PrivNUD: Effective Range Query Processing under Local Differential Privacy

Ning Wang[1,§], Yaohua Wang[1,4,§], Zhigang Wang[1,*], Jie Nie[1], Zhiqiang Wei[1], Peng Tang[2], Yu Gu[3], Ge Yu[3]

[1]*School of Computer Science and Engineering, Ocean University of China, China*
[2]*Key Lab of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, China*
[3]*School of Computer Science and Engineering, Northeastern University, China*
[1]{wangning8687,wangzhigang,niejie,weizhiqiang}@ouc.edu.cn,
[2]tangpeng@sdu.edu.cn,[3]{guyu,yuge}@mail.neu.edu.cn,[4]wangyaohua@stu.ouc.edu.cn

***Abstract*—Local differential privacy (LDP) has been established as a strong privacy standard for collecting sensitive information from users. Although it has attracted much research attention in recent years, the majority of existing works focus on applying LDP to frequency distribution estimation for each individual value in a discrete domain. This paper concerns the important range queries involving multiple discrete values. Till now, only a few works target this problem. They all rely on the B-ary tree to construct a uniform and hierarchical decomposition, so as to decrease the error when answering large range queries. However, the uniform splitting manner ignores the properties of decomposed sub-domains and processes them equally without preferences, which leads to significant performance penalty.**

In this paper, we tackle the problem head on: our proposal, privNUD, is a novel domain hierarchical decomposition mechanism. It dynamically decomposes each domain with a tailored granularity into some sub-domains, which sensitively considers the potential chances to answer one range query. The issue of granularity is carefully analyzed for better performance. It also can smartly prune the sub-domains with small frequencies. Besides, an adaptive user allocation technique is designed to dynamically decide the scale of users that are involved in each sub-domain's frequency estimation. Extensive experiments using real and synthetic datasets demonstrate that privNUD achieves significantly higher result accuracy compared to the up-to-date solutions.

***Index Terms*—local differential privacy, range query, hierarchical decomposition**

I. Introduction

Local differential privacy (LDP) is a rigorous privacy protection standard for collecting sensitive data from individual users. It has been adopted and widely deployed by many well-known companies such as Google [1], Apple [2] and Microsoft [3]. Specifically, LDP guarantees that no adversary, including the *aggregator* itself, can possibly infer the exact values of private information with high confidence, regardless of the adversary's background knowledge. This is done by letting each user perturb her data record locally to satisfy differential privacy [4], and send only the perturbed version of the record to an *aggregator*. Then the latter performs computations on the collected noisy data to estimate statistical analysis results on the original data. Undoubtedly, LDP provides a strong privacy assurance to users, as the true values never leave local devices.

As a relatively new concept, most of existing LDP-related works are devoted to applying LDP to the frequency distribution estimation on the entire domain of a given attribute, which answers the unit query with high accuracy. However, more applications ranging from market analysis to quantitative economics research involve the frequency estimation of a certain range on a domain. That is, we care about the number of users with several values covered by a range. This paper thereby focuses on answering such range queries under LDP with high result utility.

A straightforward way for this task is to aggregate the noisy frequencies provided by the existing methods [5], [6], which is originally designed for answering the unit bin queries. However, the accumulated noise will lead to a useless result, especially when the range is large. A natural method of improving utility is to keep additional bins over sub-intervals of the domain as auxiliary information. The benefit is that the range query involving many unit bins can be answered with a rightly small number of sub-intervals, which exactly covers the query range. The criterion of choosing sub-intervals is of course a key issue. Currently, most of the existing works rely on the B-ary tree structure to choose sub-intervals, which split the coarse-grained domain (corresponding to one parent node) into B fine-grained sub-domains with uniform length (corresponding to children nodes) iteratively. The root and leaf nodes indicate the entire domain and unit values respectively, and all sub-domains associated with the nodes except the root are regarded as sub-intervals.

The problem with hierarchical tree-based methods is that the uniform splitting manner leads to each sub-domain treated with equal weight. However, even if without query workload given beforehand, i.e., any range query generated from the domain could be submitted with the same probability [7], the decomposed sub-domains from different levels, even the ones from the same level have quite different impact on the result accuracy of range queries. For example, Fig. 1 shows a hierarchical decomposition provided by the 2-ary tree on the attribute domain $\{0,1,...,63\}$. Queries answered by a sub-domain are the ones with ranges covering it but not its

§Co-first authors
*Corresponding author

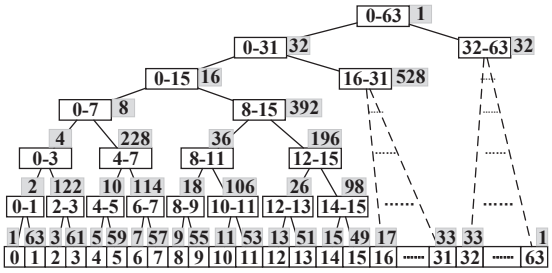


Fig. 1. A hierarchical decomposition tree with a uniform decomposition granularity

parent. So the sub-domains [0,7] and [8,15] located in the same level can be involved in answering 8 queries and 392[1] queries respectively, which implies [8,15] is more important. Accordingly, a more fine-grained decomposition can be done for this sub-domain with less noise added. Existing works ignore this fact and treat each sub-domain without preference. They then add the same scale of noise and hence output low utility answers. Different from them, our goal is to construct a non-uniform decomposition for these sub-domains by considering their potential chances to answer one query, instead of a uniform decomposition granularity for all sub-domains.

However, the non-uniform decomposition setting poses great challenges for finding a best hierarchical tree structure, since now the number of all possible candidates can be large. To crack this nut, this paper proposes privNUD, a novel domain decomposition mechanism for auxiliary sub-domains selection. It considers the potential chances of sub-domains to answer one query, and hence obtains higher result accuracy for range queries.

Specifically, most existing solutions adopt a pre-defined granularity to iteratively decompose the domain. Different from them, privNUD performs the decomposition using a non-uniform and changeable granularity customized for each sub-domain to be decomposed. Note that each sub-domain owns different properties such as the length and the number of possible queries involving it. As a result, the customized granularity contributes to improving the result accuracy of queries involving the sub-domain. Further, the granularity setting is of course a key issue. We carefully analyze the error source of the query answer obtained by privNUD, and then simulate it as an optimization problem. By solving this problem, privNUD can find the best granularity to provide answers with the highest accuracy. Moreover, privNUD also designs a pruning strategy tailored for the non-uniform decomposition mechanism. The goal is to avoid constructing those sub-domains whose frequency estimations are overwhelmed by the injected noises. Besides, we are aware that the non-uniform decomposition can generate a direct negative impact. That is, the paths on the tree are with different lengths. The traditional way for data collection is not applicable, because it groups users into disjoint parts and each part is involved in the frequency estimations on the sub-domains from the same level. privNUD thereby adopts an adaptive user allocation technique, which decides the number of users allocated to one node based on the associated domain adaptively. Last but not least, for multidimensional range queries, we rely on different structures adaptive to domain dimensionality to decompose the domain, which achieves significant performance improvement.

[1] $(8-0+1)\times(63-15+1)-(0-0+1)\times(63-15+1)=392$.

Contributions. To summarize, this paper makes the following contributions:

- We design a novel domain decomposition mechanism privNUD for one-dimensional range query under LDP, which decomposes the sub-domains with non-uniform granularities. It includes a guideline for choosing the decomposition granularity for each sub-domain as well as a prune strategy tailored for the proposed mechanism to remove the sub-domains with small frequencies.
- We design an adaptive user allocation technique, which decides the number of users allocated to one node based on the associated domain adaptively.
- We rely on different structures adaptive to domain dimensionality to decompose the domain, which achieves significant performance improvement.
- We conduct extensive experiments on both real and synthetic datasets to confirm the high utility of privNUD in both 1-dimensional and multi-dimensional queries. The results show that privNUD outperforms existing approaches by around one order of magnitude.

Organization. The remainder of this paper is organized as follows. Section II reviews existing studies about the problem this paper focuses on. Section III presents preliminaries about local differential privacy, problem definition and the existing methods. Section IV gives the detailed design of privNUD. Section V evaluates the usefulness of our proposals, and Section VI finally concludes the paper.

II. RELATED WORK

This section reviews some related works in both centralized differential privacy (CDP) with a trusted aggregator and local differential privacy.

Range Query under DP. Some existing works pay attention to answering range queries under traditional CDP. The hierarchical decomposition mechanism is originally proposed by Hay et al. [8]. They boost the result accuracy by enforcing consistency among the noisy frequencies along the tree. Following that, several works are proposed based on the hierarchical trees. Qardaji et al. [7] analyze the factors affecting range query accuracy under a hierarchical structure, and propose a way to decide the granularity of hierarchical decomposition, i.e., the fanout of the tree structure. Zhang et al. [9] propose a perturbation mechanism which decides whether a sub-domain on the tree should be split, without worrying about the privacy budget consumption. Cormode et al. [10] complete the privacy

space decomposition with quadtrees and kd-trees to answer various ranges of queries. Besides, Xiao et al. [11] use the Haar wavelet transform method to provide answers to range queries. All the above methods focus on the setting without the query workload given beforehand. Namely, they are devoted to designing the mechanisms for answering all possible range queries.

Other techniques for analyzing general query workloads under ϵ-DP are also discussed. Li et al. [12] utilize the matrix-based optimization mechanism to select some auxiliary sub-intervals to improve the result accuracy. Instead of representing query workloads as fully-materialized matrices in [12], McKenna et al. [13] use a compact implicit matrix representation of the queries, which can effectively find the better auxiliary sub-intervals to answer the queries. Since the complexity of these methods depends on the number of queries, it is impractical to utilize them to deal with the query workloads consisting of all possible queries.

Range Query under LDP. There are also a handful of works on range queries under LDP. Cormode et al. [14] analyze two range query mechanisms based on wavelet transform and hierarchical structure. And experimentally demonstrate that the wavelet transform has high query accuracy at high privacy budgets and the hierarchy works well at low privacy budgets, but they do not extend the model to multi-dimensional scenarios. Following that, HIO [15], AHEAD [16], HDG [17] are also proposed, which are described in detail in Section III-C. All the above methods focus on answering all possible range queries. But they do not consider the importance of different sub-domains. Besides, McKenna et al. [18] propose a new LDP mechanism that adapts to a given query workload, which uses a projected gradient descent algorithm to find the optimal mechanism. Similar to that in CDP, this method has difficulty in dealing with all possible queries.

Marginal Release under LDP. Our work is also related to the line of marginal release. Cormode et al. [19] implement marginal release under LDP based on transformations. Ren et al. [20] propose a multidimensional joint distribution algorithm based on Expectation Maximization and lasso regression, but could not guarantee the computational efficiency in high-dimensional scenarios. Zhang et al. [21] design the CALM mechanism, which firstly chooses some attribute combinations as views, and then publishes the frequency distribution estimations on these views. Following that, any marginal query can be derived based on the auxiliary information. Liu et al. [22] do a similar thing but with the junction tree structure. Although a λ-D range query can be derived based on a λ-way marginal by aggregating the noisy frequencies of cells located in this query, the accumulated noise lead to low result utility if the range is large.

III. PRELIMINARIES

This section introduces necessary background knowledge about local differential privacy (Section III-A), our problem definition (Section III-B) and the most representative existing methods (Section III-C).

A. Local Differential Privacy

The LDP setting involves an aggregator and a number of (say, n) users, each of which possesses a data record t_i containing private information. To protect privacy, each user u_i ($1 \leq i \leq n$) locally perturbs her record t_i and sends the perturbed data t_i^* to the aggregator. Then, the latter computes statistical models based on the collected data from all users. The perturbation ensures that any third party including the aggregator cannot infer the real record t_i from the perturbed one t_i^* with high confidence. Formally, LDP is defined as follows.

Definition III.1 (ϵ-local differential privacy)**.** *A perturbation mechanism f satisfies ϵ-LDP, if and only if for any two input tuples t and t' in the domain of f, and for any output t^* of f, we have:*

$$\Pr\left[f(t) = t^*\right] \leq e^{\epsilon} \cdot \Pr\left[f(t') = t^*\right]. \tag{1}$$

where ϵ is the privacy budget, which controls the level of privacy protection. A smaller ϵ means stricter privacy protection, and vice versa.

Fundamental LDP mechanisms. A basic mechanism for enforcing ϵ-LDP is Randomized Response (RR) [23], which deals with the situation that (i) each user possesses a single bit (i.e., either 0 or 1) of information, and (ii) the aggregator aims to compute the number of users who possesses 1, under ϵ-LDP. The recent work [5] presents a variant of the RR mechanism, called optimized RR. Specifically, each user u_i executes optimized RR with her private record t_i (i.e., a single bit) as input, and reports the output $\tilde{t_i}$ to the aggregator, where $\tilde{t_i}$ keeps the value of t_i with probability $p_1 = 0.5$ when $t_i = 1$ and $p_0 = 1/(e^{\epsilon} + 1)$ when $t_i = 0$. The latter then computes $\frac{\sum_i \tilde{t_i} - n \cdot p_0}{p_1 - p_0}$, as an estimate for the target value $\sum_i t_i$. Note that the output of optimized RR can have only two possible values, both of which can be calculated at the aggregator without private information; hence, it suffices for each user to transmit only a single bit to the aggregator, which minimizes communication overhead. Optimized RR can be extended to applications where each user's record is not a single bit. Each user u_i possesses a categorical value t_i (e.g., an emoji [24]), and the aggregator's goal is to estimate the frequency of each value in the domain, under LDP. A common approach is to transform t_i into a bit vector (e.g., using one-hot encoding [1]), and apply optimized RR to each bit in the vector [1], [5], [25].

B. Problem definition

This paper focuses on the problem of range query on the multidimensional data, without foreknown query workload, i.e., any possible range query may be requested. Formally, suppose there are n users, and u_i ($1 \leq i \leq n$) denotes the ith user. Each user u_i's private data contains d ordinal attribute

values and is represented by a tuple $t_i = (t_i^1, t_i^2, ..., t_i^d)$, where t_i^j indicates the jth value on attribute A_j. Without loss of generality, we assume that all attributes have the same domain [D]={1,2, ..., D}.

Based on the multidimensional data setting, a λ-dimensional (λ-D) range query performs on λ different interested attributes associated with specific ranges. Let $\phi_1, \phi_2, ..., \phi_\lambda$ be the indexes of interested attributes and $[l_{\phi_i}, r_{\phi_i}]$ be the specific query range on attribute A_{ϕ_i}. Then λ-D query q can be written as

$$q = (A_{\phi_1}, [l_{\phi_1}, r_{\phi_1}]) \wedge (A_{\phi_2}, [l_{\phi_2}, r_{\phi_2}]) \wedge \cdots \wedge (A_{\phi_\lambda}, [l_{\phi_\lambda}, r_{\phi_\lambda}]).$$

The answer to the above range query q is the fraction of users whose records satisfy the range constraints on all interested attributes, which can be represented as

$$\frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{\cap \left\{ l_{\phi_j} \le t_i^{\phi_j} \le r_{\phi_j} \right\}_{j=1}^{\lambda}},$$

where $\mathbb{I}_\gamma$ is an indicator function whose value is equal to 1 if the predicate γ is true and 0 otherwise.

The goal of this paper is to let the untrusted aggregator collect data from users to construct LDP-compliant frequency estimations on some well-chosen subintervals, in order to maximize the result utility of all possible range queries.

C. Existing methods

In the following, we presents several representative works for range queries under LDP.

Discrete Haar Wavelet Transform (DHT). DHT [14] chooses the Haar wavelet coefficients as the auxiliary information for answering range queries on one-dimensional data. Specifically, such coefficients can be organized by a complete binary tree, in which each leaf corresponds to the frequency on a unit from the attribute domain. And the non-leaf node corresponds to a coefficient which is equal to the difference between the frequency sum of leaves from left subtree and the one from right subtree. DHT splits users into disjoint parts, each of which is involved in computing the coefficients from one layer under LDP. Based on the noisy coefficients, any range query can be answered. The main benefit for DHT is that a range query with large length can be answered with a small number of coefficients, which can improve the result accuracy.

Hierarchical Interval Decomposition (HIO). HIO [15] decomposes the attribute domain into some intervals and the frequency estimations on these intervals are regarded as the auxiliary information to answer range queries. In particular, it relies on the hierarchical tree with fanout k to iteratively decompose the domain into k sub-domains uniformly until minimum granularity units on the domain are reached. The domain to be decomposed corresponds to one node in the tree and the decomposed sub-domains correspond to k children. Then HIO splits users into disjoint parts, each of which is involved in estimating the numbers of users located in the sub-domains on one layer under LDP. Based on that, any range query can be answered by using the minimum number of intervals from different levels, thus substantially reducing the cumulative error.

Adaptive Hierarchical Interval Decomposition (AHEAD). Similar to HIO, AHEAD [16] also follows the hierarchical decomposition framework to derive the auxiliary information. The main difference between them is that AHEAD adaptively decides whether a node (interval) continues to being decomposed by comparing the associated frequency with a threshold. Such operation could avoid decomposing the intervals with small frequencies, since small frequencies are usually useless and overwhelmed by the noise injected. On the other hand, when answering multidimensional range queries, Du et al. [16] directly apply the above mechanism and use disjoint parts of users to decompose the 2-dimensional domains of all attribute pairs. Then give a λ-D query, AHEAD derives C_λ^2 2-D sub-queries, each of which can be answered based on the corresponding hierarchical structures. With these results, the maximum entropy principle is run to achieve the estimation for this λ-D query.

Grid Decomposition (HDG). HDG [17] is proposed to answer multidimensional range queries. Different from the methods above, HDG adopts the frequency estimations on grids as the auxiliary information to answer range queries. In particular, given two parameters g_1 and g_2 customized with ϵ, HDG partitions the one-dimensional domain of each attribute into g_1 grids and two-dimensional domain of each attribute pair into $g_2 \times g_2$ grids. Then the users are split into $d + C_d^2$ parts, each of which is involved in the frequency estimations on 1-D or 2-D grids. With these estimations on attributes A_j, A_k as well as attribute pair (A_j, A_k), HDG iteratively constructs the response matrix for any attribute pair (A_j, A_k), which is used to answer 2-D range queries. Finally, HDG uses a similar way to AHEAD to answer any λ-D range query based on 2-D range query results. The main benefit of HDG is that it uses 1-D grids to capture finer-grained information to optimize the estimations of 2-D grids, thus improving the result accuracy.

Prefix-sum-based Cube Construction (PRISM). PRISM [26] constructs one-dimensional prefix-sum cube on each attribute and two-dimensional cubes on some carefully chosen attribute pairs respectively as the auxiliary information to answer range queries. Specifically, for an attribute with domain D, PRISM collects information from users to derive the frequency estimation for range queries $[1, i]$ for any $i \in [1, |D|]$ with sensitivity $|D|$, which forms the prefix-sum cube associated with this attribute. Then for any range query $[p, q]$, PRISM can answer it by using 2 pieces of prefix-sums from ranges $[1, p]$ and $[1, q]$. However, the high sensitivity $|D|$ for prefix-sum publication leads to useless prefix-sum results. PRISM alleviates it by decomposing the domain into g parts uniformly and just publishing prefix-sums on the coarse-grained domain. In this way, it reduces the sensitivity from $|D|$ to g. Following that, PRISM calculates the two-dimensional prefix-sum cubes among k attributes, which are selected by its attribute selection strategy. Finally, given a λ-D range query,

it uses the existing prefix-sums cubes related to this queries and runs the maximum entropy model to derive the query result. Although PRISM can use fewer noisy estimations to answer each query, the error variance of each estimation is big. That makes the prefix-sum-based technique usually perform worse than the hierarchical decomposition-based technique. We analyze the error variances from these two kinds of methods in Appendix.A.

In summary, no matter the methods for one dimensional queries or multidimensional ones, all of them decompose a given domain in a uniform manner. Recall that these works and ours all aim to answer all possible range queries. It is undoubtedly different sub-domains are involved in different number of range queries. Uniform decomposition leads to some frequently used sub-domains with high noise, further generating significant performance penalty.

IV. PrivNUD:Dynamic Non-uniform Decomposition

This section describes our proposal privNUD, a novel dynamic domain decomposition mechanism with a non-uniform granularity. To better show it, we firstly use an example to illustrate the motivation in Section IV-A. And then Section IV-B presents a general framework of privNUD. Following that, Sections IV-C and IV-D derive a guideline to choosing decomposition granularity for each sub-domain and a prune strategy tailored for privNUD respectively. Finally, Section IV-E introduces how to answer multidimensional range queries with privNUD.

A. Motivation

We use an example to illustrate the limitation of the existing hierarchical decomposition mechanism with uniform granularity [14]–[16] and the rationality of privNUD. Recall that Fig. 1 has demonstrated that different sub-domains can answer different numbers of queries. Fig. 2(a) gives a customized decomposition granularity solution, so that the sub-domain with a big query number like $[5, 9]$ can have a shorter sub-tree and then be allocated more users, to improve query accuracy. And the value in grey shadow also denotes the number of queries answered by the associated sub-domain and here it is termed as weight. We measure the performance of the different granularity choices by the error variance, which is imported when answering related queries using the sub-tree rooted by the given sub-domain. Here, the related queries include not only the ones directly answered by this sub-domain, but also those directly answered by its descendant sub-domains in the sub-tree. Since queries are answered by the given sub-domain and/or its descendants, we can compute the partial error caused by each of them, and then sum up partial errors as the error variance. The partial error is the product of the weight and the estimation error about the noisy frequency of a sub-domain. The latter is expressed by $\frac{A}{n'}$, where A is a value related to ϵ and n' is the number of allocated users. Take the sub-tree rooted by [0, 4] as an example. We can compute the error variance by $(59 + 3 + 123 + 1 + 63 + 3 + 61) * \frac{A}{1/3*X} + 240 * \frac{A}{2/3*X} = 1299A/X$, where X denotes the number

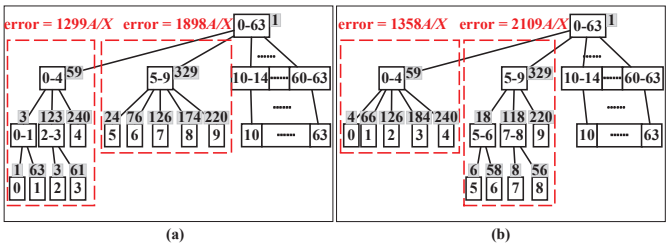


Fig. 2. An example to illustrate the motivation of non-uniform decomposition

of users available for the sub-tree rooted with [0, 4]. Here the factor $\frac{2}{3}$ is because $[4, 4]$ does not have children, and then users originally allocated to its children are now used by it. Fig.2(b) shows a counterpart that sub-domains [0, 4] and [5, 9] exchange their granularity choices. Then the errors respectively increase to $1358A/X$ and $2109A/X$. This reveals that using the uniform decomposition granularity for [0, 4] and [5, 9] with fanout 3 or 5 achieves sub-optimal performance, compared with the customized policy employed in Fig.2(a). Thus, we are motivated to apply non-uniform decomposition granularity for different sub-domains to achieve better query accuracy.

B. The framework of privNUD

This section describes the general framework of the proposed solution privNUD, highlighting its novel dynamic non-uniform decomposition design and leaving out several important details that are covered later in Sections IV-C and IV-D.

It is obvious that with the non-uniform decomposition granularity setting, the size of possible hierarchical tree structures is so large that it is hard to find the one providing the most accurate results for range queries. To tackle with that, privNUD adopts a heuristic algorithm to iteratively search a better granularity for the domain to be decomposed, in order to improve the performance when answering all possible range queries. It is worth mentioning that the uniform hierarchical tree in the existing methods [14]–[16] is the worst case from privNUD, which makes privNUD yield a better performance than uniform hierarchical decomposition. Besides, a direct negative effect imported by non-uniform decomposition is that the paths on the tree are with different lengths. That makes the traditional way for data collection not applicable, which groups users into disjoint parts and each part is involved in the frequency estimations on the sub-domains from the same level. So privNUD adopts an adaptive allocation strategy, which decides the number of users allocated to one node based on the domain adaptively.

Based on the above clarifications, we design the adaptive non-uniform decomposition tree construction algorithm at the aggregator, shown in Alg. 1. Specifically, the aggregator initializes the tree $\mathcal{T}$ with a root node v_r, which is associated with the whole attribute domain and takes all users as available users (Lines 1-2). Then, starting from the root, the aggregator iteratively splits nodes, as follows. Given a non-leaf node v, the aggregator firstly chooses a proper decomposition granularity

Algorithm 1: PrivNUD Tree Construction

Input : All users' data set $\{u_1, u_2, \cdots, u_n\}$, attribute domain D, privacy budget ϵ
Output: PrivNUD Tree $\mathcal{T}$

1 Initialize the tree $\mathcal{T}$ with a single root node v_r associated with domain D, and mark v_r as unvisited;
2 Set available user set $U(v_r)$ for root v_r to $\{u_1, u_2, \cdots, u_n\}$;
3 **while** *there exists an unvisited non-leaf node* v **do**
4 Mark v as visited;
5 Let $U(v)$ be the set of available users for node v;
6 Let $D(v)$ be the domain associated with node v;
7 Compute the best decomposition granularity g of $D(v)$ based on Section IV-C;
8 Randomly sample $\frac{|U(v)|}{\log_g |D(v)|+1}$ users U' from $U(v)$;
9 Apply optimized RR to collect information from U' using privacy budget ϵ, in order to derive the estimated frequency $f'(v)$ of v;
10 **if** $f'(v)$ *satisfies the undecomposable condition in Section IV-D* **then**
11 Apply optimized RR to collect information from $U(v) - U'$ using privacy budget ϵ, to refine the estimated frequency $f'(v)$ of v;
12 Mark v as a leaf node;
13 Continue;
14 Divide domain $D(v)$ into g disjoint sub-domains $\{D_i(v)\}$;
15 **for** k *from 1 to* g **do**
16 Add a child v_c of v with sub-domain $D_k(v)$;
17 Initialize estimated frequency $f'(v_c)$ to 0;
18 Initialize available user set $U(v_c)$ to $U(v) - U'$;
19 Mark v_c as unvisited;
20 Post-process the frequencies in nodes of $\mathcal{T}$ to enforce consistency;
21 **return** $\mathcal{T}$.

g for its associated domain $D(v)$ (Line 7). Then based on the current granularity g, she temporarily regards the height of the sub-tree with root as v is $\log_g |D(v)| + 1$. So she chooses $\frac{|U(v)|}{\log_g |D(v)|+1}$ users from v's available users set $U(v)$ to compute an estimate $f'(v)$ of $f(v)$, the number of users with values located in $D(v)$. Let U' be the set of chosen users (Line 8). The other users in $U(v)$, i.e., the ones in $U(v) - U'$ are left for the frequency estimations of the remaining nodes in this sub-tree. Then she invokes the LDP perturbation component optimized RR clarified in Section III-A to collect data from U' to derive the estimate $f'(v)$ (Line 9). Then, she checks if $f'(v)$ satisfies an undecomposable condition, which prevents the sub-domain with small frequency from being decomposed, since small frequencies trend to be overwhelmed by the injected noises. Such condition setting is nontrivial which needs to consider the true data distribution, query distribution as well as decomposition granularity. We discuss this in detail in Section IV-D. If the condition is satisfied, the aggregator marks v as a leaf node and uses all the remaining user data $U(v)$ to refine the estimated frequency of this node (Line 11); otherwise, the domain $D(v)$ is split g sub-domains evenly, each corresponding to one child v_c of v and v_c is with the remaining users in $U(v) - U'$ as the available users

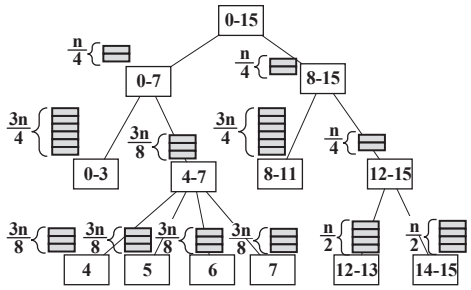


Fig. 3. An example to show the non-uniform decomposition mechanism

(Lines 14-19). After that, the aggregator continues to the next iteration, and attempts to decompose another domain with a non-leaf node in the tree. When there exists no unvisited non-leaf node, the iteration terminates and the non-uniform hierarchical decomposition is derived. Since the aggregator collects data from users to estimate the frequencies from different nodes independently, the consistency between the parent node's frequency and the sum of children nodes' frequencies is destroyed. So a post-processing operation on the tree for consistency enforcement is used to boost the result accuracy further.

Fig. 3 shows a possible tree structure output by privDUA in Alg. 1. The nodes can have different fanouts, i.e., the domains associated with these nodes are decomposed with different granularities. Besides, the lengths of the paths from leaves to root are various. The grey boxes besides the nodes indicate the users allocated to this node for frequency estimation and the height reflects the number of users. Obviously, the scale of users for each node is different even for the nodes in the same level. This is done mainly through an adaptive user allocation strategy. When the aggregator plans to compute one node's frequency, she estimates the length l of the path from leaf to this node with current decomposition granularity and splits the available users into l parts. Only one part of users is allocated to the node. Different decomposition granularities tailored for domains lead to different scales of users in this part, which makes user allocation adaptive to the domain.

Theorem IV.1. *Alg. 1 satisfies ϵ-LDP.*

Its proof appears in Appendix.B.

C. Adaptive decomposition granularity choosing

In privNUD, the key issue is of course the granularity choosing which controls the structure of hierarchical tree and decides the performance of our proposal. This section gives a guideline to choose a proper granularity tailored for each domain to be decomposed.

Recall that this paper aims to construct the hierarchical decomposition tree to optimize the result accuracy of all possible queries generated from the attribute domain. Namely, the goal is to minimize the average error variance for all range queries. On the other hand, when choosing a granularity for

domain $[l_v, r_v]$ associated with node v, the one associated with the parent node $p(v)$ has been known, due to the iterative construction manner. So the granularity for $[l_v, r_v]$ just has an effect on the error variances of the queries that intersects with this domain or covers this domain but intersects with the one associated with $p(v)$, since hierarchical decomposition mechanism answers a query with the coarse-grained sub-domains this query covers instead of the fine-grained ones. Consequently, we aim to choose the decomposition granularity for $[l_v, r_v]$ by minimizing the sum of error variances brought by answering the effected queries with the decomposed sub-domains from $[l_v, r_v]$. For example, as shown in Fig. 1, assuming we want to find a proper decomposition granularity for domain [4,7], different decomposition affects the intersected queries such as query [2,5] and the ones covering domain [4,7] but not [0,7] such as query [3,7]. So we want to optimize the error variances of these queries which are caused by the noisy estimations of decomposed sub-domains or this domain [4,7] itself.

Based on the above analysis, we can derive an objective function for the decomposition granularity choosing as follows,

$$\sum_{q \in Q_a} \sum_{o \in N(q,g)} E(o, g),$$

where Q_a denotes the effected queries with $[l_v, r_v]$, $N(q, g)$ denotes the decomposed sub-domains from $[l_v, r_v]$ involved in answering query q with granularity as g and $E(o, g)$ is the error variance of frequency estimation for sub-domain o. And the above expression can be written as

$$\sum_{o \in N(g)} \omega(o) E(o, g). \tag{2}$$

where $N(g)$ denotes the set of decomposed sub-domains with granularity g and $[l_v, r_v]$ itself, and $\omega(o)$ denotes the number of queries answered by sub-domain o. The value for $\omega(o)$ can be computed by the number of queries covering o minus the number of ones covering o's parent $p(o)$, where we slightly abuse o to denote the node associated with domain o. Let l_o (r_o) and $l_{p(o)}$ ($r_{p(o)}$) be the left (right) points of domain o and her parent domain $p(o)$ respectively. $\omega(o)$ is equal to $(l_o + 1) \cdot (D - r_o) - (l_{p(o)} + 1) \cdot (D - r_{p(o)})$.

On the other hand, assuming n' is the size of v's available users set $U(v)$, with granularity g, the height of the hierarchical sub-tree with root v is $\log_g(r_v - l_v + 1) + 1$. PrivNUD plans to adopt optimized RR [5] to collect data from $\frac{n'}{\log_g(r_v - l_v + 1) + 1}$ users to derive the frequency estimations of sub-domains in each level. Based on the conclusion from [5], the error variance $E(o, g)$ is $\frac{4[\log_g(r_v - l_v + 1) + 1]e^{\varepsilon}}{n'(e^{\varepsilon} - 1)^2}$.

Observing Eqn.5, we find that a smaller granularity g means fewer times of sub-domains used to answer the effected queries, but a higher sub-tree structure, further more noises injected to frequencies of each decomposed sub-domain. So there is a tradeoff between $\omega(o)$ and $E(o, g)$. We can traverse all g from 1 to $D(v)$ to find the granularity which minimizes Eqn.5.

D. Undecomposable condition setting

When constructing the non-uniform decomposition tree, for a domain with smaller frequency, it is pointless to continue decomposing this domain. That is because small frequency trends to be overwhelmed by the injected noise, further useless results generated. So it is an effective way to improve result accuracy by stoping decomposing the sub-domain with smaller frequency and estimating the frequency distribution on the domain with uniform distribution assumption.

The key issue is of course the undecomposable condition setting. If the condition is too strict, the sub-domains with relatively bigger frequencies stop being decomposed, which leads to higher error brought by the uniform distribution assumption; otherwise, useless results dominated by noise are generated. So we need to do a thorough analysis about the error variances generated in the two cases, with decomposition or without decomposition. And then a decision is generated based on the comparison of error variances. We show the error analysis in the following.

The case with decomposition. As shown in Alg. 1, if node v with noisy frequency $f'(v)$ continues to be split, the optimal decomposition granularity g has been known, since the number of users for computing $f'(v)$ depends on g. So the error variances of queries results incurred by v and its children $c(v)$ can be written as

$$[\omega(v) + \sum_{x \in c(v)} \omega(x)] \cdot \frac{4e^{\epsilon}}{n_1(e^{\epsilon} - 1)^2} \tag{3}$$

where $n_1 = \frac{|U(v)|}{\log_g |D(v)| + 1}$. Note that we just consider the query errors brought by v and v's children, not other descendant nodes. That is because these nodes are often judged as pruned in the subsequent iterations.

The case without decomposition. Recall that if node v is not split further, all of the available users owned by v are involved in computing $f'(v)$, which is shown in Lines 9 and 11. So the error variance for $f'(v)$ is $\frac{4e^{\epsilon}}{(|U(v)|)(e^{\epsilon} - 1)^2}$. With the uniform distribution assumption, given a decomposition granularity g, the frequency for the sub-domain associated with v's child is estimated as $\frac{f'(v)}{g}$. So the error variances of queries results incurred by v and its children $c(v)$ can be written as

$$\frac{4\omega(v)e^{\epsilon}}{(|U(v)|)\,(e^{\epsilon} - 1)^2} + \sum_{x \in c(v)} \omega(x) \cdot \left(\frac{f'(v)}{g} - f(x) \right)^2$$

When the real data distribution is extremely skewed on $D(v)$, i.e., the users located in $D(v)$ is all located in the sub-domain associated with v's child x^*, the above equation has the maximum, which is listed as follows.

$$\begin{aligned} &\frac{4\omega(v)e^{\epsilon}}{(|U(v)|)\,(e^{\epsilon} - 1)^2} + \sum_{x \in c(v) - \{x^*\}} \omega(x) \cdot \left(\frac{f'(v)}{g} \right)^2 \\ &+ \omega(x^*) \left(\frac{f'(v)}{g} - f(v) \right)^2 \end{aligned} \tag{4}$$

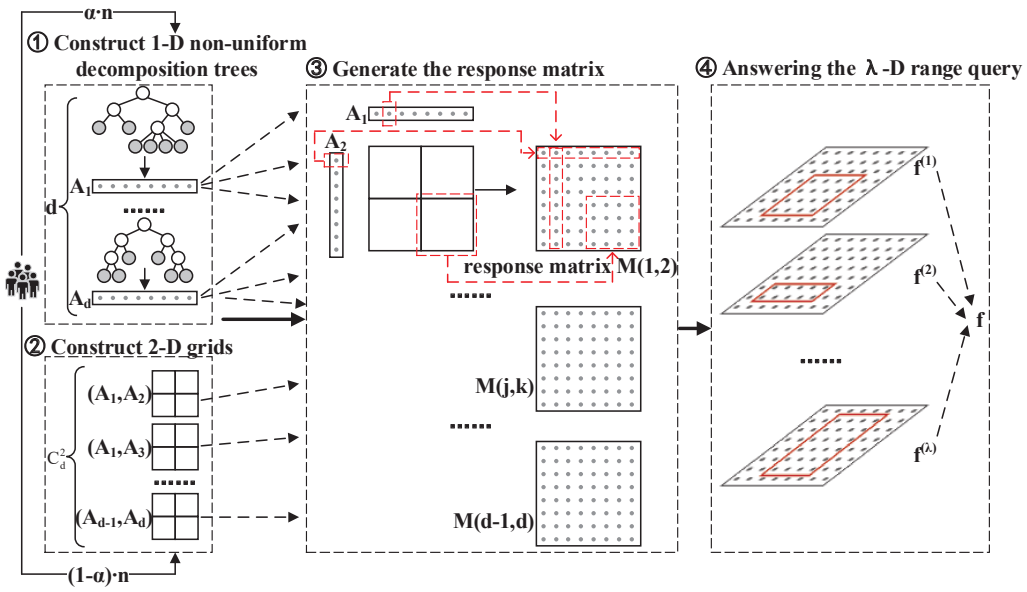


Fig. 4. The framework for multidimensional range queries

To derive the maximum of the above Equation, we let x^* be $\arg\max_{x \in c(v)} \omega(x)$ and set $f(v)$ as $f'(v)$. If Eqn.3 is smaller than Eqn.4, node v continues to be split, and stops otherwise.

E. Extend to Multi-dimensional Scenarios

Our proposal privNUD can answer one-dimensional range queries with great accuracy improvement over the existing methods shown in Section V-A. However, many datasets in practice have more than one dimensions. It is thus important to apply privNUD for processing multidimensional range queries.

To our knowledge, the current state-of-the-art techniques for multidimensional range queries are the grid-based and hierarchical tree-based methods, both of which rely on the frequency estimations on 1-D and 2-D structures to support the multidimensional queries. We find that hierarchical tree-based method works well on the 1-D range queries, since many sub-domains are constructed and the large range query can be answered with fewer sub-domains. However, on 2-D domain, it just performs better on the data with strong attribute correlations. In most cases, grid-based method performs better for 2-D queries, since it uses more users with scale of $\frac{n}{d+C_d^2}$ to compute the frequency estimations on the sub-domains from a 2-dimensional domain and can provide a better estimation results. In contrast, hierarchical tree-based method just uses $\frac{n}{h \cdot C_d^2}$ users to estimate the frequencies of sub-domains, where h denotes the height of a tree. Usually useless results are generated. Because of this, we prefer to use different structures to build 1-D and 2-D domain decomposition. In particular, we adopt privNUD to derive the frequency estimations on the 1-dimensional domain and then use the 1-dimensional estimations to refine the estimations from 2-dimensional grids. As shown in Fig. 4, the framework for multidimensional range queries includes four steps, which are described as follows:

Step 1: constructing non-uniform decomposition sub-domains on 1-dimensional domains. The aggregator randomly chooses αn users, which are split into d groups evenly. Taking each group of users as input, privNUD constructs the non-uniform hierarchical decomposition tree for one dimension. With the consistency enforcement technique in privNUD, the frequency distribution estimations on the 1-dimensional domain can be derived.

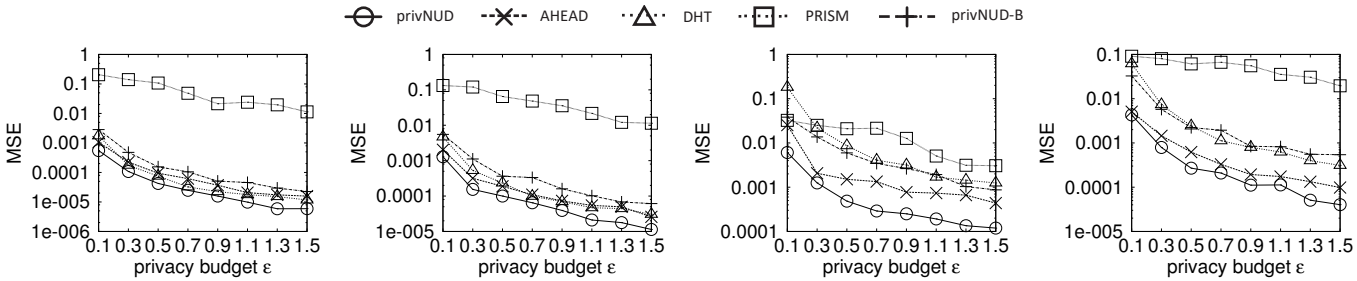
Step 2: constructing the grids. With the same grid granularity setting as that in [17], the aggregator splits the 2-dimensional domain for any attribute pair into $g_2 \times g_2$ grids. The remaining $(1-\alpha)n$ users are split into C_d^2 groups, each of which is used to estimate the frequency distribution estimation on the grids from one attribute pair.

Step 3: generating the response matrix. To generate response matrix for the attribute pair (A_j, A_k), we use the frequency distribution estimations on A_j and A_k from privNUD as well as the one on the grids of attribute pair (A_j, A_k). And a similar way to [17] is adopted to build the matrix.

Step 4: answering λ-dimensional range queries. Following the classical way in [17], [21], [28], [29], given a λ-D range query q, the aggregator splits q into C_λ^2 associated 2-D range queries, each of which can be answered with the response matrix. Based on the C_λ^2 results, the "weighted update technique"[2] [17], [28], [29] is applied to derive the estimation of the answer to q.

A guideline to set α. Since the ratio of users allocated for Step 1 can directly affect the utility of privNUD, we design

[2]We describe it in Appendix.C, since it is beyond the main scope of this paper.



(a) Loan $D = 256$ (b) Bfive $D = 512$ (c) Adult $D = 1024$ (d) Salaries $D = 2048$

Fig. 5. Result accuracy for 1-D range queries on real datasets by varying privacy budget ϵ.

the following guideline for setting α.

The goal of a proper α setting is to guarantee that the frequency estimations for one-dimensional and two-dimensional sub-domains are with the same scale of error, since both the estimations have an effect on the result accuracy of a multidimensional query. For the estimation of one-dimensional sub-domain, it is challenging to express its error variance in a formula. That is because the adaptive decomposition mechanism may make the nodes absorb different number of users to derive the frequency estimations. To tackle that, we use the error variance from uniform hierarchical decomposition structure with fanout 2 as the upper bound of estimation error from privNUD, which is written as $\frac{4d\log De^{\epsilon}}{\alpha n(e^{\epsilon}-1)^2}$. As for the estimations on two-dimensional sub-domains, the error variance is $\frac{4C_d^2e^{\epsilon}}{(1-\alpha)n(e^{\epsilon}-1)^2}$. By letting these two error variance be equal, we have $\alpha = \frac{d\log D}{C_d^2+d\log D}$.

V. Experiments

This section evaluates privNUD against four state-of-the-art techniques for range query processing under LDP, DHT [14], AHEAD [16], HDG [17] and PRISM [26]. For a fair comparison, these four methods are applied with the same parameter setting as in the original papers. Note that we do not consider another existing solution HIO [15], since it is shown in [14], [17] to be inferior to DHT on 1-D range queries and HDG on multidimensional ones. Besides, we also implement a version of privNUD with splitting the privacy budget instead of splitting users as a competitor, which is termed as privNUD-B.

All experiments were performed on a PC with Intel(R) Xeon(R) E-2224G CPU @ 3.50GHz and 16 GB of memory.

Datasets. We conduct experiments on five real datasets and four synthetic datasets. Among these real datasets, there are four low-dimensional datasets and one high-dimensional dataset. Table II-Table V in Appendix.D show the correlations between two attributes on the low-dimensional real datasets.

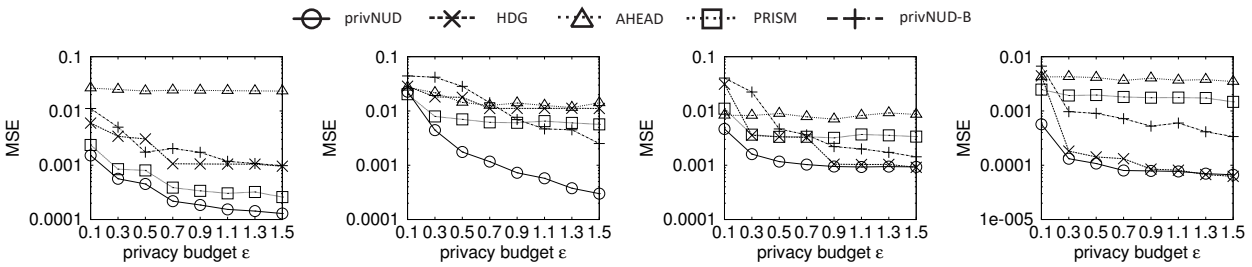
- Salaries [30]: The dataset is around 149 thousand records with 5 attributes "BasePay", "OvertimePay", "OtherPay", "TotalPay" and "TotalPayBenefits" about San Francisco employee salaries from 2011 to 2014.
- Adult [31]: The dataset is from the UCI ML repository with about 33 thousand records with 5 attributes "Age", "Fnlwgt", "Capital-gain", "Capital-loss" and "Hours-per-week".
- Loan [32]: The dataset contains the club's data on loans originated from 2007 to 2018. It contains around 2 million records with 5 attributes "Loan_amnt", "Int_rate", "Installment", "Annual_inc" and "Total_pymnt".
- Bfive [33]: The dataset is collected by an interactive online test platform that describes the time spent on each problem in milliseconds, with around 1 million records with 5 attributes "EXT1_E", "EXT2_E", "EXT3_E", "EXT4_E" and "EXT5_E".
- Ipums [34]: The dataset is from the IPUMS repository with around 3 million records and 30 attributes.
- Normal (Laplace): The datasets are synthesized from a multidimensional Normal (Laplace) distribution with a mean of 0 and a variance of 1. To show the performance on the datasets with different scales of users, the ones with 1 million and 10 million records are generated respectively. The settings are the same as that in [16].

Utility Metric. Following previous works [14], [16], we assess the performance of each technique based on *mean square error*, which is defined as $\frac{\sum_{q\in Q}\left(f_q-f_q^*\right)^2}{|Q|}$ for a query set Q, where $f^*(q)$ $(f(q))$ denotes the noisy (true) answer to query q. For each experimental setup, we randomly choose 200 queries from the attribute domains to form the set Q and show the result accuracy of these queries.

A. Evaluation Results on 1-D Range Query

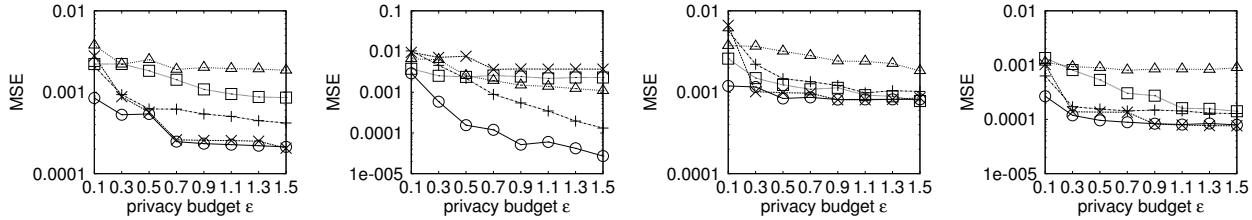
In this part for 1-D range query, we just compare our proposal with the competitors DHT, AHEAD, PRISM and privNUD-B, in which HDG is omitted due to its feasibility only for multidimensional range queries. We choose attributes "Total_pymnt", "EXT1_E", "Age" and "TotalPayBenefits" from the datasets "Loan", "Bfive", "Adult" and "Salaries" respectively to form one-dimensional datasets for evaluations. The same as that in Ref. [16], we also use different bucketize granularity $\{256, 512, 1024, 2048\}$ to bucketize the records from different datasets so that the evaluations are run on various domain size settings. Note that privNUD is qualified to process the dataset with any attribute domain size, not limited to the one with domain size equaling to a power of 2.

Fig. 5 shows the MSEs for comparing privNUD against all the competitors by varying ϵ from 0.1 to 1.5 on four real datasets. As expected, observe that privNUD consider-



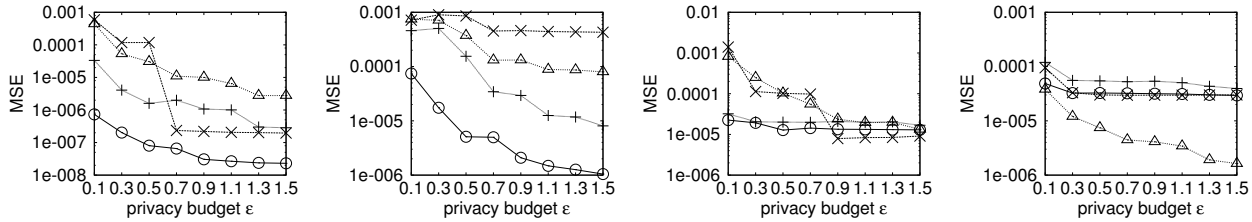
(a) Loan (b) Adult (c) Salaries (d) Bfive

Fig. 6. Result accuracy for 2-D range queries on real datasets containing 5 attributes by varying privacy budget ϵ.



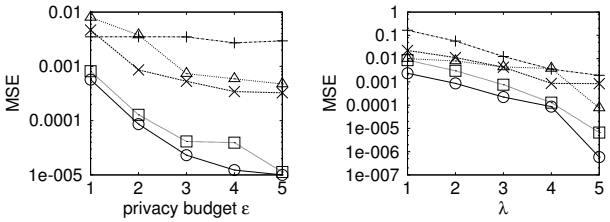
(a) Loan (b) Adult (c) Salaries (d) Bfive

Fig. 7. Result accuracy for 3-D range queries on real datasets containing 5 attributes by varying privacy budget ϵ.



(a) Loan (b) Adult (c) Salaries (d) Bfive

Fig. 8. Result accuracy for 5-D range queries on real datasets containing 5 attributes by varying privacy budget ϵ.



(a) Result accuracy vs. ϵ (b) Result accuracy vs. λ

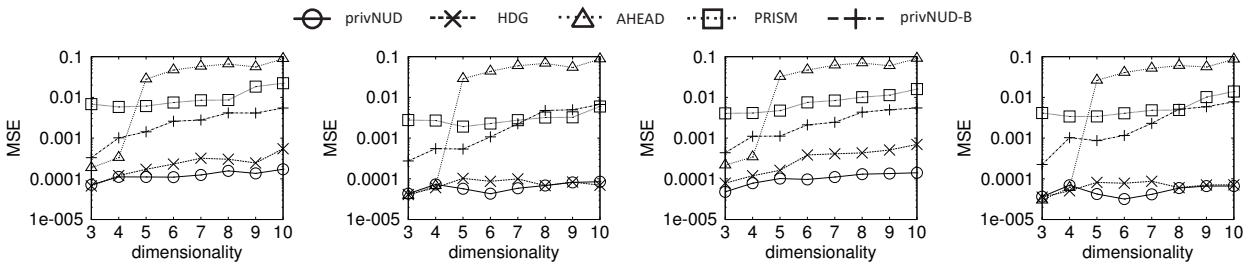
Fig. 9. Evaluation on the high dimensional dataset Ipums.

ably outperforms the other other competitors. The superior performance of the former is due to its dynamic decomposition granularity deciding strategy, which can achieve a better granularity and improve the result accuracy of queries involving the decomposed domain. As for the privacy budget splitting method privNUD-B, its theoretical error variance is roughly h times as big as that of privNUD with splitting users, where h denotes the number of parts that users or budget is split to. That offsets the performance gain brought by the adaptive decomposition granularity choosing technique, further leading to inferior performance of privNUD-B. Besides, it is also observed that the hierarchical tree-based methods including AHEAD and privNUD are superior to prefix-sum-based PRISM on the one-dimensional datasets. The main reason is that PRISM collects data with sensitivity as g to derive the frequency estimations involved in the prefix-sum cube. The high sensitivity leads to big error variance in the estimations, further low query accuracy, even if only two estimations are used to answer any query. In addition, Fig. 5 also shows that privNUD obtains different performance improvement over various datasets. For the dataset with smaller domain Loan, privNUD is slightly better than DHT and AHEAD. But for the ones with larger domain Bfive, Adult and Salaries, privNUD achieves significant advantages over the other two methods. We speculate that this is mainly because with larger domains, more hierarchical decomposition need to be carried out, which provides more chances for our decomposition mechanism, leading to significant performance improvement.

B. Evaluation Results on Multidimensional Range Query

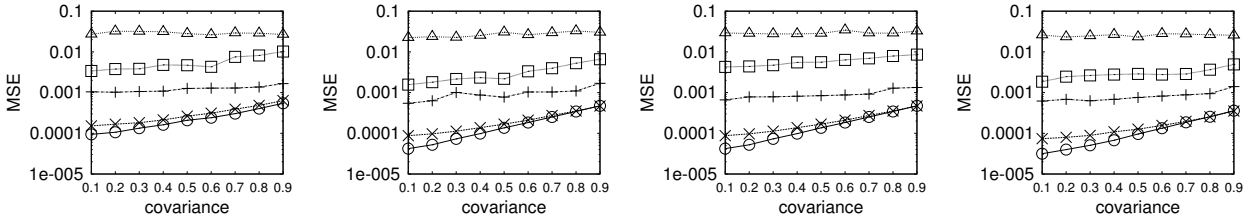
In this set of experiments, we consider the task of range queries on multidimensional data under LDP. We just compare our proposal with the competitors HDG, AHEAD, PRISM and privNUD-B, in which DHT is omitted due to its feasibility only for 1-D range queries.

Evaluations on low-dimensional datasets. Fig.6-Fig.8 illustrate the performance of privNUD and its competitors on four low-dimensional datasets and various privacy budgets for 2-D, 3-D and 5-D range queries respectively. For a fair comparison, we bucketize each attribute domain from the real datasets into 64 bins, which is the same as the processing involved in AHEAD and HDG. Not surprisingly, the accuracy



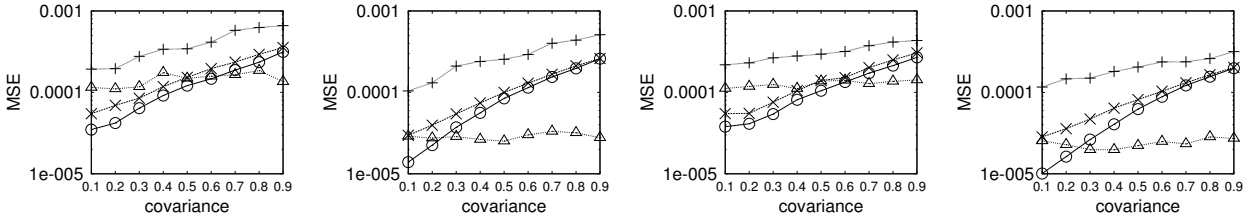
(a) Gauss 10^6 (b) Gauss 10^7 (c) Laplace 10^6 (d) Laplace 10^7

Fig. 10. Result accuracy for 3-D range queries on synthetic datasets with $cov = 0.2$ and $\epsilon = 1$ by varying dimensionality.



(a) Gauss 10^6 (b) Gauss 10^7 (c) Laplace 10^6 (d) Laplace 10^7

Fig. 11. Result accuracy for 3-D range queries on synthetic datasets with $d = 5$ and $\epsilon = 1$ by varying covariance.



(a) Gauss 10^6 (b) Gauss 10^7 (c) Laplace 10^6 (d) Laplace 10^7

Fig. 12. Result accuracy for 5-D range queries on synthetic datasets with $d = 5$ and $\epsilon = 1$ by varying covariance.

of all approaches becomes better when ϵ grows. Among these approaches, in most cases, privNUD performs the best and roughly one order of magnitude better than the other approaches. This is because the customized decomposition granularity for each decomposing domain imports a smaller overall estimation error for the queries involving this domain than the counterparts including HDG, PRISM and AHEAD those employ the uniform decomposition. Besides, we also observe that privNUD achieves a high accuracy on the Loan and Adult datasets, where the MSE of privNUD on Adult is almost two order of magnitude smaller than the competitors except privNUD-B. For the Salaries and Bfive datasets, privNUD has obvious advantages in the low ϵ region, but the improvement is less significant for a larger ϵ. The main reason for remarkable performance improvement on Loan and Adult is that the correlations among attributes on the two are weak as shown in Tables IV and II in Appendix.D. That makes the result accuracy for multidimensional range query be dominated by the frequency estimations on 1-D domains, since privNUD adopts estimations from 1-D to capture finer-grained information for the 2-D grids with attribute independence assumption. And our dynamic decomposition mechanism can provide better estimations on 1-D domains, further significant performance improvement for multidimensional queries. On the other hand, the attributes on the other two datastes are with strong correlations as shown in Tables III and V in Appendix.D, which makes the correlation capture for any attribute pair more significant. In this way, with high ϵ, a more accurate pruning judgement result in AHEAD or a more fine-grained grids in HDG can play an important role in correlation capturing, which makes them obtain good accuracy on Adult and Salaries. In spite of this, we can see that the utility of privNUD is still comparable to the competitors except that in Fig. 8(d). The reason for this outlier is that high attribute correlations and large scale of users on Bfive contribute to AHEAD finding a better decomposition for 2-D domains, further more accurate estimations on 2-D domains. On the other hand, with the increasing of query dimensionality, more answers to 2-D range queries are involved. Further, the superiority of AHEAD on 5-D range queries is reflected more clearly. In addition, we also observe that PRISM performs better than the hierarchical tree-based method AHEAD in some cases on the multidimensional datasets. That benefits from the attribute selection strategy for constructing prefix-sum cubes on partial attribute pairs, instead of all possible pairs. Note that Fig.8 does not show the performance of PRISM. That is because when estimating λ-D range queries with 1-D and 2-D prefix-sum cubes, PRISM needs to generate an intermediate matrix with size $g^\lambda \times g^\lambda$. When $\lambda = 5$ and $g = 10$, it requires a large size of memory and leads to memory overflow.

Evaluations on high-dimensional datasets. Fig. 9(a) reports

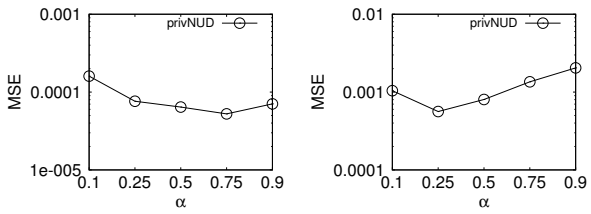
the MSEs of 4-D range queries for comparing privNUD against all the competitors on the high dimensional dataset Ipums, which is used in the work [26] for evaluating PRISM. This dataset contains 30 attributes, each of which is with the domain size as 30. Here, we also use the same budget setting $\{1, 2, 3, 4, 5\}$ as that in Ref. [26]. Not surprisingly, it is shown that privNUD still outperforms the other competitors on the high dimensional dataset on all ϵs. Besides, it is worth noting that PRISM consistently performs better than the other competitors on Ipums, instead of in some cases on the low-dimensional datasets shown in Fig. 6-Fig. 8. That is because for high dimensional datasets, the attribute selection strategy in PRISM can remove more attributes for 2-D prefix-sum cubes construction, which makes it obtain good accuracy. In spite of this, we can see that the utility of privNUD is still superior to PRSIM. Fig. 9(b) shows the MSEs of range queries by varying the query dimensionality λ on Ipums with ϵ as 1. It is observed that the MSEs of all methods are inversely proportional to λs. The main reason is that a higher query dimensionality means more noisy estimations for 1-D and 2-D queries take part in estimating λ-D range queries. With the "weight update" technique, that can be thought more consistency constraints are used to derive the estimations of λ-D queries, which leads to query accuracy improvement.

C. Impact of Different Parameters

This section evaluates the performance of our proposal and competitors on the datasets with different dimensionalities, attribute correlations and parameter αs respectively.

The impact of dimensionality. In this part, we use synthetic datasets following multidimensional Normal or Laplace distribution with correlation coefficient $cov = 0.2$ to evaluate this impact on $\epsilon = 1$, where the coefficient setting better reflects the correlations in real datasets as shown in Tables II- V in Appendix.D. Fig.10 illustrates the MSEs of five methods by varying dimensionality d of datasets. Not surprisingly, privNUD consistently outperforms the other methods. Besides, we observe the performance of AHEAD drops sharply with the increment of d, while that of grids-based or prefix-sum-based methods changes slightly. The main reason is that with the increment of d, users need to be split into more groups, each of which is involved in frequency estimations for a 2-D domain. Further, AHEAD splits each group of users into h smaller sub-groups for computing frequency estimations on the sub-domains in one level, where h is 6 on the datasets in this figure. Instead of that, grid-based methods just use the whole users in one group to derive the estimations on a 2-D grid. Obviously, increased dimensionality has a bigger effect on the number of users involved in the frequency estimation of a sub-domain in AHEAD, which leads to heavy performance penalty. As for PRISM, it adopts attribute selection strategy to remove some attributes and reduce the number of 2-D prefix-sum cubes, which weakens the effect of increased dimensionalities.

The impact of correlation. This section studies the impact of the correlation between two attributes measured by covariance



(a) Adult (b) Ipums

Fig. 13. Result accuracy for 4-D range queries with $\epsilon = 1$ by varying α.

based on the synthetic datasets with 5 attributes. As shown in Fig. 11 with 3-D range queries, the MSE of privNUD is two orders of magnitude lower than that of AHEAD or PRISM. Compared with HDG, privNUD has obvious performance improvement in the low covariance region, but the improvement is slight for a large covariance. Fig. 12 shows the results with 5-D range queries. It is observed that these methods perform similarly to that in Fig. 11, except that AHEAD shows a better result accuracy with high covariance. Such phenomena are consistent with those in Section V-B. Besides, it also describes that the margin between HGD and privNUD is larger on the datasets with more users in small covariance. The main reason is that smaller covariance means a lower attribute correlation, which makes the accuracy of frequency estimations on 1-D dominate the accuracy of multidimensional queries. And more users reinforce the benefit of our proposed dynamic decomposition strategy, which leads to significant accuracy improvement.

The impact of α. In this set of experiments, we take the low-dimensional dataset Adult and high-dimensional dataset Ipums as examples to evaluate the effectiveness of guideline for α setting. Fig. 13 reports the MSEs of privNUD by varying α, where α is set as 75% on Adult and 25% on Ipums by our designed guideline respectively. It is shown that the MSEs are concave with the increase of α and our guideline can help privNUD to achieve the result accuracy close to that in "sweet spot", which means that the guideline for α setting can work well under the datasets with different dimensionalities.

VI. CONCLUSION

This paper investigates the problem of range queries under ϵ-LDP. We propose privNUD which dynamically decomposes each domain with a tailored granularity into some sub-domains, by considering their potential chances to answer one range query. In the future, we plan to publish multiple versions of 2-D grids with different granularities to capture attribute correlations better while consuming less privacy budget.

VII. ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61902365 and 61902366), and Open Project Program from Key Lab of Cryptologic Technology and Information Security, Ministry of Education, Shandong University.

REFERENCES

[1] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067. ACM, 2014.

[2] Apple differential privacy team. Learning with privacy at scale. https://machinelearning.apple.com/research/learning-with-privacy-at-scale, 2017.

[3] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *NIPS*, pages 3571–3580, 2017.

[4] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[5] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In *USENIX Security Symposium*, pages 729–745. USENIX Association, 2017.

[6] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438, 2013.

[7] Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. Understanding hierarchical methods for differentially private histograms. *Proc. VLDB Endow.*, 6(14):1954–1965, 2013.

[8] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proc. VLDB Endow.*, 3(1):1021–1032, 2010.

[9] Jun Zhang, Xiaokui Xiao, and Xing Xie. Privtree: A differentially private algorithm for hierarchical decompositions. In *SIGMOD Conference*, pages 155–170. ACM, 2016.

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

[11] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *IEEE Trans. Knowl. Data Eng.*, 23(8):1200–1214, 2011.

[12] Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *VLDB J.*, 24(6):757–781, 2015.

[13] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. Optimizing error of high-dimensional statistical queries under differential privacy. *Proc. VLDB Endow.*, 11(10):1206–1219, 2018.

[14] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. Answering range queries under local differential privacy. *Proc. VLDB Endow.*, 12(10):1126–1138, 2019.

[15] Tianhao Wang, Bolin Ding, Jingren Zhou, Cheng Hong, Zhicong Huang, Ninghui Li, and Somesh Jha. Answering multi-dimensional analytical queries under local differential privacy. In *SIGMOD Conference*, pages 159–176. ACM, 2019.

[16] Linkang Du, Zhikun Zhang, Shaojie Bai, Changchang Liu, Shouling Ji, Peng Cheng, and Jiming Chen. AHEAD: adaptive hierarchical decomposition for range query under local differential privacy. In *CCS*, pages 1266–1288. ACM, 2021.

[17] Jianyu Yang, Tianhao Wang, Ninghui Li, Xiang Cheng, and Sen Su. Answering multi-dimensional range queries under local differential privacy. *Proc. VLDB Endow.*, 14(3):378–390, 2020.

[18] Ryan McKenna, Raj Kumar Maity, Arya Mazumdar, and Gerome Miklau. A workload-adaptive mechanism for linear queries under local differential privacy. *Proc. VLDB Endow.*, 13(11):1905–1918, 2020.

[19] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. Marginal release under local differential privacy. In *SIGMOD Conference*, pages 131–146. ACM, 2018.

[20] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A. McCann, and Philip S. Yu. Lopub: High-dimensional crowdsourced data publication with local differential privacy. *IEEE Trans. Inf. Forensics Secur.*, 13(9):2151–2166, 2018.

[21] Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, and Jiming Chen. CALM: consistent adaptive local marginal for marginal release under local differential privacy. In *CCS*, pages 212–229. ACM, 2018.

[22] Gaoyuan Liu, Peng Tang, Chengyu Hu, Chongshi Jin, and Shanqing Guo. Multi-dimensional data publishing with local differential privacy. In Julia Stoyanovich, Jens Teubner, Nikos Mamoulis, Evaggelia Pitoura, and Jan Mühlig, editors, *EDBT*, pages 183–194, 2023.

[23] S. L. Warner. Randomised response: a survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[24] A.G. Thakurta, A.H. Vyrros, U.S. Vaishampayan, G. Kapoor, J. Freudinger, V.V. Prakash, A. Legendre, and S. Duplinsky. Emoji frequency detection and deep link frequency, July 11 2017. US Patent 9,705,908.

[25] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *CCS*, pages 192–203. ACM, 2016.

[26] Yufei Wang and Xiang Cheng. PRISM: prefix-sum based range queries processing method under local differential privacy. In *ICDE*, pages 433–445. IEEE, 2022.

[27] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory Comput.*, 8(1):121–164, 2012.

[28] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In *NIPS*, pages 2348–2356, 2012.

[29] Kaggle. Sf salaries. https://www.kaggle.com/datasets/kaggle/sf-salaries, 2017.

[30] Dheeru Dua and Casey Graf. Machine learning repository. http://archive.ics.uci.edu/ml.

[31] Kaggle. All lending club loan data. https://www.kaggle.com/wordsforthewise/lending-club.

[32] Kaggle. Big five personality test. https://www.kaggle.com/datasets/tunguz/big-five-personality-test.

[33] Matthew Sobek and Steven Ruggles. The ipums project: An update. *Historical Methods: A Journal of Quantitative and Interdisciplinary History*, 32(3):102–110, 1999.

APPENDIX

A. The error variance comparison between the hierarchical decomposition-based technique and PRISM.

In the following, we analyze the query errors of PRISM and PrivNUD on the one-dimensional dataset theoretically to illustrate the prefix-sum-based method is not a good choice compared with the hierarchical decomposition method. Given a single attribute with domain $|D|$, the error variance of one noisy prefix-sum estimation on this attribute provided by PRISM is $\frac{\exp(\frac{\epsilon}{g})}{n\left(\exp(\frac{\epsilon}{g})-1\right)^2}$. Since any range query is answered by two prefix-sum estimations, the error variance of any query on one-dimensional dataset is $\frac{2\exp(\frac{\epsilon}{g})}{n\left(\exp(\frac{\epsilon}{g})-1\right)^2}$, where $g=\sqrt{0.2n\epsilon}$. As for our proposal privNUD, it is not an easy thing to derive a formula to express its error variance, since the fanout of each node constructed by privNUD is adaptive to its associated domain and not a constant. On the other hand, privNUD is an optimizer of a uniform hierarchical decomposition technique and can yield a decomposition solution providing more accurate query results. So we use the error from a uniform hierarchical decomposition technique as the upper bound of privNUD's error variance. Let b be the decomposition granularity (fanout) of the uniform hierarchical decomposition technique. Then the error variance of each node's noisy frequency estimation is $\frac{4\log_b|D|\exp(\epsilon)}{n(\exp(\epsilon)-1)^2}$ with the LDP protocol OUE [5]. Besides, with such tree structure, a range query can be answered by $2\log_b|D|\cdot(b-1)$ nodes at most. So the error variance of one query is $\frac{8\log_b^2|D|\exp(\epsilon)(b-1)}{n(\exp(\epsilon)-1)^2}$ at the worst case. Under the above analysis, Table I shows the error variances of one query result provided by privNUD and PRISM with different ns, $|D|$s and ϵs, where the former and latter numbers denote the errors of these two methods respectively. Here, we set b as 4. Obviously, the error at the worst case for a hierarchical decomposition-based technique is smaller than the error for

PRISM in one-dimensional dataset on all settings except when $n = 10^4$, $D = 2^{10}$ and $\epsilon = 0.1$.

B. The proof of Theorem IV.1.

Based on Alg. 1, we know that a user can be involved in estimating the frequency estimations of multiple nodes, in which any two nodes are not located in one path along $\mathcal{T}$. That is decided by Line 18 in Alg. 1. Given a user u_i, let $N(u_i)$ be the set of nodes whose frequency estimations involve u_i. Let $I_v(t)$ be an indicator function with value as 1 when t is located in the domain associated with v; otherwise, 0. For any two possible data t and t' of u_i, and any output sequence o consisting of the outputs when estimating the frequencies of nodes in $N(u_i)$, where o_v denotes the output of u_i for estimating the frequency for node v, we have

$$\begin{aligned}\frac{\prod_{v\in N(u_i)}\Pr[o_v|I_v(t)]}{\prod_{v\in N(u_i)}\Pr[o_v|I_v(t')]} &= \frac{\Pr[o_v|I_v(t)]\cdot\Pr[o_{v'}|I_{v'}(t)]}{\Pr[o_v|I_v(t')]\cdot\Pr[o_{v'}|I_{v'}(t')]}\\ &\le \frac{\Pr[I_v(t)|I_v(t)]\cdot\Pr[I_{v'}(t)|I_{v'}(t)]}{\Pr[I_v(t)|I_v(t')]\cdot\Pr[I_{v'}(t)|I_{v'}(t')]}\\ &= \left[\frac{1}{2}\cdot\frac{e^\epsilon}{e^\epsilon+1}\right]\Big/\left[\frac{1}{2}\cdot\frac{1}{e^\epsilon+1}\right]\\ &= e^\epsilon\end{aligned} \tag{5}$$

Therefore, this theorem is proved.

C. Weight update technique for answering λ-dimensional range queries.

Given the λ-D range queries $q = (A_{\phi_1}, [l_{\phi_1}, r_{\phi_1}]) \wedge (A_{\phi_2}, [l_{\phi_2}, r_{\phi_2}]) \wedge \cdots \wedge (A_{\phi_\lambda}, [l_{\phi_\lambda}, r_{\phi_\lambda}])$, we adopt the same method "weight update technique" used in Ref. [17] to derive the answer to q by depending on the response matrixes for 2-D queries. Specifically, letting $AS_q = \{A_{\phi_1}, A_{\phi_2}, ..., A_{\phi_\lambda}\}$, we firstly generate C_λ^2 2-D range queries from q, which is expressed by

$$\{q^{(j,k)} = (A_j, [l_j, r_j]) \wedge (A_k, [l_k, r_k]) | A_j, A_k \in AS_q\},$$

and then derive the answers to these queries $\{f_{q^{(j,k)}}|A_j, A_k \in AS_q\}$ based on the response matrixes constructed by Step 3. Following that, C_λ^2 answers are involved in updating the values of one vector $\mathbf{z}$ with size 2^λ. Each bit in $\mathbf{z}$ corresponds to a λ-D range queries from set $Q(q) = \{\wedge_t(A_t, [l_t, r_t] \text{ or } [l_t, r_t]') | A_t \in AS_q\}$, where the range $[l_t, r_t]'$ denotes the complement of $[l_t, r_t]$ on A_t's domain. In this way, after the updating, we regard the value from the bit associated with q as the answer to q.

Alg. 2 describes the updating process for $\mathbf{z}$. Specifically, each element in $\mathbf{z}$ is initialized to $\frac{1}{2^\lambda}$. Then each answer $f_{q^{(j,k)}}$ to query $q^{(j,k)}$ from the associated 2-D queries is traversed to update elements in $\mathbf{z}$, in order to make the sum of some elements equal to $f_{q^{(j,k)}}$ (Lines 3-8). Firstly, this algorithm finds the set of queries $Q(q)^{(j,k)}$ which contains $2^{\lambda-2}$ λ-D queries and is defined as $\{\wedge_t(A_t, [l_t, r_t] \text{ or } [l_t, r_t]') \wedge (A_j, [l_j, r_j]) \wedge (A_k, [l_k, r_k]) | A_t \in AS_q / \{A_j, A_k\}\}$. Then it

Algorithm 2: Updating the vector $\mathbf{z}$

Input : The answers to C_λ^2 2-D range queries $\{f_{q^{(j,k)}}|A_j, A_k \in AS_q\}$
Output: vector $\mathbf{z}$ with size 2^λ

1 Initialize all 2^λ elements in the vector $\mathbf{z}$ as $\frac{1}{2^\lambda}$;
2 **repeat**
3 **for** *each* $f_{q^{(j,k)}} \in \{f_{q^{(j,k)}}|A_j, A_k \in AS_q\}$ **do**
4 Find the set of queries $Q(q)^{(j,k)}$ corresponding to $q^{(j,k)}$;
5 Calculate $Y = \sum_{q'\in Q(q)^{(j,k)}} \mathbf{z}[q']$;
6 **if** $Y \neq 0$ **then**
7 **for** *each query* $q' \in Q(q)^{(j,k)}$ **do**
8 $\mathbf{z}[q'] \leftarrow \frac{\mathbf{z}[q']}{Y} \cdot f_{q^(j,k)}$;
9 **until** *convergence*
10 **return** $\mathbf{z}[q']$.

sums the elements in the bits of $\mathbf{z}$ associated with these $2^{\lambda-2}$ queries as Y. Finally, the elements from these bits are updated by $f_{q^{(j,k)}}$ with weight $\frac{\mathbf{z}[q']}{Y}$, where $\mathbf{z}[q']$ denote the current element value associated with query $q' \in Q(q)^{(j,k)}$. The above process is repeated until the sum of the changes of all elements in $\mathbf{z}$ after each update process is lower than a given threshold.

D. The correlations in real datasets.

Tables II-V show the attribute correlations on datasets Adult, Bfive, Loan and V respectively.

TABLE II
CORRELATION BETWEEN ATTRIBUTES OF ADULT

	A1	***A2***	***A3***	***A4***	***A5***
A1	1	-0.0431	0.0007	-0.0110	-0.0177
A2	-0.0431	1	0.1199	0.0800	0.1458
A3	0.0007	0.1199	1	-0.0298	0.0755
A4	-0.0110	0.0800	-0.0298	1	0.0538
A5	-0.0177	0.1458	0.0755	0.0538	1

TABLE III
CORRELATION BETWEEN ATTRIBUTES OF BFIVE

	A1	***A2***	***A3***	***A4***	***A5***
A1	1	0.1703	0.1514	0.1797	0.1359
A2	0.1703	1	0.1877	0.1987	0.1685
A3	0.1514	0.1877	1	0.1868	0.1752
A4	0.1797	0.1987	0.1868	1	0.1813
A5	0.1359	0.1685	0.1752	0.1813	1

TABLE IV
CORRELATION BETWEEN ATTRIBUTES OF LOAN

	A1	***A2***	***A3***	***A4***	***A5***
A1	1	0.0976	0.9451	0.0021	0.6668
A2	0.0976	1	0.1238	-0.0016	0.0999
A3	0.9451	0.1238	1	0.0014	0.6649
A4	0.0021	-0.0016	0.0014	1	-0.0002
A5	0.6668	0.0999	0.6649	-0.0002	1

TABLE I
ERROR VARIANCE FROM PRISM VS. ERROR VARIANCE FROM THE HIERARCHICAL DECOMPOSITION-BASED TECHNIQUE

n, $\lvert D\rvert$ \ ϵ	***0.1***	***0.3***	***0.5***	***0.7***	***0.9***	***1.1***	***1.3***	***1.5***
10^4, 2^5	3.92, 2.16	1.28, 0.24	0.77, 0.08	0.42, 0.04	0.25, 0.02	0.17, 0.02	0.12, 0.01	0.09, 0.01
10^4, 2^8	3.92, 3.84	1.28, 0.42	0.77, 0.15	0.56, 0.08	0.44, 0.04	0.35, 0.03	0.30, 0.02	0.26, 0.01
10^4, 2^{10}	**3.92, 6.00**	1.28, 0.66	0.77, 0.24	0.56, 0.12	0.44, 0.07	0.35, 0.04	0.30, 0.03	0.26, 0.02
10^5, 2^5	2.05, 0.22	0.23, 0.02	0.08, 0.01	0.04, 0.004	0.03, 0.002	0.02, 0.002	0.01, 0.001	0.01, 0.001
10^5, 2^8	3.87, 0.38	1.32, 0.04	0.80, 0.02	0.57, 0.01	0.44, 0.004	0.36, 0.003	0.31, 0.002	0.27, 0.001
10^5, 2^{10}	3.87, 0.60	1.32, 0.07	0.80, 0.02	0.57, 0.01	0.44, 0.01	0.36, 0.004	0.31, 0.003	0.27, 0.002
2×10^5, 2^5	1.02, 0.11	0.11, 0.01	0.04, 0.004	0.02, 0.002	0.01, 0.001	0.01, 0.001	0.01, 0.001	0.005, 0.0004
2×10^5, 2^8	3.97, 0.19	1.32, 0.02	0.80, 0.01	0.57, 0.004	0.44, 0.002	0.36, 0.001	0.31, 0.001	0.26, 0.001
2×10^5 ,2^{10}	3.97, 0.30	1.32, 0.03	0.80, 0.01	0.57, 0.01	0.44, 0.003	0.36, 0.002	0.31, 0.002	0.26, 0.001
3×10^5, 2^5	0.68, 0.07	0.08, 0.01	0.03, 0.003	0.01, 0.001	0.01, 0.001	0.01, 0.001	0.004, 0.0004	0.003, 0.0003
3×10^5, 2^8	3.95, 0.13	1.33, 0.01	0.80, 0.005	0.57, 0.003	0.44, 0.001	0.36, 0.001	0.26, 0.001	0.19, 0.0005
3×10^5, 2^{10}	3.95, 0.20	1.33, 0.02	0.80, 0.008	0.57, 0.004	0.44, 0.002	0.36, 0.001	0.31, 0.001	0.27, 0.001
4×10^5, 2^5	0.51, 0.05	0.06, 0.01	0.02, 0.002	0.01, 0.001	0.01, 0.001	0.004, 0.0004	0.003, 0.0003	0.002, 0.0002
4×10^5, 2^8	3.96, 0.10	1.32, 0.01	0.80, 0.004	0.57, 0.002	0.40, 0.001	0.27, 0.001	0.19, 0.0005	0.15, 0.0004
4×10^5, 2^{10}	3.96, 0.15	1.32, 0.02	0.80, 0.006	0.57, 0.003	0.44, 0.002	0.36, 0.001	0.31, 0.001	0.27, 0.001
5×10^5, 2^5	0.41, 0.04	0.05, 0.005	0.02, 0.002	0.01, 0.001	0.01, 0.0005	0.003, 0.0003	0.002, 0.0002	0.002, 0.0002
5×10^5, 2^8	4.00, 0.08	1.33, 0.01	0.80, 0.003	0.53, 0.002	0.32, 0.001	0.22, 0.001	0.16, 0.0004	0.12, 0.0003
5×10^5, 2^{10}	4.00, 0.12	1.33, 0.01	0.80, 0.005	0.57, 0.002	0.44, 0.001	0.36, 0.001	0.31, 0.001	0.27, 0.0004
6×10^5, 2^5	0.34, 0.04	0.04, 0.004	0.01, 0.001	0.01, 0.001	0.004, 0.0004	0.003, 0.0003	0.002, 0.0002	0.002, 0.0001
6×10^5, 2^8	3.96, 0.06	1.32, 0.01	0.79, 0.003	0.45, 0.001	0.27, 0.001	0.18, 0.0005	0.13, 0.0003	0.10, 0.0002
6×10^5, 2^{10}	3.96, 0.10	1.32, 0.01	0.79, 0.004	0.57, 0.002	0.44, 0.001	0.36, 0.001	0.31, 0.001	0.27, 0.0004
7×10^5, 2^5	0.29, 0.03	0.03, 0.003	0.01, 0.001	0.01, 0.001	0.004, 0.0004	0.002, 0.0002	0.002, 0.0002	0.001, 0.0001
7×10^5, 2^8	3.98, 0.05	1.32, 0.01	0.75, 0.002	0.38, 0.001	0.23, 0.001	0.15, 0.0004	0.11, 0.0003	0.08, 0.0002
7×10^5, 2^{10}	3.98, 0.09	1.32, 0.01	0.80, 0.003	0.57, 0.002	0.44, 0.001	0.36, 0.001	0.31, 0.0004	0.27, 0.0003
8×10^5, 2^5	0.26, 0.03	0.03, 0.003	0.01, 0.001	0.01, 0.001	0.003, 0.0003	0.002, 0.0002	0.002, 0.0001	0.001, 0.0001
8×10^5, 2^8	3.97, 0.05	1.33, 0.01	0.66, 0.002	0.33, 0.001	0.20, 0.001	0.14, 0.0004	0.10, 0.0002	0.07, 0.0002
8×10^5, 2^{10}	3.97, 0.07	1.33, 0.01	0.80, 0.003	0.57, 0.001	0.44, 0.001	0.36, 0.001	0.31, 0.0004	0.27, 0.0003
9×10^5, 2^5	0.23, 0.02	0.03, 0.003	0.01, 0.001	0.005, 0.0005	0.003, 0.0003	0.002, 0.0002	0.001, 0.0001	0.001, 0.0001
9×10^5, 2^8	3.99, 0.04	1.33, 0.005	0.58, 0.002	0.30, 0.001	0.18, 0.0005	0.12, 0.0003	0.09, 0.0002	0.06, 0.0002
9×10^5, 2^{10}	3.99, 0.07	1.33, 0.01	0.80, 0.003	0.57, 0.001	0.44, 0.001	0.36, 0.0005	0.31, 0.0003	0.27, 0.0002
10^6, 2^5	0.20, 0.02	0.02, 0.002	0.01, 0.001	0.004, 0.0004	0.003, 0.0002	0.002, 0.0002	0.001, 0.0001	0.001, 0.0001
10^6, 2^8	3.98, 0.04	1.32, 0.004	0.52, 0.002	0.27, 0.001	0.16, 0.0004	0.11, 0.0003	0.08, 0.0002	0.06, 0.0001
10^6, 2^{10}	3.98, 0.06	1.32, 0.01	0.80, 0.002	0.57, 0.001	0.44, 0.001	0.36, 0.0004	0.31, 0.0003	0.27, 0.0002

TABLE V
CORRELATION BETWEEN ATTRIBUTES OF SALARIES

	A1	***A2***	***A3***	***A4***	***A5***
A1	1	0.2565	0.2393	0.9524	0.9451
A2	0.2565	1	0.2680	0.4933	0.4555
A3	0.2393	0.2680	1	0.4215	0.3735
A4	0.9524	0.4933	0.4215	1	0.9756
A5	0.9451	0.4555	0.3735	0.9756	1