dim scenarios, Cormode *et al.* [11] applied the Haar wavelet tranform to the LDP setting and proposed DHT. Wang *et al.* [62] leveraged the idea of hierarchical intervals and presented HIO mainly for answering 1-dim and 2-dim range queries. For answering 2-dim and high-dimensional range queries, Wang *et al.* [73] designed the state-of-the-art method HDG, which is inspired by the Adaptive Grids approach [50] under DP. There are some works that utilized an relaxation of ϵ-LDP [69] or leveraged the properties of workload [19] to achieve significant gains in utility of query answer.

Besides the above problems, DP (LDP) has been also used for other data analysis tasks, such as heavy hitters [5, 53, 58, 63], location [4, 9], graph data [30, 31, 54, 57], key-value data [24, 75], evolving data [37], machine learning [1, 34, 67]. Their problem definitions are different from ours, thus not suitable for comparison.

8 CONCLUSION

In this work, we propose a novel LDP protocol for the one and multi-dimensional range query problem, by leveraging adaptive hierarchical decomposition. Our method satisfies rigorous LDP guarantees while achieving advantageous utility performance with the theoretically-derived parameters. Through theoretical analysis as well as extensive experimental evaluation, we show the effectiveness of AHEAD in balancing utility and privacy for range queries and its significant advantages over the state-of-the-art methods. Furthermore, by studying various parameter settings, we conclude several important observations for adopting AHEAD in practice. Our source code is available on GitHub at https://github.com/linkzju/ccs21-AHEAD.

ACKNOWLEDGEMENT

We would like to thank Lisa M. Tolles from Sheridan Communications and the anonymous reviewers for their helpful comments. This work was partly supported by NSFC under Grants 62088101, 61833015, 61772466, U1836202, the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020003, and the Fundamental Research Funds for the Central Universities (Zhejiang University NGICS Platform). Changchang Liu was partly sponsored by the Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA).

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 308–318.

[2] Jayadev Acharya, Clément L Canonne, Yuhan Liu, Ziteng Sun, and Himanshu Tyagi. 2020. Interactive inference under information constraints. *arXiv preprint arXiv:2007.10976* (2020).

[3] Gergely Acs, Claude Castelluccia, and Rui Chen. 2012. Differentially private histogram publishing through lossy compression. In *2012 IEEE 12th International Conference on Data Mining*. IEEE, 1–10.

[4] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 901–914.

[5] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. 2017. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*. 2288–2296.

[6] Raef Bassily and Adam Smith. 2015. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 127–135.

[7] Burton H Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13, 7 (1970), 422–426.

[8] Rui Chen, Gergely Acs, and Claude Castelluccia. 2012. Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 638–649.

[9] Rui Chen, Haoran Li, A Kai Qin, Shiva Prasad Kasiviswanathan, and Hongxia Jin. 2016. Private spatial data aggregation in the local setting. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 289–300.

[10] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2018. Marginal release under local differential privacy. In *Proceedings of the 2018 International Conference on Management of Data*. 131–146.

[11] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2019. Answering range queries under local differential privacy. *Proceedings of the VLDB Endowment* 12, 10 (2019), 1126–1138.

[12] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. 2012. Differentially private spatial decompositions. In *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 20–31.

[13] Apple differential privacy team. [n.d.]. learning with privacy at scale. https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf. Accessed April 9, 2020.

[14] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*. 3571–3580.

[15] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2013. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 429–438.

[16] C Dwork. 2006. Differential Privacy In: Proc. of the 33rd International Colloquium on Automata, Languages and Programming (ICALP), 1–12.

[17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[18] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[19] Alexander Edmonds, Aleksandar Nikolov, and Jonathan Ullman. 2020. The power of factorization mechanisms in local and central differential privacy. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 425–438.

[20] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.

[21] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. 2016. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies* 2016, 3 (2016), 41–61.

[22] Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F Ilyas. 2020. Kamino: Constraint-Aware Differentially Private Data Synthesis. *arXiv preprint arXiv:2012.15713* (2020).

[23] Andy Greenberg. [n.d.]. Marketing Firm Exactis Leaked a Personal Info Database With 340 Million Records. [EB/OL]. https://www.wired.com/story/exactis-database-leak-340-million-records/ Accessed April 13, 2020.

[24] Xiaolan Gu, Ming Li, Yueqiang Cheng, Li Xiong, and Yang Cao. 2020. {PCKV}: Locally Differentially Private Correlated Key-Value Data Collection with Optimized Utility. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 967–984.

[25] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems*. 2339–2347.

[26] Michael Hay, Ashwin Machanavajjhala, Gerome Miklau, Yan Chen, and Dan Zhang. 2016. Principled evaluation of differentially private algorithms using dpbench. In *Proceedings of the 2016 International Conference on Management of Data*. 139–154.

[27] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 1021–1032.

[28] Jianping He, Lin Cai, and Xinping Guan. 2020. Differential Private Noise Adding Mechanism and Its Application on Consensus Algorithm. *IEEE Transactions on Signal Processing* 68 (2020), 4069–4082.

[29] Yannis Ioannidis. 2003. The history of histograms (abridged). In *Proceedings 2003 VLDB Conference*. Elsevier, 19–30.

[30] Shouling Ji, Weiqing Li, Neil Zhenqiang Gong, Prateek Mittal, and Raheem A Beyah. 2015. On Your Social Network De-anonymizablity: Quantification and Large Scale Evaluation with Seed Knowledge.. In *NDSS*.

[31] Shouling Ji, Weiqing Li, Prateek Mittal, Xin Hu, and Raheem Beyah. 2015. Secgraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*. 303–318.

[32] Shouling Ji, Weiqing Li, Mudhakar Srivatsa, and Raheem Beyah. 2014. Structural data de-anonymization: Quantification, practice, and implications. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1040–1053.

[33] Noah Johnson, Joseph P Near, and Dawn Song. 2017. Practical differential privacy for SQL queries using elastic sensitivity. *arXiv preprint arXiv:1706.09479* (2017).

[34] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. 2018. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*.

[35] Matthew Joseph, Jieming Mao, Seth Neel, and Aaron Roth. 2019. The role of interactivity in local differential privacy. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 94–105.

[36] Matthew Joseph, Jieming Mao, and Aaron Roth. 2020. Exponential separations in local differential privacy. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 515–527.

[37] Matthew Joseph, Aaron Roth, Jonathan Ullman, and Bo Waggoner. 2018. Local differential privacy for evolving data. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2381–2390.

[38] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.

[39] Chao Li, Michael Hay, Gerome Miklau, and Yue Wang. 2014. A data-and workload-aware algorithm for range queries under differential privacy. *Proceedings of the VLDB Endowment* 7, 5 (2014), 341–352.

[40] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. 2010. Optimizing linear counting queries under differential privacy. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 123–134.

[41] Ninghui Li, Min Lyu, Dong Su, and Weining Yang. 2016. Differential privacy: From theory to practice. *Synthesis Lectures on Information Security, Privacy, & Trust* 8, 4 (2016), 1–138.

[42] Edgar Lopez-Rojas, Ahmad Elmir, and Stefan Axelsson. 2016. PaySim: A financial mobile money simulator for fraud detection. In *28th European Modeling and Simulation Symposium, EMSS, Larnaca*. Dime University of Genoa, 249–255.

[43] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2018. Optimizing error of high-dimensional statistical queries under differential privacy. *arXiv preprint arXiv:1808.03537* (2018).

[44] Sheera Frenkel Mike Isaac. [n.d.]. Facebook Security Breach Exposes Accounts of 50 Million Users. [EB/OL]. https://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html Accessed April 13, 2020.

[45] Shan Muthukrishnan, Viswanath Poosala, and Torsten Suel. 1999. On rectangular partitionings in two dimensions: Algorithms, complexity and applications. In *International Conference on Database Theory*. Springer, 236–256.

[46] Thông T Nguyên, Xiaokui Xiao, Yin Yang, Siu Cheung Hui, Hyejin Shin, and Junbum Shin. 2016. Collecting and analyzing data from smart device users with local differential privacy. *arXiv preprint arXiv:1606.05053* (2016).

[47] Adam Satariano Nicole Perlroth, Amie Tsang. [n.d.]. Marriott Hacking Exposes Data of Up to 500 Million Guests. [EB/OL]. https://www.nytimes.com/2018/11/30/business/marriott-data-breach.html Accessed April 13, 2020.

[48] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 75–84.

[49] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2016. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755* (2016).

[50] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Differentially private grids for geospatial data. In *2013 IEEE 29th international conference on data engineering (ICDE)*. IEEE, 757–768.

[51] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding hierarchical methods for differentially private histograms. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1954–1965.
[52] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2014. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1435–1446.
[53] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2016. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 192–203.
[54] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2017. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 425–438.
[55] Sofya Raskhodnikova, Adam Smith, Homin K Lee, Kobbi Nissim, and Shiva Prasad Kasiviswanathan. 2008. What can we learn privately. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science*. 531–540.
[56] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A McCann, and S Yu Philip. 2018. LoPub: High-Dimensional Crowdsourced Data Publication With Local Differential Privacy. *IEEE Transactions on Information Forensics and Security* 13, 9 (2018), 2151–2166.
[57] Haipei Sun, Xiaokui Xiao, Issa Khalil, Yin Yang, Zhan Qin, Hui Wang, and Ting Yu. 2019. Analyzing subgraph statistics from extended local views with decentralized differential privacy. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 703–717.
[58] Ning Wang, Xiaokui Xiao, Yin Yang, Ta Duy Hoang, Hyejin Shin, Junbum Shin, and Ge Yu. 2018. PrivTrie: Effective frequent term discovery under local differential privacy. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 821–832.
[59] Shaowei Wang, Liusheng Huang, Pengzhan Wang, Hou Deng, Hongli Xu, and Wei Yang. 2016. Private weighted histogram aggregation in crowdsourcing. In *International Conference on Wireless Algorithms, Systems, and Applications*. Springer, 250–261.
[60] Shaowei Wang, Yuqiu Qian, Jiachun Du, Wei Yang, Liusheng Huang, and Hongli Xu. 2020. Set-valued data publication with local privacy: tight error bounds and efficient mechanisms. *Proceedings of the VLDB Endowment* 13, 8 (2020), 1234–1247.
[61] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally differentially private protocols for frequency estimation. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 729–745.
[62] Tianhao Wang, Bolin Ding, Jingren Zhou, Cheng Hong, Zhicong Huang, Ninghui Li, and Somesh Jha. 2019. Answering multi-dimensional analytical queries under local differential privacy. In *Proceedings of the 2019 International Conference on Management of Data*. 159–176.
[63] Tianhao Wang, Ninghui Li, and Somesh Jha. 2018. Locally differentially private frequent itemset mining. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 127–143.
[64] Tianhao Wang, Ninghui Li, and Somesh Jha. 2019. Locally differentially private heavy hitter identification. *IEEE Transactions on Dependable and Secure Computing* (2019).
[65] Tianhao Wang, Zitao Li, Ninghui Li, Milan Lopuhaä-Zwakenberg, and Boris Skoric. 2019. Consistent and accurate frequency oracles under local differential privacy. *arXiv preprint arXiv:1905.08320* (2019).
[66] Tianhao Wang, Milan Lopuhaä-Zwakenberg, Zitao Li, Boris Skoric, and Ninghui Li. [n.d.]. Locally Differentially Private Frequency Estimation with Consistency. ([n. d.]).
[67] Xin Wang, Hideaki Ishii, Linkang Du, Peng Cheng, and Jiming Chen. 2020. Privacy-preserving distributed machine learning via local randomization and ADMM perturbation. *IEEE Transactions on Signal Processing* 68 (2020), 4226–4241.
[68] Stanley L Warner. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *J. Amer. Statist. Assoc.* 60, 309 (1965), 63–69.
[69] Zhuolun Xiang, Bolin Ding, Xi He, and Jingren Zhou. 2019. Linear and Range Counting under Metric-based Local Differential Privacy. *arXiv* (2019), arXiv–1909.
[70] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2010. Differential privacy via wavelet transforms. *IEEE Transactions on knowledge and data engineering* 23, 8 (2010), 1200–1214.
[71] Yonghui Xiao, Li Xiong, Liyue Fan, and Slawomir Goryczka. 2012. DPCube: differentially private histogram release through multidimensional partitioning. *arXiv preprint arXiv:1202.5358* (2012).
[72] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, Ge Yu, and Marianne Winslett. 2013. Differentially private histogram publication. *The VLDB Journal* 22, 6 (2013), 797–822.
[73] Jianyu Yang, Tianhao Wang, Ninghui Li, Xiang Cheng, and Sen Su. 2020. Answering multi-dimensional range queries under local differential privacy. *Proceedings of the VLDB Endowment* 14, 3 (2020), 378–390.
[74] Lei Yang, Mengyuan Zhang, Shibo He, Ming Li, and Junshan Zhang. 2018. Crowd-empowered privacy-preserving data aggregation for mobile crowdsensing. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 151–160.
[75] Qingqing Ye, Haibo Hu, Xiaofeng Meng, and Huadi Zheng. 2019. PrivKV: Key-value data collection with local differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 317–331.
[76] Jun Zhang, Xiaokui Xiao, and Xing Xie. 2016. Privtree: A differentially private algorithm for hierarchical decompositions. In *Proceedings of the 2016 International Conference on Management of Data*. 155–170.
[77] Xiaojian Zhang, Rui Chen, Jianliang Xu, Xiaofeng Meng, and Yingtao Xie. 2014. Towards accurate histogram publication under differential privacy. In *Proceedings of the 2014 SIAM international conference on data mining*. SIAM, 587–595.
[78] Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, and Jiming Chen. 2018. Calm: Consistent adaptive local marginal for marginal release under local differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 212–229.
[79] Zhikun Zhang, Tianhao Wang, Ninghui Li, Jean Honorio, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. 2020. PrivSyn: Differentially Private Data Synthesis. *arXiv preprint arXiv:2012.15128* (2020).

A THE RATIONALITY OF SAMPLING PRINCIPLE

AHEAD can adopt two strategies to utilize the privacy budget: the privacy budget splitting strategy and the user partition strategy. More specifically, the privacy budget splitting strategy divides the whole privacy budget ϵ into c pieces and estimates the frequency distribution with all users' reports under privacy budget ϵ/c. The user partition strategy randomly assigns the users into c groups and uses the whole privacy budget to obtain the frequencies from each group of users. For the same node, the variances of the two strategies are recorded as Var_1 and Var_2. From Equation 2, we have $\mathrm{Var}_1 = \frac{4e^{\frac{\epsilon}{c}}}{N\left(e^{\frac{\epsilon}{c}}-1\right)^2}$ and $\mathrm{Var}_2 = \frac{c}{N}\frac{4e^{\epsilon}}{(e^{\epsilon}-1)^2}$. Since ϵ is greater than 0 and c is a positive integer greater than 1, we know $\mathrm{Var}_1 > \mathrm{Var}_2$. Therefore, under the same setting, the user partition strategy has a less noise error than the privacy budget splitting strategy.

For inconsistency of different levels, previous studies [46, 61, 64] have proven that the error of user partition is small. In addition, when answering queries, AHEAD uses the combination with the least number of nodes, so the inconsistency will not cause conflicts when answering queries.

B AHEAD SATISFIES ϵ-LDP

AHEAD can answer range query while satisfying rigorous LDP guarantees as shown in the following theorem.

Theorem B.1. AHEAD *satisfies ϵ-LDP.*

Proof. In Step 1, the users randomly select their group ID in $[1, 2, \cdots, c]$ without privacy budget consumption. In Step 2 and Step 3, AHEAD sequentially interacts with users, and each user produces a single output. In Step 4, AHEAD does not touch users' private data, thus incurring no additional privacy budget. Therefore, if the interaction with users (Step 2 and Step 3) meets the ϵ-LDP, AHEAD satisfies ϵ-LDP [36].

In each interactive round, AHEAD constructs the noisy frequency based on the OUE protocol with privacy budget ϵ in Step 2. For any pair of possible values $v_1, v_2 \in D$ belonging to the same user in the group g, a noisy binary vector O is every potential output in the

range of OUE.

$$\begin{aligned}\frac{\Pr[O|v_1,g]}{\Pr[O|v_2,g]} &= \frac{\Pi_{i=1}^{\ell}\Pr[O[i]|v_1,g]}{\Pi_{i=1}^{\ell}\Pr[O[i]|v_2,g]}\\ &= \frac{\Pi_{i=1}^{\ell}\Pr[O[i]|v_1]\cdot\Pr[g]}{\Pi_{i=1}^{\ell}\Pr[O[i]|v_2]\cdot\Pr[g]}\\ &\le \frac{\Pr[O[v_1]=1|v_1]\Pr[O[v_2]=0|v_1]}{\Pr[O[v_1]=1|v_2]\Pr[O[v_2]=0|v_2]}\\ &= \frac{p}{q}\cdot\frac{1-q}{1-p}\\ &= \frac{1/2}{1/(1+e^{\epsilon})}\cdot\frac{1-1/(1+e^{\epsilon})}{1-1/2} = e^{\epsilon}, \end{aligned} \tag{11}$$

where ℓ is the length of O. The p and q are the flipping probabilities of the OUE protocol. When $p=\frac{1}{2}$ and $q=\frac{1}{(e^{\epsilon}+1)}$, OUE can obtain the minimal variance [61]. From Equation 11, Step 2 satisfies ϵ-LDP. Since Step 3 processes uploaded noisy data, i.e., not using the users' original private data, there is no additional privacy budget consumption. Because of the above, the overall process satisfies ϵ-LDP. □

C PROOF OF THEOREM 4.1

Using Equation 3 to combine the frequencies of child nodes, the node n can achieve the minimal updated variance.

Proof. According to [66], when D is large and ϵ is not too large, each interval's estimated frequency value $\hat{f}$ is approximate to $f+X$, where f is the true frequency value and X is a random variable following *Gaussian* distribution $\mathcal{N}(0,\mathrm{Var}_{\mathrm{OUE}})$. If u is a leaf node, without loss of generality, we assume that

$$\hat{f}(n) = f(n) + \mathcal{N}(0,\mathrm{Var}_{(n)})$$

$$\sum_{u\in child(n)}\hat{f}(u) = \sum_{u\in child(n)} f(u) + \mathcal{N}(0,\mathrm{Var}_{(u)})$$

The interval of node n equals to the combination of its children intervals, thus

$$f(n) = \sum_{u\in child(n)} f(u)$$

Assuming the used coefficients for weighted average are λ_1 and λ_2, the weighted average between $f(n)$ and $\sum_{u\in child(n)} f(u)$ is an unbiased estimation of node n's frequency with $\lambda_1+\lambda_2=1$.

$$\tilde{f}(n) = \lambda_1\hat{f}(n) + \lambda_2\sum_{u\in child(n)}\hat{f}(u)$$

The variance of $\tilde{f}(n)$ is equal to $\lambda_1^2\mathrm{Var}_{(n)} + \lambda_2^2\mathrm{Var}_{child(n)}$. When

$$\lambda_1 = \frac{\mathrm{Var}_{child(n)}}{\mathrm{Var}_{child(n)}+\mathrm{Var}_{(n)}},\ \lambda_2 = \frac{\mathrm{Var}_{(n)}}{\mathrm{Var}_{child(n)}+\mathrm{Var}_{(n)}},$$

we can minimize the variance of $\tilde{f}(n)$. After the weighted average, the variances still fit *Gaussian* distribution. Thus the above proof still holds when u is not a leaf node. □

D THE PSEUDO-CODE OF 2-DIM AHEAD

As shown in Algorithm 3 and Algorithm 4, the main difference between 1-dim and 2-dim AHEAD is the decomposition process (lines 4, 9 of Algorithm 3).

Algorithm 3 2-dim AHEAD Tree Construction

Input: All users' value set $V=\{v_1,v_2,\ldots,v_N\}$, attribute domain D, tree fanout B, privacy budget ϵ, threshold θ
Output: AHEAD Tree T

1: $c = \log_B|D|$
2: // Step 1: User partition
3: Randomly divide users into c parts $\{V_1,V_2,\ldots,V_c\}$
4: Create the root node of tree T with initial interval $e_0^0=[1,|D|]\times[1,|D|]$ and $T.\mathrm{node}(e_0^0).frequency=1$
5: **for** i from 1 to c **do**
6: // Step 2: New decomposition generation
7: **for** j, node in enumerate($T.\mathrm{node}(level=i-1)$) **do**
8: **if** node.frequency $>\theta$ **then**
9: Divide interval e_j^{i-1} into B disjoint intervals $\{e_{j,k}^{i-1}\}$
10: **for** k from 1 to B **do**
11: node.add_child($e_{j,k}^{i-1}$)
12: **end for**
13: **else**
14: node.add_child(e_j^{i-1})
15: **end if**
16: **end for**
17: // Step 3: Noisy frequency construction
18: $F=\mathrm{FO}(V_i,T.\mathrm{node}(level=i).\mathrm{interval},\epsilon)$
19: **for** k, node in enumerate $T.\mathrm{node}(level=i)$ **do**
20: $node.\mathrm{frequency}=F[k]$
21: **end for**
22: **end for**
23: // Step 4: Post-processing
24: Run Algorithm 4
25: return T

Algorithm 4 Post-processing

Input: AHEAD tree T, tree fanout B
Output: AHEAD tree T

for i from 1 to c **do**
  norm_sub(T.node(level = i) . frequency)
end for
for j from $c-1$ to 1 **do**
  for _, node in enumerate $T.\mathrm{node}(level=j)$ **do**
    f_1 = node.frequency, $f_2=\sum$ node. children(). frequency
    node.frequency = $\lambda_1 f_1+\lambda_2 f_2$
  end for
end for
for k from 1 to c **do**
  for _, node in enumerate $T.\mathrm{node}(level=k)$ **do**
    if node.children() == None **then**:
      node . add_children()
      node . children() . frequency = node . frequency/B
    end if
  end for
end for

E COMPLEXITY ANALYSIS

Here, we provides a detailed complexity analysis of the algorithms used in our evaluation. For ease of exposition, we assume that all attributes have the same domain D.

Table 3: Comparison of complexity for different methods. The table lists the server-side computation, server-side storage, and client-server commutation.

		Time	Space	Comm
1-dim	AHEAD	$O(log_{\lvert D\rvert}2 \cdot N \cdot \lvert D\rvert)$	$O(\lvert D\rvert)$	$O(\lvert D\rvert)$
	CALM	$O(N \cdot \lvert D\rvert)$	$O(\lvert D\rvert)$	$O(\lvert D\rvert)$
	HIO	$O(log_{\lvert D\rvert}5 \cdot N \cdot \lvert D\rvert)$	$O(\lvert D\rvert)$	$O(log(\lvert D\rvert))$
	DHT	$O(N + \lvert D\rvert^3)$	$O(\lvert D\rvert^2)$	$O(log(\lvert D\rvert))$
2-dim	AHEAD	$O(log_{\lvert D\rvert}2 \cdot N \cdot \lvert D\rvert^2)$	$O(\lvert D\rvert^2)$	$O(\lvert D\rvert^2)$
	HDG	$O(N)$	$O(N^{\frac{1}{2}})$	$O(log\lvert D\rvert)$

Time Complexity. For 1-dim scenarios, the computation of AHEAD is mainly from processing users' reports, which takes $O(\frac{N}{log_2^{|D|}} \cdot 2^i)$ for the i-th group of users, *i.e.*, $O(log_{|D|}2 \cdot N \cdot |D|)$ in total. The similar arguments also hold for the hierarchy-based HIO method, *i.e.*, $O(log_{|D|}5 \cdot N \cdot |D|)$. Because the domain size D is usually large in the range query scene, CALM adopts OUE to aggregate the users' data, which takes $O(|D|)$ for each user, and $O(N \cdot |D|)$ in total. For DHT, the running time is dominated by the sum operation and inverse transformation. Specifically, the method first sums the reports of users with the same index, which should count all of them for each user, *i.e.*, $O(N)$. After that, DHT also needs an inverse transform process to produce the estimated frequency, *i.e.*, $O(|D|^3)$. For 2-dim scenarios, the domain size changes from $|D|$ to $|D| \times |D|$. AHEAD selects fanout $B = 4$ and the computation becomes $O(log_{|D|}4 \cdot N \cdot |D|^2)$. To estimate the user frequency distribution, HDG should evaluate hash functions for each report from users, *i.e.*, $O(N)$ in total.

Space Complexity. We measure the storage needed except that occupied by the inputs and outputs, which is the same amount of storage for all methods. For 1-dim scenarios, AHEAD and HIO needs to maintain the hierarchical tree structure, requiring $O(|D|)$ storage. The CALM method sums length-$|D|$ vector reported by each user to calculate the frequency distribution, which also needs $O(|D|)$ storage. Due to the inverse transform process, for DHT, a storage to count all possible intermediate results are needed, *i.e.*, $O(|D|^2)$. For 2-dim scenarios, the domain becomes $D \times D$, thus AHEAD require $O(|D|^2)$ storage to maintain the hierarchical tree structure. The storage of HDG depends on the granularity of 2-dim grids. Base on the analysis in [73], HDG needs $O(N^{\frac{1}{2}})$ storage.

Communication Complexity. Since HIO, DHT and HDG uses OLH [61] as FO, the report by OLH is only one bit, plus the index, which can be represented by $log|D|$ bits or $log|D|^2$ bits for 2-dim HDG. The CALM method adopts OUE, where each user should report a length-$|D|$ binary vector, requiring $O(|D|)$ bits. AHEAD is dominated by distributing the domain decomposition to each user. Considering the worst case, AHEAD needs $O(|D|^2)$.

Algorithm 5 Associated 2-dim AHEAD Tree Construction

Input: All users' value set $V = \{v_1, v_2, \ldots, v_N\}$, private attribute dimensions m, private attributes set A, attribute domain D, tree fanout B, privacy budget ϵ, threshold θ
Output: AHEAD Forest $\{T\}$

$c = C_m^2$
Randomly divide users into c parts $\{V_1, V_2, \ldots, V_c\}$
// Step 1: Building Block Construction
for k, $\{a_x, a_y\}_{x \neq y}$ in enumerate(pairwise attributes) **do**
  $\{T\}$.add(2dim_AHEAD_tree($V_k[a_x, a_y], D, B, \epsilon, \theta$))
end for
// Step 2: Consistency on Attributes
for ℓ from 1 to T.height **do**
  for attribute a_x in enumerate(A) **do**
    make $\{T.\text{node}(level = \ell).\text{frequency}\}$ consistent on a_x
  end for
end for
return $\{T\}$

Algorithm 6 Estimating Answer of m-dim Range Query

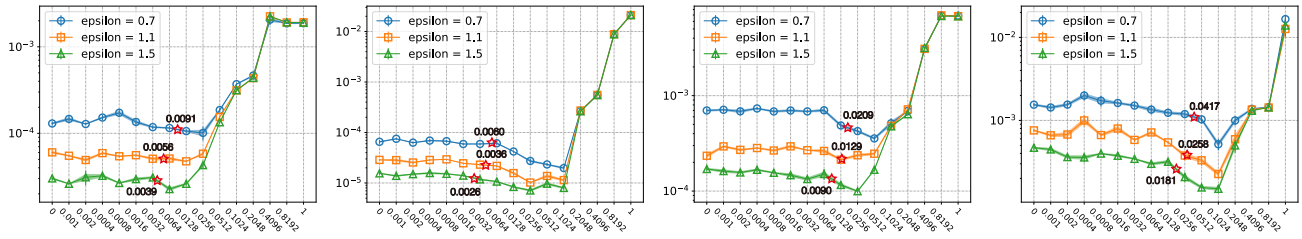
Input: AHEAD forest $\{T\}$, m-dim range query $R_{\bigcap [\alpha_j, \beta_j]_{j=1}^m}$
Output: answer of m-dim range query

// generate a set of m-dim range queries
$Q(q) = \left\{ \wedge \left(a_j, \left[\alpha_j, \beta_j\right] \text{ or } \overline{\left[\alpha_j, \beta_j\right]} \right) \mid a_j \in A \right\}$
// associated 2-dim queries' answers
for $\_, \{a_j, a_k\}_{j \neq k}$ in enumerate(pairwise attributes) **do**
  $Q(q^{(x,y)}) = \left\{ \wedge \left(a_j, \left[\alpha_j, \beta_j\right] \text{ or } \overline{\left[\alpha_j, \beta_j\right]} \right) \mid a_j \in (a_x, a_y) \right\}$
  $f_{q^{(x,y)}}(Q(q^{(x,y)})) = T^{(x,y)}.\text{frequency}(Q(q^{(x,y)}))$
end for
// Step 3: Maximum Entropy Optimization
maximize $-\sum_{g \in Q(q)} f_q(g) \cdot \log\left(f_q(g)\right)$
return $f_q(R_{\bigcap [\alpha_j, \beta_j]_{j=1}^m})$

F EXTENDING AHEAD TO HIGH-DIMENSIONAL RANGE QUERY

The extending process contains three steps as follows.

- **Step 1: Building Block Construction.** AHEAD groups all attributes in pairs to form C_m^2 2-dim attribute pairs. Then, AHEAD estimates the frequency distributions for the 2-dim attribute pairs separately, *i.e.*, a total of C_m^2 2-dim trees.
- **Step 2: Consistency on Attributes.** AHEAD achieves consistency on all m attributes among the related 2-dim trees. For example, the attribute a is involved in $(m-1)$ attribute pairs. Assume these $(m-1)$ 2-dim trees are $\{T_1, T_2, T_3, \cdots, T_{m-1}\}$ and each tree has ℓ layers except for the root node. For an integer $k \in [1, B^{\ell/2}]$, we define $f_{T_i}(a, \ell, k)$ to be the sum of frequencies of T_i nodes in level ℓ, whose specified sub-domain corresponds to a is in $[(k-1) \cdot \frac{|D|}{B^{\ell/2}} + 1, k \cdot \frac{|D|}{B^{\ell/2}}]$. To make all $f_{T_i}(a, \ell, k)$ consistent, AHEAD calculates their weighted average as $f(a, \ell, k) = \sum_{i=1}^{m-1} \lambda_i \cdot f_{T_i}(a, \ell, k)$, where λ_i is the weight of $f_{T_i}(a, \ell, k)$ and $\sum_{i=1}^{m-1} \lambda_i = 1$. Then, we should select the values of weight λ to minimize the variance of



(a) Loan, $|D| = 256$, vary θ (b) Financial, $|D| = 512$, vary θ (c) BlackFriday, $|D| = 512$, vary θ (d) Salaries, $|D| = 1024$, vary θ

Figure 10: Impact of threshold setting under various privacy budgets. The red star corresponds to our theoretical threshold obtained from Equation 4. The results are shown in log scale.

$f(a,\ell,k)$, *i.e.*, $\mathrm{Var}[f(a,\ell,k)] = \sum_{i=1}^{m-1} \lambda_i^2 \cdot \mathrm{Var}[f_{T_i}(a,\ell,k)]$, where $\mathrm{Var}[f_{T_i}(a,\ell,k)]$ is the total variance of the nodes used for the calculation of $f_{T_i}(a,\ell,k)$. Based on the analysis in [73], when the weights are inversely proportional to the variance of the estimates, $\mathrm{Var}[f(a,\ell,k)]$ achieves the minimum. Thus, the optimal weight $\lambda_i = \frac{1}{\mathrm{Var}[f_{T_i}(a,\ell,k)]} / \sum_{i=1}^{m-1} \frac{1}{\mathrm{Var}[f_{T_i}(a,\ell,k)]}$. We should update each involved node in T_i by adding the amount of change $(f(a,\ell,k) - f_{T_i}(a,\ell,k))/B^{\ell/2}$ to make each $f_{T_i}(a,\ell,k)$ equal to $f(a,\ell,k)$. Based on the analysis in [52], the consistency process makes $\{T_1, T_2, T_3, \cdots, T_{m-1}\}$ agree on attribute a without changing the frequency distributions of other attributes. Thus, following any order of these attributes, AHEAD can achieve consistency on all m attributes. Noting that each 2-dim tree has ℓ layers, AHEAD needs to conduct the above process for all ℓ layers.

- **Step 3: Maximum Entropy Optimization.** The problem we faced is to estimate the frequency of the m-dim query with partial information from 2-dim queries. We adopt the principle of Maximum Entropy [52]. Specifically, for an m-dim range query q, we can define a set of range queries as

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

where $[\alpha_t, \beta_t]$ represents the query interval and $\overline{[\alpha_t, \beta_t]}$ is the complement of it. For 2^m queries $g \in Q(q)$, we define $f_q(g)$ as the set of answers for queries in $Q(q)$. Similarly, for 2-dim scenarios, we can obtain the query set

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

and answer set $f_{q^{(j,k)}}$. For any query $g^{(j,k)} \in Q(q^{(j,k)})$, we use $f_{q^{(j,k)}}(g^{(j,k)})$ to denote its answer. In particular, for a $g^{(j,k)} \in Q(q^{(j,k)})$, $f_q(g^{(j,k)})$ means $g^{(j,k)}$'s answer constructed from f_q by summing up the answers of the associated queries in $Q(q)$. Then we can formulate the following optimization problem:

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

The above optimization problem can be addressed by an off-the-shelf convex optimization tool. To solve the frequency estimation problem more efficiently, AHEAD can adopt the Weighted Update [73], which achieves almost the same accuracy as the Maximum Entropy.

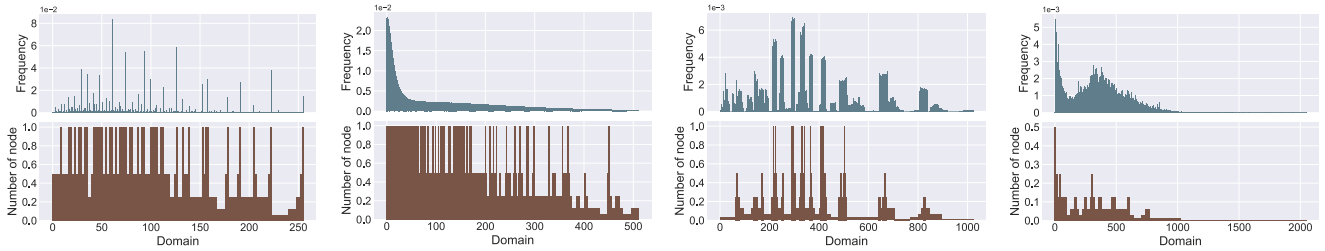
G VALIDATION OF THRESHOLD CHOICE

Recalling the analysis in Section 4.4, by setting a threshold, we do not divide the sub-domains whose frequencies are smaller than the threshold. A reasonable threshold setting can balance the noise error and non-uniform error thus minimizing the overall estimation error.

Here, we omit the post-processing step in AHEAD to highlight the effect of the threshold values. Figure 10 shows the impact of different threshold settings in AHEAD. The horizontal axis of each plot represents the threshold from 0 to 1. The red star on each line is the threshold value obtained by Equation 4 in Section 4.4.

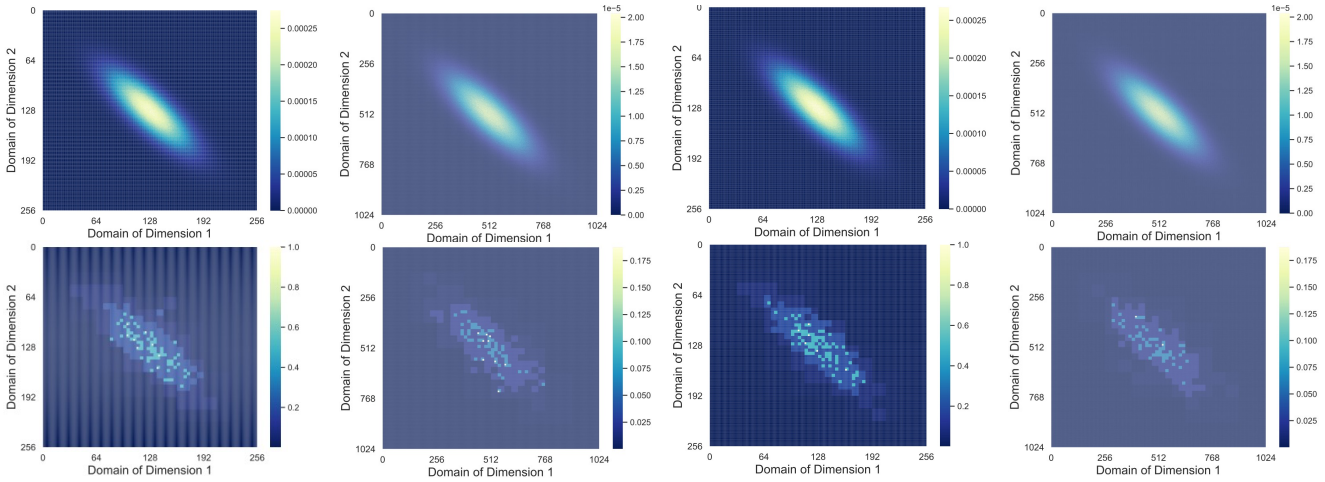
From Figure 10, we have the following observations consistent with our analysis. 1) A large threshold will cause the MSE to increase significantly. The reason is that a larger threshold would make the estimated frequency distribution closer to the uniform distribution. In this way, the non-uniform error will dominate the estimation error (recall Equation 8) thus degrading the overall accuracy. On the Loan dataset, when the threshold value is larger than 0.4096, the MSE becomes more than 2×10^{-3}, which is ten times of the minimum MSE. 2) A small threshold has little impact on MSE, *e.g.*, on the BlackFriday dataset, the MSE hardly changes when the threshold is less than 0.0064. In this case, the MSE is mainly caused by the noise error, which is related to the privacy budget. 3) On different datasets, the optimal experimental threshold values are also various. For example, on the BlackFriday dataset, the optimal experimental threshold is 0.0256 for $\epsilon = 1.5$. While on the Salaries dataset, the optimal experimental threshold is 0.1024. 4) It is worth noting that the theoretical θ is not exactly the empirically observed optimum in some cases. In the derivation of θ, we use the true frequency value f. However, in the practical implementation of AHEAD, we only have access to the estimated frequency value $\hat{f}$, *i.e.*, the true frequency value f with a random noise variable X. Since OUE is an unbiased protocol, the expected value of $\hat{f}$ is equal to f, making the theoretical θ close to the empirically observed optimum.

We also provide the distribution of the number of leaves in Figure 11 and Figure 12. The upper part of each sub-graph is the true frequency distribution, and the following is the corresponding node number distribution of AHEAD with $\epsilon = 1.1$. From Figure 11 and Figure 12, the node number distributions are almost the same with the true frequency distributions, which confirms the rationality of the threshold selection in Section 4.4. More specifically, when the frequency of the sub-domain is large, AHEAD further divides



(a) Loan, $|D| = 256$, $\theta = 0.006$ (b) Financial, $|D| = 512$, $\theta = 0.004$ (c) BlackFriday, $|D| = 1024$, $\theta = 0.013$ (d) Salaries, $|D| = 2048$, $\theta = 0.026$

Figure 11: The distribution of the number of leaves in AHEAD in 1-dim scenes. The top of each picture shows the true frequency distribution, and the bottom shows the corresponding distribution of the number of leaves with $\epsilon = 1.1$.



(a) *Laplacian*, $|D| = 256^2$, $\theta = 0.003$ (b) *Laplacian*, $|D| = 1024^2$, $\theta = 0.004$ (c) *Gaussian*, $|D| = 256^2$, $\theta = 0.003$ (d) *Gaussian*, $|D| = 1024^2$, $\theta = 0.004$

Figure 12: The distribution of the number of leaves in AHEAD in 2-dim scenes. The top of each picture shows the true frequency distribution, and the bottom shows the corresponding distribution of the number of leaves with $\epsilon = 1.1$.

the sub-domain, *i.e.*, more nodes for estimation, to reduce the non-uniform error. Otherwise, AHEAD does not decompose the sub-domain, *i.e.*, less nodes for estimation, to suppress the noise error. In addition, from Figure 5(d) and Figure 11(d), when the frequencies are more uniformly distributed across nodes in most subtrees (the nodes fell into the interval $[1024, 2048]$), AHEAD will behave better owing to smaller non-uniform errors (refer to more details in Section 4.3).

H DATASET DESCRIPTION

Table 4: Correlation between attributes of Salaries.

	a_1	a_2	a_3	a_4	a_5
a_1	1	-0.6638	-0.3631	0.2344	0.2344
a_2	-0.6638	1	0.0208	0.2442	0.2442
a_3	-0.3631	0.0208	1	0.4665	0.4665
a_4	0.2344	0.2442	0.4665	1	1
a_5	0.2344	0.2442	0.4665	1	1

Here, we provide a detailed description of the datasets used in our evaluation.

Table 5: Correlation between attributes of BlackFriday.

	a_1	a_2	a_3	a_4	a_5
a_1	1	-0.0508	-0.0030	-0.0062	0.0083
a_2	-0.0508	1	-0.0552	-0.4010	-0.3749
a_3	-0.0030	-0.0552	1	0.0774	0.0566
a_4	-0.0062	-0.4010	0.0774	1	0.3239
a_5	0.0083	-0.3749	0.0566	0.3239	1

Table 6: Correlation between attributes of Loan.

	a_1	a_2	a_3	a_4	a_5
a_1	1	0.0018	0.9431	0.2398	0.1519
a_2	0.0018	1	0.0460	0.0436	0.0481
a_3	0.9431	0.0460	1	0.2633	0.1651
a_4	0.2398	0.0436	0.2633	1	0.9612
a_5	0.1519	0.0481	0.1651	0.9612	1

- Salaries[1]: This dataset is about San Francisco city employee salary data on an annual basis from 2011 to 2014. It contains

[1]https://www.kaggle.com/kaggle/sf-salaries

148,654 records and 13 attributes, where we select 5 attributes as shown in Table 4.
- BlackFriday[2]: This dataset is a sample of the transaction records in a retail store, who wants to know the customer purchase behavior against different products. It contains 537,577 records and 12 attributes, where we select 5 attributes as shown in Table 5
- Loan[3]: This dataset provides the complete loan data of Lending Club for all loans issued through 2007-2018. It contains 2,260,668 records and 150 attributes, where we select 5 attributes as shown in Table 6
- Financial[4]: This synthetic dataset is generated by the PaySim mobile money simulator [42]. It contains 6,362,620 records and 11 attributes, including transaction type, customer ID and transaction amount. Since the distribution of transaction amount is quite skewed, we truncate the data greater than 500,000. After the truncation, the processed dataset has 6,022,336 records, *i.e.*, 94.6% of the original data remaining. Then, we divide the range of transaction amount [0, 500000] into slots of a fixed length and bucketize the records with a domain size of 512.

I HIGH-DIMENSIONAL RANGE QUERY ON REAL DATASETS

In this section, we evaluate AHEAD on high-dimensional real-world datasets, where the correlations between the selected attributes are shown in Table 4, Table 5 and Table 6. With domain size $|D| = 64$ for each dimension, we show the MSE of AHEAD for 2-dim, 3-dim and 5-dim range query, respectively. Under each setting, we set 8 different privacy budgets.

Base on the results in Figure 13, we have the following observations which are consistent with our analysis in Section 4.5 and Section 5.4. 1) AHEAD outperforms HDG throughout most cases. 2) AHEAD with LLE obtains lower MSEs than DE. 3) The data utility of HDG changes significantly with the correlation of attributes, and becomes worse with a stronger correlation. For instance, as shown in Figure 13(g), Figure 13(h) and Figure 13(i), the superiority of AHEAD will increase with the stronger correlation between attributes, *i.e.*, 'installment' in 3-dim and 'last_pymnt_amnt' in 5-dim (recall Table 6).

J PRACTICAL DEPLOYMENT OF AHEAD

In this section, we systematically analyze the impact of privacy budget, user scale, domain size, data skewness, data dimension and attribute correlation on the performance of AHEAD and the state-of-the-art methods. When focusing on one variable, we should ensure the others are consistent. Therefore, the experiments are mainly conducted on the four synthetic datasets. The observations in our following analysis can help adopt AHEAD in practice as well as assess existing LDP-based range query frameworks.

J.1 Impact of User Scale

Setup. We compare the algorithms' performance at increasing user scales, *i.e.*, from 10^3 to 10^7. Under each user scale N, we set 8 different privacy budgets. As shown in Figure 14, we use heatmaps to illustrate the impact of user scale and privacy budget on the MSE.

In practical use, the privacy budget is usually specified by users for satisfying their privacy requirements. Then, the aggregator needs to ensure query accuracy under the fixed privacy budget. For instance, when privacy budget $\epsilon = 1$, the aggregator desires MSE not higher than 10^{-2} to ensure the accuracy of the query results. Figure 14(a) and Figure 14(d) show the MSE of AHEAD over different coupling of user scale and privacy budget. To meet the accuracy demand, the aggregator needs to collect at least $5 \cdot 10^3$ user records with AHEAD. In comparison, previous works require more user records (more than 10^4) for satisfying the same level of accuracy. For stronger privacy protection, *i.e.*, smaller privacy budget, all methods require more user records. Based on the above observations, we summarize the following observation for selecting a proper user scale.

Observation 1 (Necessity of choosing proper user scale). *When the privacy protection strength is fixed, i.e., a determined privacy budget ϵ, one needs to use an appropriate user scale to ensure algorithm performance.*

Next, we take a step further to analyze the relation between user scale and privacy budget in order to provide easy methods for selecting user scale. As shown in Figure 14(a) and Figure 14(d), AHEAD has similar MSEs under different combinations of user scale and privacy budget. When user scale $N_1 = 10^6$ and privacy budget $\epsilon_1 = 0.1$, the MSE of AHEAD is $10^{-3.0809}$ on the *Zipf* dataset. In addition, AHEAD obtains similar MSE ($10^{-3.0563}$) with $N_1 = 10^4$ and $\epsilon_2 = 1$. Based on the results in Figure 14(b), Figure 14(e), Figure 14(c) and Figure 14(f), HIO and DHT also have the user scale & privacy budget exchangeability. All the three evaluated methods leverage the FO protocol when collecting user private data. From Equation 2, the variance of OUE is $\frac{4e^{\epsilon}}{N(e^{\epsilon}-1)^2}$. The Var can be converted to $\frac{4}{N(e^{\epsilon}+e^{-\epsilon}-2)}$, which is approximate to $\frac{4}{N\epsilon^2}$ when ϵ is not large ($\epsilon \leq 2$). Then, the variance of the estimated frequency is approximately the same if the product $N\epsilon^2$ of two combinations is equal. Therefore, we have the following observation.

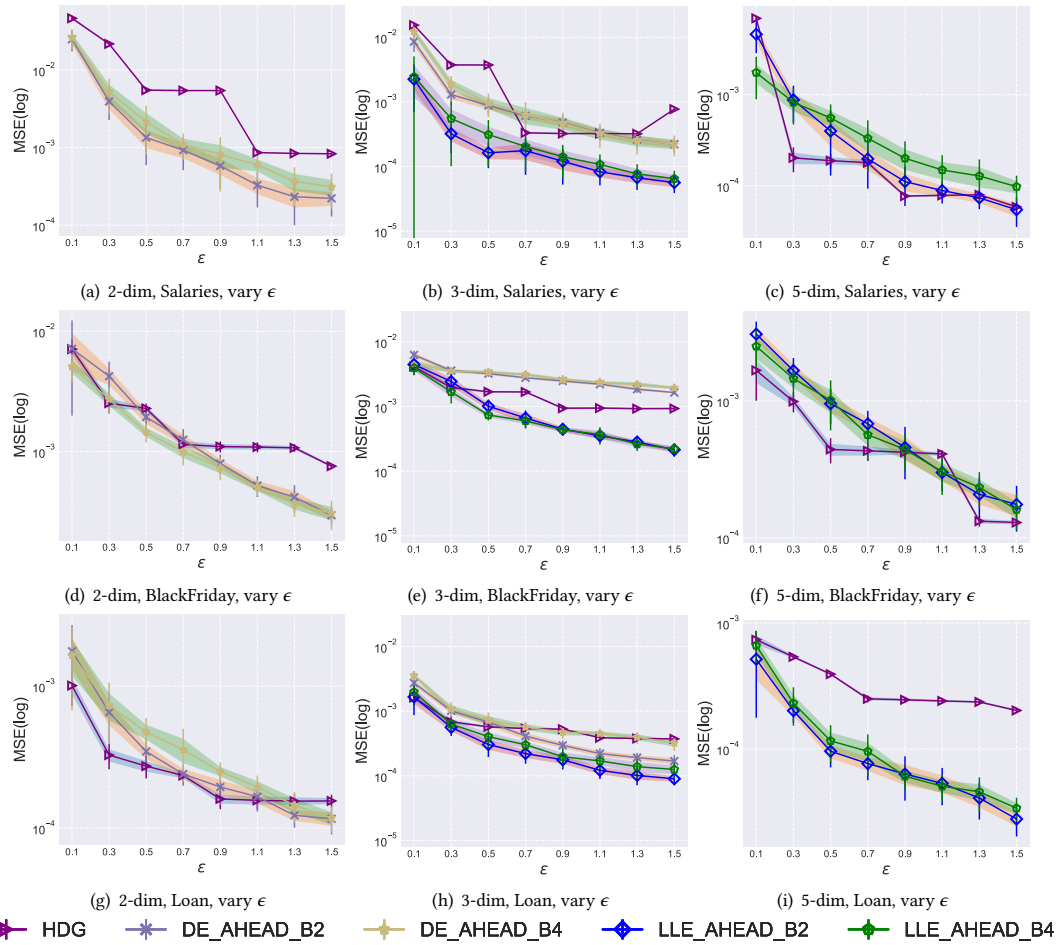
Observation 2 (The exchangeability between user scale and privacy budget). *For any two pairs of user scale and privacy budget combination (N_1, ϵ_1) and (N_2, ϵ_2), when ϵ is not large and $N_1\epsilon_1^2 \approx N_2\epsilon_2^2$ is satisfied,* AHEAD *can achieve a similar MSE under two pairs (N_1, ϵ_1) and (N_2, ϵ_2).*

J.2 Impact of Domain Size

Setup. There are 8 different domain sizes used in the experiments. In addition, we set 7 different privacy budgets to explore the coupling effect of domain size $|D|$ and privacy budget ϵ on MSE.

1) From Figure 15(b), Figure 15(c), Figure 15(e) and Figure 15(f), the MSEs of HIO and DHT have a tendency to increase with the

[2]https://www.kaggle.com/roshansharma/black-friday
[3]https://www.kaggle.com/wordsforthewise/lending-club
[4]https://www.kaggle.com/ntnu-testimon/paysim1

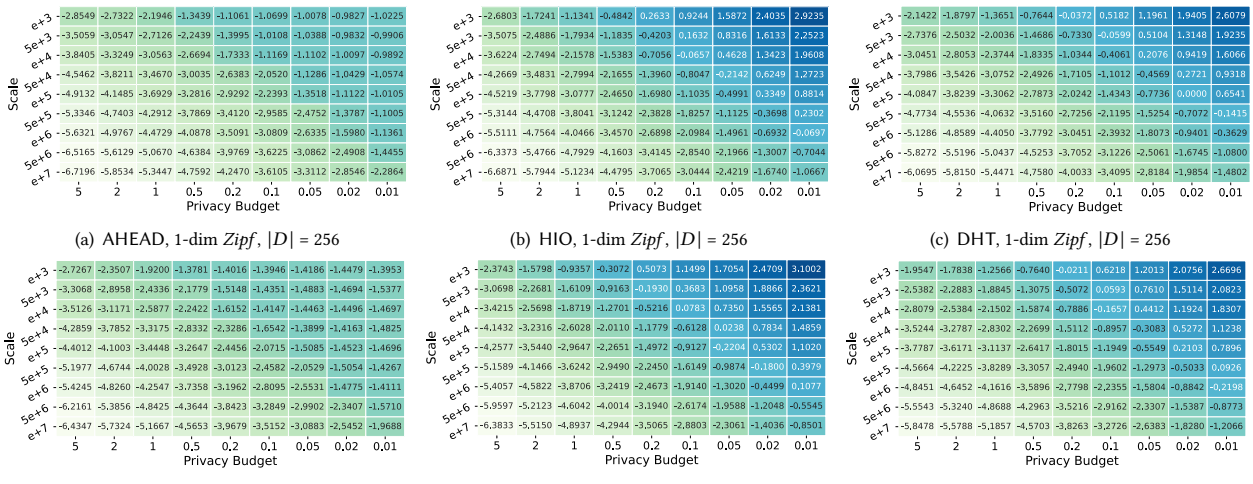


(a) 2-dim, Salaries, vary ϵ (b) 3-dim, Salaries, vary ϵ (c) 5-dim, Salaries, vary ϵ

(d) 2-dim, BlackFriday, vary ϵ (e) 3-dim, BlackFriday, vary ϵ (f) 5-dim, BlackFriday, vary ϵ

(g) 2-dim, Loan, vary ϵ (h) 3-dim, Loan, vary ϵ (i) 5-dim, Loan, vary ϵ

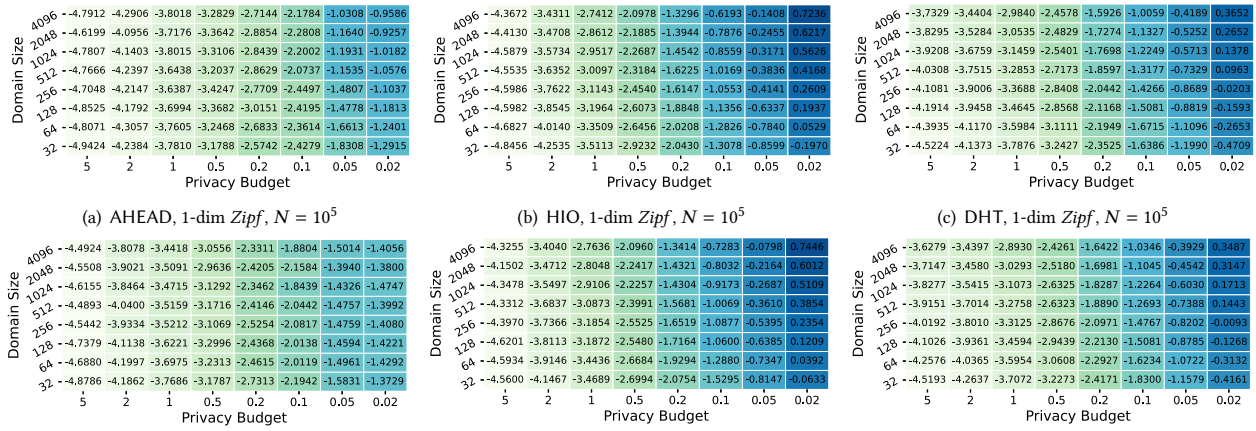
Figure 13: Comparison of different methods on high-dimensional real datasets under various privacy budgets. DE **and** LLE **respectively represent two high-dimensional expansion methods,** ***i.e.,*** **"direct estimation" and "leveraging low-dimensional estimation".** HDG **is a baseline method. The results are shown in log scale.**



(a) AHEAD, 1-dim *Zipf*, $|D| = 256$ (b) HIO, 1-dim *Zipf*, $|D| = 256$ (c) DHT, 1-dim *Zipf*, $|D| = 256$

(d) AHEAD, 1-dim *Cauchy*, $|D| = 1024$ (e) HIO, 1-dim *Cauchy*, $|D| = 1024$ (f) DHT, 1-dim *Cauchy*, $|D| = 1024$

Figure 14: The MSE of different methods when varying user scales and privacy budgets. The results are shown in log scale.



(a) AHEAD, 1-dim *Zipf*, $N = 10^5$ (b) HIO, 1-dim *Zipf*, $N = 10^5$ (c) DHT, 1-dim *Zipf*, $N = 10^5$

(d) AHEAD, 1-dim *Cauchy*, $N = 10^5$ (e) HIO, 1-dim *Cauchy*, $N = 10^5$ (f) DHT, 1-dim *Cauchy*, $N = 10^5$

Figure 15: The MSE of different methods when varying domain sizes and privacy budgets. The results are shown in log scale.

increase of domain size. Since all the three algorithms adopt the user partition strategy and the number of groups grows with the domain size, the user scale in each group becomes smaller for a larger domain size. For example, when domain size rises from 32 to 4096, the number of user groups enlarges from 5 to 12 for DHT. Based on Equation 1 and Equation 2, the variance of the FO algorithm is inversely proportional to user scale. 2) As shown in Figure 15(a) and Figure 15(d), AHEAD is less affected by the changes in domain size. Due to the threshold setting, AHEAD makes the actual domain size smaller by amalgamating some lower frequency sub-domains. In addition, AHEAD averages the estimated frequency values of repeated sub-domains, *e.g.*, sub-domains $[0, 3]$ and $[6, 7]$ in Figure 3, to reduce the influence of noise. Based on the results in Figure 15 and the above analysis, we have the following observation on the impact of domain size.

Observation 3 (Robustness under various domain sizes). *The impact of varying domain size on* AHEAD *is different from* HIO *and* DHT. AHEAD *reacts more robust to domain size changes.*

J.3 Impact of Data Skewness

Setup. We compare the algorithms' performance at various data skewnesses, where the data records are sampled from low skewness (0 for both *Gaussian* and *Laplacian*) to high skewness (0.9 for *Gaussian* and 2.0 for *Laplacian*). As shown in Figure 16, under each user scale, we set 8 different privacy budgets and use heatmaps to illustrate the impact of skewness and privacy budget on the MSE.

1) From Figure 16(a) and Figure 16(d), when privacy budget $\epsilon \leq 0.1$, the MSEs of AHEAD have a tendency to decrease with the increase of data skewness. Since there may exist more nodes suitable for pruning for data with higher skewness, *i.e.*, more sparse sub-domains, AHEAD can significantly suppress the injected noises. 2) With the increase of privacy budget or user scale, the impact of skewness becomes insignificant on MSE of AHEAD. Recalling Equation 4 in Section 4.4, the threshold θ becomes small with a large privacy budget or user scale. For instance, $\theta = 0.11$ when $\epsilon = 0.1$ and $N = 10^6$, and $\theta = 0.003$ when $\epsilon = 1$ and $N = 10^7$. For large privacy budgets or user scales, AHEAD reduces the intensity of tree pruning, where the impact of data skewness on the tree construction will weaken.

Observation 4 (The benefit from high skewness). AHEAD *tends to have smaller MSE on highly skew data, where the impact of skewness will weaken as the privacy budget or user scale increases.*

J.4 Impact of Attribute Correlation

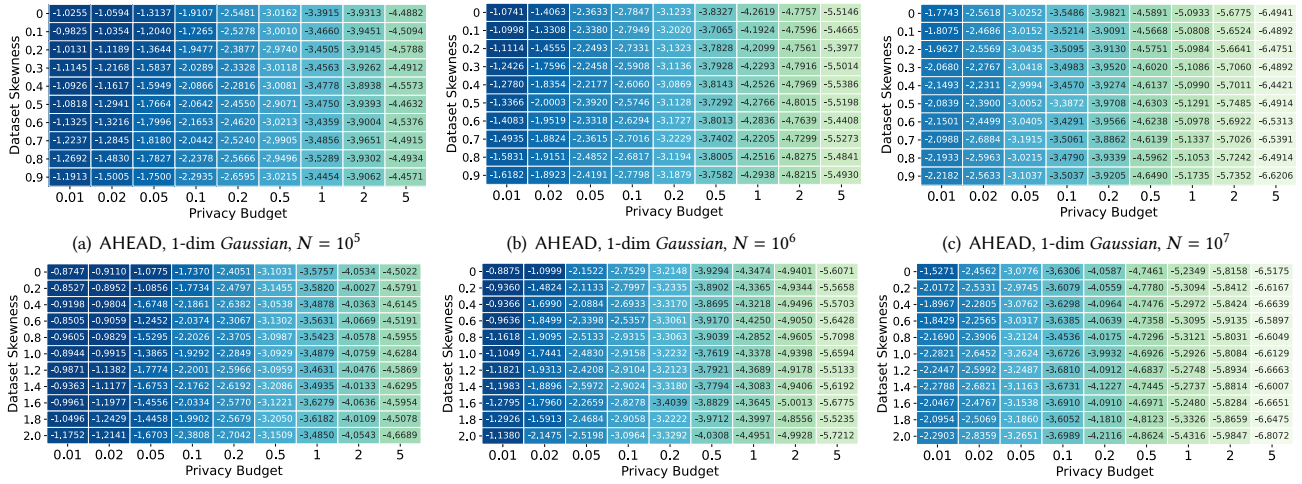
Setup. Next, we evaluate the impact of different attribute correlations on query errors as shown in Figure 17 and Figure 18. The experimental settings are similar to 2-dim scenes, *i.e.*, varying the correlation coefficient r from 0.1 (*representing weakly correlated*) to 0.9 (*representing strongly correlated*) with a fixed privacy budget $\epsilon = 1.1$.

From the results, we have the following findings. 1) The LLE method can protect the correlation between attributes in high-dimensional situations. Instead of decomposing all dimensions simultaneously like DE, LLE uses the 2-dim AHEAD tree to estimate the answers of high-dimensional range queries. The MSE of AHEAD behaves consistently across different correlations, thus the correlation of the data is well preserved by LLE. 2) For high-dimensional scenarios, the impact of attribute correlation becomes less on HDG. With the increase of dataset dimension, the number of 1-dim grids (C_m^2) increases faster than the 1-dim grids (C_m^1). Therefore, as shown in Figure 18, the negative impact of the 1-dim grids on the correlation is reduced.

Observation 5 (Robustness under various attribute correlation). *The MSE of* AHEAD *behaves consistently on attribute correlation changes. As the data dimension increases, the impact of correlation on* HDG *decreases.*

J.5 Remarks

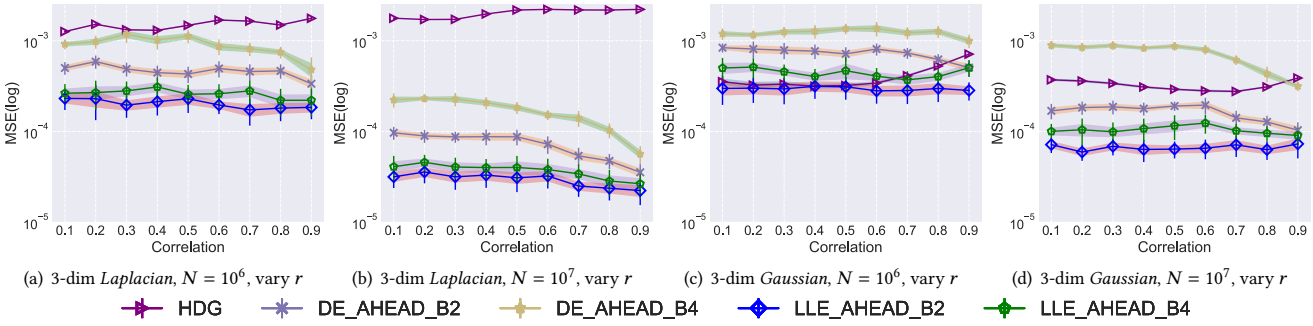
The six observations mentioned above reflect the impact of various factors on the performance of AHEAD. Observation 1 and Observation 2 describe the coupling influence of user scale and privacy budget on algorithm accuracy and demonstrate the transformation relationship between the user scale and privacy budget. Observation Observation 4 provides an encouraging of practical adoption of AHEAD when facing highly skew data. Observation 3 and Observation 5 demonstrate the advantageous robustness of AHEAD under different domain sizes and attribute correlations.



(a) AHEAD, 1-dim *Gaussian*, $N = 10^5$ (b) AHEAD, 1-dim *Gaussian*, $N = 10^6$ (c) AHEAD, 1-dim *Gaussian*, $N = 10^7$

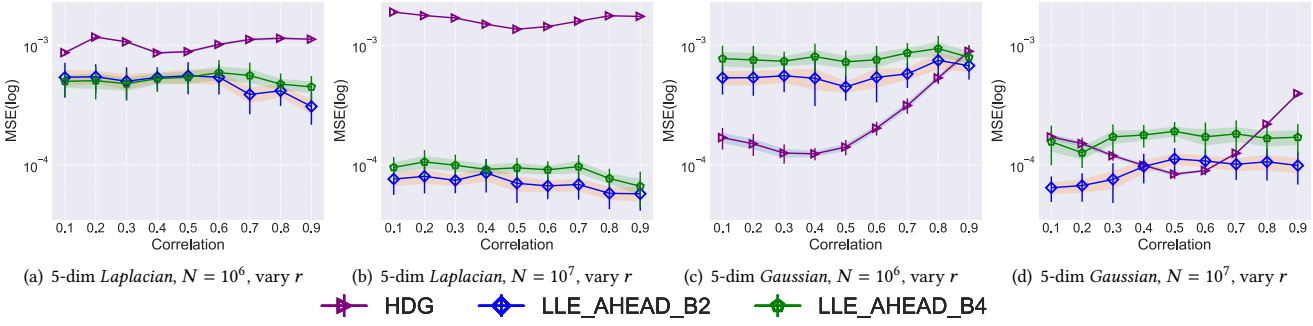
(d) AHEAD, 1-dim *Laplacian*, $N = 10^5$ (e) AHEAD, 1-dim *Laplacian*, $N = 10^6$ (f) AHEAD, 1-dim *Laplacian*, $N = 10^7$

Figure 16: The MSE of AHEAD when varying data skewnesses and privacy budgets. The results are shown in log scale.



(a) 3-dim *Laplacian*, $N = 10^6$, vary r (b) 3-dim *Laplacian*, $N = 10^7$, vary r (c) 3-dim *Gaussian*, $N = 10^6$, vary r (d) 3-dim *Gaussian*, $N = 10^7$, vary r

Figure 17: Comparison of different methods on 3-dim *Laplacian* and *Gaussian* datasets under various attribute correlations. DE and LLE respectively represent two high-dimensional expansion methods, *i.e.*, "direct estimation" and "leveraging low-dimensional estimation". HDG is a baseline method. The results are shown in log scale.



(a) 5-dim *Laplacian*, $N = 10^6$, vary r (b) 5-dim *Laplacian*, $N = 10^7$, vary r (c) 5-dim *Gaussian*, $N = 10^6$, vary r (d) 5-dim *Gaussian*, $N = 10^7$, vary r

Figure 18: Comparison of different methods on 5-dim *Laplacian* and *Gaussian* datasets under various attribute correlations.

In detail, 1) when the privacy protection strength (the privacy budget ϵ) is fixed, according to Observation 1, AHEAD obtains more accurate query answers with larger user scales. 2) If the user scale is insufficient, the aggregator tends to pay the cost, such as compensation fee, to trade with users to reduce the strength of privacy protection (increasing the privacy budget ϵ). From Observation 2, the aggregator can readily calculate the appropriate ϵ based on the user scale. 3) For the continuous attribute, the aggregator divides the total domain into slots of a fixed length and bucketize the records at a suitable granularity for queries [11, 62]. Therefore, when the domain size is 1024, the bucketize granularity is two times that of the domain size 512. From Observation 3, since AHEAD is more robust to domain size changes, the aggregator can choose a larger domain size to obtain a higher granularity. 4) When private data tends to have a high skewness, such as data from income, social networks and Web surfing, practitioners should give priority to AHEAD from Observation 4. 5) From Observation 5, in practice, AHEAD can handle the impact of attribute correlation varying.