Multi-Dimensional Data Publishing with Local Differential Privacy

Gaoyuan Liu †1, Peng Tang ‡1*, Chengyu Hu ‡2, Chongshi Jin †2, Shanqing Guo ‡3

Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Shandong University, Qingdao, Shandong, 266237, China

School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, 266237, China

†{gaoyuan_liu[1], jinchongshi[2]}@mail.sdu.edu.cn, ‡{tangpeng[1], hcy[2], guoshanqing[3]}@sdu.edu.cn

ABSTRACT

This paper studies the publication of multi-dimensional data with local differential privacy (LDP). This problem raises tremendous challenges in terms of both computational efficiency and data utility. The state-of-the-art solution addresses this problem by first constructing a junction tree (a kind of probabilistic graphical model, PGM) to generate a set of noisy low-dimensional marginals of the input data and then using them to approximate the distribution of the input dataset for synthetic data generation. However, there are two severe limitations in the existing solution, i.e., calculating a large number of attribute pairs' marginals to construct the PGM and not solving well in calculating the marginal distribution of large cliques in the PGM, which degrade the quality of synthetic data. To address the above deficiencies, based on the sparseness of the constructed PGM and the divisibility of LDP, we first propose an incremental learning-based PGM construction method. In this method, we gradually prune the edges (attribute pairs) with weak correlation and allocate more data and privacy budgets to the useful edges, thereby improving the model's accuracy. In this method, we introduce a high-precision data accumulation technique and a low-error edge pruning technique. Second, based on joint distribution decomposition and redundancy elimination, we propose a novel marginal calculation method for the large cliques in the context of LDP. Extensive experiments on real datasets demonstrate that our solution offers desirable data utility.

1 INTRODUCTION

Differential privacy (DP) [14, 15] has been increasingly accepted as the *de facto* standard for data privacy in the research community. Recently, techniques for satisfying differential privacy in the local setting, which is referred to as LDP, have been deployed. In the local setting for DP, there are many users and one aggregator. Unlike the centralized setting, the aggregator does not see the actual private data of each individual. Instead, each user sends randomized information to the aggregator, who attempts to infer the data distribution. LDP techniques enable the gathering of statistics while preserving the privacy of every user without relying on trust in a single trusted third party. Previous works on LDP focus on estimating the frequencies of frequent values the user possesses [6, 16, 30, 37, 39, 40]. The natural and more general setting is when each user has multiple attributes, and the aggregator is interested in the joint distribution of these attributes. With multi-dimensional, a huge amount of potential information and patterns behind the data can be mined or extracted to provide accurate dynamics and reliable predictions for both groups and individuals[25].

Following the works of publishing multi-dimensional data under differential privacy in the centralized setting (CDP), Ren et al. [33] proposed the LoPub method to publish multi-dimensional data under LDP, which first constructs a junction tree (a kind of probabilistic graphical model, PGM) based on the noisy aggregated information to generate a set of noisy low-dimensional marginals of the input data, and then use them to approximate the distribution of the input dataset for synthetic data generation. However, there are two severe limitations in LoPub: (i) LoPub needs to calculate a vast amount of attribute pairs' correlations under LDP to construct the PGM. This leads to injecting a prohibitive amount of noise into the constructed PGM and degrading the resulting synthetic data quality. (ii) This method does not solve well in calculating the marginal distribution of large cliques in the PGM and still faces high dimensionality.

To address the first deficiency, we observe that the constructed PGM is often *sparse*, i.e., although there exist a large number of attribute pairs in the multi-dimensional data, only the attribute pairs with strong correlations have dependencies in the constructed graph. And we find that local differential privacy satisfies *divisibility*. Specifically, in LDP, the users perturb their data locally. For a group of users, to estimate some frequencies, (i) in one way, the aggregate can let all users upload their perturbed data at the same time, and (ii) in another way, we can divide these users into multiple batches, let them upload their perturbed data in order by batch, and then accumulate the aggregated results continuously to get the aggregation of all users. On the one hand, both methods can get the same result. On the other hand, by uploading users' perturbed data in order by batch, the aggregate can obtain some helpful information in advance to guide subsequent estimations (please see more detailed discussion in Subsection 5.2.1). Based on the sparseness of the constructed PGM and the divisibility of LDP, we propose an incremental learning-based PGM construction method. This incremental learning-based method has two obvious advantages. (i) It can gradually prune the edges (attribute pairs) with weak correlations and allocate more data and privacy budget to the useful edges. (ii) Based on the divisibility of the local differential privacy, it will not reduce the accuracy of the correlations of the valid attribute pairs due to uploading the users' perturbed data in batches. Besides, in this method, we introduce a *high-precision data accumulation* technique and a *low-error edge pruning* technique. For the second deficiency, based on joint distribution decomposition and redundancy elimination, we propose a novel marginal calculation method for the large cliques in the context of LDP. Extensive experiments on real datasets demonstrate that our solution offers desirable data utility.

Our key contributions are summarized as follows.

*Peng Tang is corresponding author.

- We first propose an incremental learning-based method for constructing the probabilistic graph model in publishing the multi-dimensional data with LDP.
- To better perform the incremental learning-based PGM construction method, we introduce a high-precision data accumulation technique and a low-error edge pruning technique.
- Based on joint distribution decomposition and redundancy elimination, we propose a novel marginal calculation method for the large cliques under LDP.
- Finally, we conduct an extensive experimental study over several real datasets. The experimental results suggest that our methods are practical to offer desirable data utility.

Roadmap. The remainder of this paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we introduce the problem statement and review the primitives that underlie our proposed schemes. In Section 4, we review the existing solution and point out its limitations. In Section 5, we introduce our PrivIncr solution, including the overview, the key techniques, and the privacy and complexity analysis. We show our experimental results in Section 6. Finally, we provide concluding remarks and future work in Section 7.

2 RELATED WORK

The research on multi-dimensional data publishing under centralized differential privacy (CDP) [15] has been fruitful. The state-of-the-art solution utilizes probabilistic graphical models to solve this problem. In particular, Zhang et al. [43] propose PrivBayes, which constructs a differentially private Bayesian network to approximate the full-dimensional distribution and then uses it to generate synthetic data. Chen et al. [9] propose JTree, which builds a junction tree by identifying all attributes' pairwise independence and then generates synthetic datasets by a sampling-based algorithm from the noisy marginals generated from the junction tree algorithm. Mckenna et al. [28] propose DP-PGM, which shows how to use some measurements over low-dimensional marginals to construct a Markov network and then infer any dimensional distribution without privacy budget consumption. However, DP-PGM has not provided an automatic method to identify low-dimensional marginals [45]. To solve this problem, following DP-PGM, Cai et al. [8] propose PrivMRF. It first constructs a junction tree with large cliques and utilizes *CFS* to select one marginal for each attribute from all the cliques' attribute subsets that satisfy θ-useful, and then gradually chooses the items with more additional information to optimize the model. Zhang et al. [45] propose PrivSyn. It first selects a set of large numbers of low-degree marginals and generates a random dataset where each attribute matches one-way marginals in the set. After that, gradually update the generated dataset to be consistent with all noisy marginals in the set. Meanwhile, there also exist some other data publishing methods based on alternative techniques under CDP, including the game based methods [19, 20, 35], the GAN based methods [2, 7, 18], the GNN based method [1], and the Copula Functions based method [24]. However, in comparison with the probabilistic graphical-based methods, the computational efficiency or performance of these methods are unsatisfactory [45].

In the local setting, one fundamental problem is that all data is distributed locally, so no user has global statistical information. This poses a greater challenge to learning the distribution of multi-dimensional data. Most existing research [11, 16, 26, 31, 37] with LDP mainly focuses on frequency (resp. distribution) estimation over a single categorical (resp. numerical) attribute. However, when each user has multiple attributes, the aggregator is interested in the joint distribution of some of these attributes. Fanti et al. [17] propose an expectation-maximization (EM) based method to estimate the joint distribution of any two attributes under LDP, and then generalize by Ren et al. [32] to handle multiple attributes. However, this method runs slowly and has a large variance because of privacy budget needs to be split into each attribute. To publish any k-way marginals under LDP, Kulkarni et al. [10] apply the Fourier-Transformation-based method (FT), which was used in publishing marginals under the centralized DP setting [4], to the local setting. Zhang et al. propose CALM [44], which uses the covering design to partition the attribute set to multiple low-dimensional marginals and then reconstruct any k-way marginals using the synopsis proposed in [29]. Nevertheless, the value of k is specified before construction, so we can't obtain the distributions above the k dimension. Furthermore, we notice that the methods for range queries can be also used to answer marginal release. There also exist some other works focus on specific tasks, e.g., range query and mean estimation. In particular, [11, 12, 38, 42] provide the privacy-preserving range query under LDP. [13, 36] focus on the specific task of mean estimation under LDP. Specifically, Wang et al. [36] propose two mechanisms, PM and HM, to estimate the mean value of each attribute in multi-dimensional data. However, in contrast with the above methods, our goal is to generate a synthetic dataset that approximates the original dataset with local differential privacy to support more data analysis tasks. Most relevantly, Ren et al. propose LoPub [33] to solve the multi-dimensional data publication problem under LDP. However, two limitations degrade the synthetic data quality of LoPub, i.e., calculating a large number of attribute pairs' marginals to construct the PGM and poor performance in calculating the distribution of large cliques. Wang et al. [41] aim to improve the performance of LoPub with copula functions but still face the same limitations.

3 PROBLEM STATEMENT AND PRELIMINARIES

In this section, we first introduce the problem statement. Then, we review the primitives that underlie our proposed schemes, including the local differential privacy definition and protocols, and the Junction tree.

3.1 Problem Statement

In this paper, the problem of *multi-dimensional data publishing with local differential privacy* is defined as follows: assuming that there exist N users and each user has only one record with d attributes, the aggregator collects the records of the N users in the context of local differential privacy (LDP) and generates a synthetic dataset $\mathcal{D}^*$ that has an approximate joint distribution with $\mathcal{D}$ composed of the records of N users.

Specifically, given an attribute set $\mathcal{A} = \{A_1, \ldots, A_d\}$. For each attribute $A_i \in \mathcal{A}$, we denote its domain by $\Omega_{A_i} = \{\omega_1, \ldots, \omega_{|\Omega_{A_i}|}\}$, where $|\Omega_{A_i}|$ is the domain size and ω_i is the i-th possible value of A_i. For any attribute set $S \subseteq \mathcal{A}$, the domain Ω_S is defined by the cartesian product of all items in $\{\Omega_A | A \in S\}$, whose size is $|\Omega_S| = \prod_{A \in S} |\Omega_A|$. Let dataset $\mathcal{D} = \{X^1, \ldots, X^N\}$ be the collection of records of N users, where $X^j = (x_1^j, \ldots, x_d^j)$ is the record of the j-th user. Note that we assume all attributes in $\mathcal{A}$ are categorical attributes, a numerical attribute can be discretized into a categorical attribute by discretizing its domain into a fixed number of equi-width ranges.

3.2 Local Differential Privacy

In the local setting of differential privacy [15], an untrusted aggregator hopes to collect the personal information of users to complete the corresponding data analysis task. Local differential privacy (LDP) [21] provides a randomized response algorithm $\mathcal{A}$ to protect the privacy of users while satisfying the aggregator's data analysis task. LDP can be defined as follows:

Definition 3.1 (Local Differential Privacy). A randomized algorithm $\mathcal{M}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ is ϵ-local differential privacy, where $\epsilon > 0$, if and only if for all possible outputs $\mathcal{S} \subseteq \mathcal{R}$ and for any two inputs $x_1, x_2 \in \mathcal{D}$, have

$$\Pr[\mathcal{M}(x_1) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{M}(x_2) \in \mathcal{S}].$$

3.3 Optimized Unary Encoding

To estimate the frequency for the discrete attribute under ϵ-LDP, Wang et al. [37] proposed the Optimized Unary Encoding (OUE) protocol. The protocol mainly consists of the following parts:

(1) **Encoding:** For a discrete attribute $\mathcal{V}$ with domain $[1, d]$, each user encodes the value $v \in \mathcal{V}$ into a binary vector S of length d like $[0, \ldots, 0, 1, 0, \ldots, 0]$, where only the v-th position is 1, and the others are 0.

(2) **Perturbation:** Each user perturbs the encoding result S bit by bit according to the following perturbation rules:

$$\Pr(\hat{S}[i] = 1) = \begin{cases} p = \frac{1}{2}, & S[i] = 1; \\ q = \frac{1}{e^{\epsilon}+1}, & S[i] = 0, \end{cases} \tag{1}$$

where ϵ is the privacy budget, $S[i]$ and $\hat{S}[i]$ denote the i-th bit of the encoding result S and the perturbation result $\hat{S}$, respectively.

(3) **Aggregation:** After collecting all perturbation reports from n users, for any value $v \in \mathcal{V}$, the aggregator can estimate its frequency $\hat{c}(v)$ by unbiased estimation as:

$$\hat{c}(v) = \frac{\sum_j \mathbb{1}_{\{v|\hat{S}^j[v]=1\}}(v) - nq}{p - q}, \tag{2}$$

where $\mathbb{1}_X(x)$ is the indicator function, and $\hat{S}^j$ represents the j-th user's perturbation report. And the variance for the estimation is:

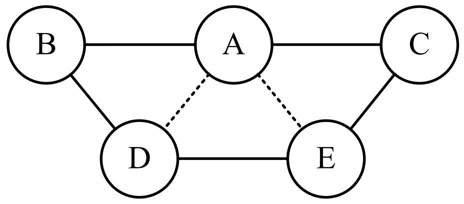
$$Var[\hat{c}(v)] = n \cdot \frac{4e^{\epsilon}}{(e^{\epsilon} - 1)^2}. \tag{3}$$

3.4 The Junction Tree

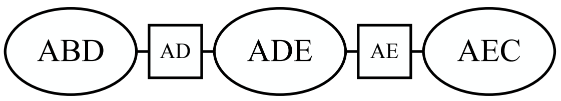
To overcome the curse of dimensionality, the key is to find conditional independence from real-world datasets to factorize the joint probability distribution into modular components. Probabilistic graphical models [23] are an elegant tool for identifying such a modular structure, and Markov networks are the most widely used graphical model based on undirected graphs. As described in [8, 9, 32], the junction tree algorithm [5] provides a feasible method for inferring the joint probability distribution accurately from the Markov network. The junction tree is defined as follows:

Definition 3.2 (Junction Tree). For a given Markov network G, a tree $\mathcal{T} = (\mathcal{C}, \mathcal{S})$ transformed from G is called a junction tree if any pairwise intersection $C_i \cap C_j$ of pairs $C_i, C_j \in \mathcal{C}$ is contained in every node in the unique path in $\mathcal{T}$ between C_i and C_j, where $C_i \in \mathcal{C}$ is a clique in $\mathcal{T}$ and $\mathcal{S}_{ij} = C_i \cap C_j$ is the intersection of two adjacent cliques.

Example 3.3. Figure 1(b) shows a junction tree constructed by the Markov network in Figure 1(a), where oval and rectangle nodes represent cliques and separators respectively, i,e., $\mathcal{C} = \{ABD, ADE, AEC\}$ and $\mathcal{S} = \{AD, AE\}$. Note that the edges linked by dotted lines (i.e., AD and AE) are introduced by the process of the junction tree algorithm.



(a) A markov network



(b) A junction tree

Figure 1: A Markov network and its junction tree

For a dataset $\mathcal{D}$ with attribute set $\mathcal{A} = \{A_1, \ldots, A_d\}$, given the structure of the junction tree $\mathcal{T}$, the joint distributions of the cliques and the separators $\Pr(C_i)$ and $\Pr(S_{ij})$, the full joint distribution of $\mathcal{A}$ can be calculated as:

$$\Pr(\mathcal{A}) = \frac{\prod_{C_i \in \mathcal{C}} \Pr(C_i)}{\prod_{S_{ij} \in \mathcal{S}} \Pr(S_{ij})}. \tag{4}$$

4 EXISTING WORK AND LIMITATIONS

Following the works of multi-dimensional data publishing under centralized differential privacy (CDP) [15], Ren et al. [33] proposed LoPub for multi-dimensional data publishing under local differential privacy (LDP), which consists of the following four phases:

(1) **Data collection:** To support subsequent calculations while satisfying LDP, each user perturbs their records and sends the perturbed results to the aggregator for aggregation.

(2) **Junction tree construction:** The aggregator learns the correlations of all pairwise attributes based on the aggregation (learned in the first phase) and constructs a dependency graph G based on the correlations. Next, the aggregator transforms the dependency graph G into a junction tree $\mathcal{T}$ composed of multiple cliques $\mathcal{C}$.

(3) **Joint distribution estimation of cliques:** The aggregator estimates the joint distribution $\Pr(C)$ of each clique $C \in \mathcal{C}$ (constructed in the second phase) based on the aggregation (learned in the first phase).

(4) **Data generation:** The aggregator generates a synthetic dataset $\mathcal{D}^*$ according to the structure of the junction tree $\mathcal{T}$ (constructed in the second phase) and the joint distribution $\Pr(C)$ of each clique $C \in \mathcal{C}$ (learned in the third phase).

Limitation of LoPub. LoPub has two significant limitations: (i) in the construction of the dependency graph G, LoPub directly calculates all pairwise correlations to determine the structure of the dependency graph. However, for multi-dimensional data, there exist a large number of attribute pairs. Directly calculating all these pairwise correlations will incur massive noise injected into the results to achieve LDP. This severely degrades the accuracy of the dependency graph structure and the synthetic data. (ii) In the estimation of the marginal $\Pr(C)$ of large cliques $C \in \mathcal{C}$, some cliques in $\mathcal{C}$ may contain too many attributes, and LoPub maybe still face high dimensionality when calculating their marginals.

5 OUR SOLUTION

5.1 Overview of Our Solution.

To address the above deficiencies, we propose PrivIncr, a solution that (i) also generates a synthetic dataset $\mathcal{D}^*$ according to a constructed junction tree $\mathcal{T}$ and the joint distribution $\Pr(C)$ of each clique C contained in $\mathcal{T}$, but (ii) provides an incremental learning-based dependency graph construction method to construct the junction tree and a novel marginal calculation method for the large cliques in the context of LDP. In the following, we will first introduce the method to construct the dependency graph (Subsection 5.2) and explain how to estimate the marginals of the cliques (Subsection 5.3). Then, we introduce the overall workflow of our PrivIncr solution (Subsection 5.4) with privacy and complexity analysis (Subsection 5.5).

5.2 Incremental Learning-based Dependency Graph Construction

5.2.1 *The Key Insights behind the Method.*

There are two key insights behind the incremental learning-based dependency graph construction:

The *Sparsity* of the Constructed Dependency Graph. Although there exists a large number of attribute pairs in the multiple dimensional data, in the constructed graph, only the attribute pairs with strong correlations are selected in the constructed dependency graph. For example, in our experiments, there exist 15 (resp. 24) attributes in the dataset *Adult* (resp. *TPC-E*), and $\binom{15}{2}$ (resp. $\binom{24}{2}$) attribute pairs. However, the number of edges in the constructed graph is less than 20, which is much smaller than the number of attribute pairs. Furthermore, we give another example.

Example 5.1. Suppose that given six binary attributes $\{A, B, C, D, E, F\}$, the value of the correlation (i.e., the mutual information, which we will describe in Subsection 5.2.3) for each attribute pair is shown in Figure 2(a). We consider that if the value of correlation is greater than a given threshold (e.g., 0.045, computed by Equation 15), the attribute pair has a strong correlation, and the edge will be selected in the constructed graph. Thus, as shown in Figure 2(b), only four attribute pairs (i.e., $\{AB, AF, BD, CE\}$) have strong correlations and have been selected as the edges in the constructed dependency graph.

The *Divisibility* of the Local Differential Privacy. In LDP, the users perturb their data locally. For a group of users, to estimate some frequencies, (i) in one way, the aggregate can let all users upload their perturbed data at the same time, and (ii) in another way, we can divide these users into multiple batches, let them upload their perturbed data in order by batch, and then accumulate the aggregated results continuously to get the aggregation of all users. On the one hand, both methods can get the same result. On the other hand, by uploading users' perturbed data in order

	A	***B***	***C***	***D***	***E***	***F***
A		**0.0776**	0.0356	0.0378	0.0280	**0.0917**
B	**0.0776**		0.0171	**0.0613**	0.0208	0.0153
C	0.0356	0.0171		0.0144	**0.0712**	0.0267
D	0.0378	**0.0613**	0.0144		0.0433	0.0384
E	0.0280	0.0208	**0.0712**	0.0433		0.0309
F	**0.0917**	0.0153	0.0267	0.0384	0.0309	

(a) The value of all pairwise correlations

	A	***B***	***C***	***D***	***E***	***F***
A		**1**	0	0	0	**1**
B	**1**		0	**1**	0	0
C	0	0		0	**1**	0
D	0	**1**	0		0	0
E	0	0	**1**	0		0
F	**1**	0	0	0	0	

(b) The selected attribute pairs with strong correlations

Figure 2: The *sparsity* of the dependency graph

by batch, the aggregate can obtain some helpful information in advance to guide subsequent estimations.

Property 5.1 (Divisibility of LDP). *Suppose that for a given variable U with the input domain $[u]$, we would like to estimate the frequency of the value $v \in [u]$ over N users under ϵ-local differential privacy. Thus, one can directly use OUE method to compute the result, i.e., $\hat{c}(v) = \frac{\sum_{j\in N} \mathbb{1}_{\{v|\hat{S}^j[v]=1\}}(v) - Nq}{p-q}$ (as shown in Equation 2). We partition these N users into m disjoint groups $\{N_1, \ldots, N_m\}$ and then estimate the frequencies on each group respectively. Thus, we can obtain m results $\left\{\hat{c}_i(v) = \frac{\sum_{j\in N_i} \mathbb{1}_{\{v|\hat{S}_i^j[v]=1\}}(v) - N_i q}{p-q} \mid 1 \le i \le m\right\}$. By accumulating these results, we can learn that:*

$$
\begin{aligned}
\hat{c}(v) &= \frac{\sum_{j\in N} \mathbb{1}_{\{v|\hat{S}_i^j[v]=1\}}(v) - Nq}{p-q} \\
&= \frac{\sum_{i=1}^{m} \sum_{j\in N_i} \mathbb{1}_{\{v|\hat{S}_i^j[v]=1\}}(v) - \sum_{i=1}^{m} N_i q}{p-q} \\
&= \sum_{i=1}^{m} \frac{\sum_{j\in N_i} \mathbb{1}_{\{v|\hat{S}_i^j[v]=1\}}(v) - N_i q}{p-q} \\
&= \sum_{i=1}^{m} \hat{c}_i(v).
\end{aligned}
\tag{5}
$$

In particular, this is different from centralized differential privacy (CDP). In CDP, if the dataset is divided into multiple subsets, noise needs to be injected into the released count of each subset. We can obtain the count of the whole dataset by summing these noisy counts. But the result will contain a larger noise scale than that obtained by directly performing statistics on the overall dataset and injecting noise.

5.2.2 *The Main Idea of the Method.*

Based on the above two insights, we propose an incremental learning-based dependency graph construction method. In this method, we divide all users into multiple groups. First (i.e., in the first round), we let some users upload their data to estimate the correlations of all attribute pairs and prune those with weak correlations. And then (i.e., in the next rounds), for the remaining attribute pairs, we collect some new group of data, accumulate these data to previous data to re-estimate their correlations, and continue to prune the attribute pairs with weak correlations until we construct the dependency graph. This incremental learning-based method has two obvious advantages. First, it can gradually prune the edges (attribute pairs) with weak correlations and allocate more data and privacy budget to the useful edges. Second, based on the divisibility of the local differential privacy, it will not reduce the accuracy of the correlations of the valid attribute pairs due to uploading in batches.

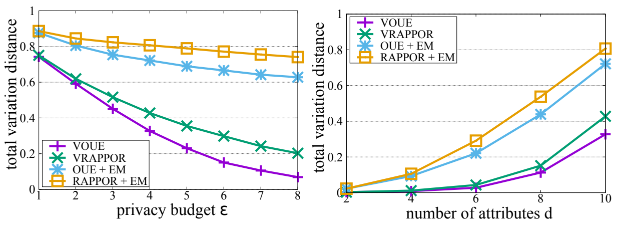
5.2.3 *The Key Steps of the Method.*

To implement the incremental learning-based dependency graph construction method, we need to address the following problems effectively: (i) How to collect users' data in each round and accumulate the data collected in different rounds with high accuracy and efficiency. These data not only include multiple groups of data of the attribute pairs that have not been pruned, but also the data of attribute pairs that have been pruned. (ii) How to prune the useless edges with low error. We use part of the data to estimate the correlation strength of attribute pairs, and there will be some sampling errors (especially in the first few rounds). In addition, in LDP, the smaller the amount of data, the larger the perturbed error. This may cause us to prune the originally highly correlated attribute pairs mistakenly. In the following, we will discuss how to solve the above problems.

Data Aggregation. In our solution, we employ a variant of OUE method to estimate the distribution of each attribute set $\mathcal{A}$ that has a moderate domain size. That is, we treat $\mathcal{A}$ as one attribute with domain size $|\Omega_{\mathcal{A}}| = \prod_{A\in\mathcal{A}} |\Omega_A|$ and employ OUE method to estimate the frequency of each value v in $\Omega_{\mathcal{A}}$; we call this process as VOUE. According to OUE, for any value $v \in \Omega_{\mathcal{A}}$, the variance for its estimation $\hat{c}(v)$ is $Var[\hat{c}(v)] = n_{\mathcal{A}} \cdot \frac{4e^{\epsilon}}{(e^{\epsilon}-1)^2}$ (as shown in Equation 3), where ϵ denotes the privacy budget and $n_{\mathcal{A}}$ denotes the number of users that upload the perturbation reports about $\mathcal{A}$.

To evaluate the performance of VOUE, we compare VOUE with other three methods: (i) VRAPPOR, this is the variant of the one-time basic RAPPOR [16]. Similar with VOUE, we treat $\mathcal{A}$ as one attribute with domain size $|\Omega_{\mathcal{A}}| = \prod_{A\in\mathcal{A}} |\Omega_A|$, and employ the one-time basic RAPPOR method to estimate the frequency of each value v in $\Omega_{\mathcal{A}}$; we call this process as VRAPPOR. (ii) OUE+EM and RAPPOR+EM. In OUE+EM (resp. RAPPOR+EM), we first divide the privacy budget ϵ into $|\mathcal{A}|$ parts, i.e., $\epsilon_i = \frac{\epsilon}{|\mathcal{A}|}$. And then, we employ OUE (resp. one-time basic RAPPOR) to encode and perturb each attribute $A_i \in \mathcal{A}$ under ϵ_i-LDP, respectively. After that, we estimate the joint distribution $\Pr(\mathcal{A})$ by EM algorithm [32]. Figure 3 illustrates the performance of these methods as privacy budget ϵ and the number of attributes d varying. The result shows that VOUE can achieve better accuracy than others in all cases.

Data Accumulation. In the incremental learning-based dependency graph construction, we will continuously collect the data for the unpruned attribute pairs to recalculate their marginal distributions. Accumulating these data can get more accurate results. Besides, for the pruned edge, we have collected some



(a) Varying ϵ ($d = 10$) (b) Varying d ($\epsilon = 4$)

Figure 3: Different distribution estimation methods

of its data in the previous rounds. Can we use these data to improve the accuracy of the correlations of the unpruned attribute pairs? In [44], Zhang et al. proposed an accumulation method. However, applying it to solve our problem will incur high computational complexity. The reason lies in that when we perform a new round of collection, the method will re-accumulate the previous rounds of data. Specifically, we denote the collected data in the i^{th} round as $data_i$ $(1 \le i)$. In the t^{th} round, the accumulation result is calculate as $ar_t = accu\,(data_1, \ldots, data_t)$. To reduce the computational complexity, we propose an efficient accumulation method, which can calculate the accumulation results of the first t rounds based on the accumulation results of the first $(t-1)$ rounds, i.e., $ar_t = accu\,(ar_{t-1}, data_t)$.

Specifically, to accumulate the distribution of attribute set $\mathcal{A}^*$, we first consider the attribute sets that contain $\mathcal{A}^*$ in each round (e.g., the t^{th} round). We denote these attribute sets as $\mathcal{A}_{t1}, \ldots, \mathcal{A}_{ti_t}$, and $\mathcal{A}^* \subseteq \mathcal{A}_{tj}$, where $1 \le j \le i_t$. Let $n_{\mathcal{A}_j}$ denote the number of users that upload the perturbation reports about $\mathcal{A}_j$, $T_{\mathcal{A}_j}$ denote the marginal table of $\mathcal{A}_j$ estimated under LDP, and $T_{\mathcal{A}_j}(v)$ denote the item in $T_{\mathcal{A}_j}$ about $v \in \Omega_{\mathcal{A}_j}$. Thus, when the attribute set $\mathcal{A}$ has a moderate domain size, we employ the VOUE method to estimate its distribution. Based on the marginal tables of $\mathcal{A}_{t1}, \ldots, \mathcal{A}_{ti_t}$, according to Maximum Likelihood Estimation (MLE), we can get the aggregated marginal table of $\mathcal{A}^*$ in the t^{th} round:

$$T^t_{\mathcal{A}^*}(v^*) = \sum_{j=1}^{i_t} \left(\frac{\frac{1}{C_{tj} \cdot n_{\mathcal{A}_{tj}}}}{\sum_{k=1}^{i_t} \frac{1}{C_{tk} \cdot n_{\mathcal{A}_{tk}}}} \cdot \mathcal{A}_{tj}(v^*) \right), \tag{6}$$

where $C_{tj} = \prod_{A\in\mathcal{A}_{tj}/\mathcal{A}^*} |\Omega_A|$ and $\mathcal{A}_{tj}/\mathcal{A}^*$ denotes the set of attributes that contained in $\mathcal{A}_{tj}$ but not in $\mathcal{A}^*$, and $\mathcal{A}_{tj}(v^*)$ is the sum of all $\mathcal{A}_{tj}(v)$ where v is the elements in $\Omega_{\mathcal{A}_{tj}}$ and the value at $\mathcal{A}^*$ is v^*. The variance of the aggregated result is:

$$Var\left[T^t_{A^*}(v^*)\right] = \frac{1}{\sum_{j=1}^{i_t} \frac{1}{C_{tj} \cdot n_{\mathcal{A}_{tj}}}} \cdot Var_0, \tag{7}$$

where $Var_0 = \frac{4e^{\epsilon}}{(e^{\epsilon}-1)^2}$ is the basic variance in Equation 3.

According to Equation 6, we learn that $T^t_{\mathcal{A}^*}(v^*)$ is a weighted average of $\left\{\mathcal{A}_{tj}(v^*) | 1 \le j \le i_t\right\}$. Let $\mathcal{T}^t_{\mathcal{A}^*}$ denote the accumulated marginal table of $\mathcal{A}^*$ of the first t rounds. Thus, $\mathcal{T}^t_{\mathcal{A}^*}$ can be calculated as:

$$\mathcal{T}^t_{\mathcal{A}^*}(v^*) = \widetilde{\omega}_{t-1} \cdot \mathcal{T}^{t-1}_{\mathcal{A}^*}(v^*) + \omega_t \cdot T^t_{\mathcal{A}^*}(v^*), \tag{8}$$

where $\mathcal{T}^1_{\mathcal{A}^*}(v^*) = T^1_{\mathcal{A}^*}(v^*)$. We have the variance of $\mathcal{T}^t_{\mathcal{A}^*}(v^*)$ as follows:

$$Var[\mathcal{T}^t_{\mathcal{A}^*}(v^*)] = \widetilde{\omega}^2_{t-1} \cdot Var[\mathcal{T}^{t-1}_{\mathcal{A}^*}(v^*)] + \omega_t^2 \cdot Var[T^t_{\mathcal{A}^*}(v^*)], \tag{9}$$

where

$$Var[\mathcal{T}^{t-1}_{\mathcal{A}^*}(v^*)] = \frac{1}{\sum_{l=1}^{t-1}\sum_{j=1}^{i_l}\frac{1}{C_{lj}\cdot n_{\mathcal{A}_{lj}}}}\cdot Var_0,$$
$$Var[T^{t}_{\mathcal{A}^*}(v^*)] = \frac{1}{\sum_{j=1}^{i_t}\frac{1}{C_{tj}\cdot n_{\mathcal{A}_{tj}}}}\cdot Var_0. \tag{10}$$

To be more clearly, let

$$\widetilde{\mu}_{t-1} = \sum_{l=1}^{t-1}\sum_{j=1}^{i_l}\frac{1}{C_{lj}\cdot n_{\mathcal{A}_{lj}}}$$
$$\mu_t = \sum_{j=1}^{i_t}\frac{1}{C_{tj}\cdot n_{\mathcal{A}_{tj}}}. \tag{11}$$

We will calculate $\widetilde{\omega}_t$ and ω_t to minimum the variance $Var[\mathcal{T}^t_{\mathcal{A}^*}(v^*)]$ in Equation 9:

$$\begin{aligned}\text{minimize}\quad & \widetilde{\omega}^2_{t-1}\cdot\frac{1}{\widetilde{\mu}_{t-1}} + \omega_t^2\cdot\frac{1}{\mu_t}\\ \text{s.t.}\quad & \widetilde{\omega}_{t-1} + \omega_t = 1.\end{aligned}$$

One can use the method of Lagrange multipliers to solve the above optimization problem, we have

$$\widetilde{\omega}_{t-1} = \frac{\widetilde{\mu}_{t-1}}{\widetilde{\mu}_{t-1}+\mu_t}\ \text{and}\ \omega_t = \frac{\mu_t}{\widetilde{\mu}_{t-1}+\mu_t}. \tag{12}$$

Thus, we have that:

$$\mathcal{T}^t_{\mathcal{A}^*}(v^*) = \frac{\widetilde{\mu}_{t-1}}{\widetilde{\mu}_{t-1}+\mu_t}\cdot\mathcal{T}^{t-1}_{\mathcal{A}^*}(v^*) + \frac{\mu_t}{\widetilde{\mu}_{t-1}+\mu_t}\cdot T^t_{\mathcal{A}^*}(v^*). \tag{13}$$

From the above equation, we can learn the accumulation results of the first t rounds based on the accumulation results of the first $(t-1)$ rounds.

Edge Selection. In each round, we select the edges with strong correlations according to the accumulated marginals. To decrease the error caused by sampling and perturbation, and increase the accuracy of edge selection, we propose a threshold relaxation-based edge pruning method. In particular, we consider the randomness brought by sampling and perturbation to the attribute pairs' correlations and regard the correlations as random variables. Then in each round of incremental learning, we use the hypothesis testing method to calculate a lower bound of the threshold and take the lower bound as a relaxed threshold to prune the edges.

Specifically, in this paper, we use the attribute pairs' mutual information to measure the pairwise correlations. For the given two attributes A_i, A_j, the mutual information I_{A_i,A_j} is defined:

$$I_{A_i,A_j} = \sum_{a_i\in\Omega_{A_i}}\sum_{a_j\in\Omega_{A_j}}\Pr(a_i,a_j)\log\frac{\Pr(a_i,a_j)}{\Pr(a_i)\cdot\Pr(a_j)}, \tag{14}$$

where $\Pr(a_i,a_j)$ is short for the joint distribution $\Pr(A_i=a_i, A_j=a_j)$. Following [9], we use a threshold τ_{A_i,A_j} controlled by the parameter ϕ to determine whether a strong correlation is between A_i and A_j. τ_{A_i,A_j} can be calculated:

$$\tau_{A_i,A_j} = \min\left(|\Omega_{A_i}|-1, |\Omega_{A_j}|-1\right)\times\phi^2/2. \tag{15}$$

A_i and A_j are strongly correlated if $I_{A_i,A_j}\geq\tau_{A_i,A_j}$.

In one round of incremental learning, for an edge (A_i,A_j) in the dependency graph G, we assume that there are n records containing attributes A_i and A_j among the records currently collected. We denote the joint distribution over these n records as $\Pr^n(A_i,A_j)$ and the mutual information as $I^n_{A_i,A_j}$. Because of the randomness caused by sampling and perturbation, $I^n_{A_i,A_j}$ can be treated as a random variable. According to Stefani et al.[34], we can determine the probability distribution of $I^n_{A_i,A_j}$ as shown in Theorem 5.2.

Theorem 5.2. *For any $\alpha\in(0,1]$ and any two attributes A_i, A_j with domain size $M_i = \left|\Omega_{A_i}\right|, M_j = \left|\Omega_{A_j}\right|$ where $M_i\leq M_j$, let*

$$\eta = \left(\frac{2}{n}\ln\frac{2^{M_i\cdot M_j}-2}{\alpha}\right)^{\frac{1}{2}},$$

If $\eta\leq 2-\frac{2}{M_i}$,

$$\Delta I(\eta) = \frac{\eta}{2}\log\left[\left(M_iM_j-1\right)\left(M_i-1\right)\left(M_j-1\right)\right] + 3\mathcal{H}\left(\frac{\eta}{2}\right),$$

otherwise,

$$\Delta I(\eta) = \log\left(M_i\right),$$

where $\mathcal{H}(x)$ denotes the binary entropy function, and $\mathcal{H}(x) = -(x\cdot\log x + (1-x)\cdot\log(1-x))$. Then, for the true mutual information I_{A_i,A_j} and empirical mutual information $I^n_{A_i,A_j}$ over n records, it holds that

$$\Pr\left(\left|I_{A_i,A_j} - I^n_{A_i,A_j}\right|\leq\Delta I(\eta)\right)\geq 1-\alpha. \tag{16}$$

For a given significance level $1-\alpha$, the relaxed threshold ℓ for I_{A_i,A_j} satisfies:

$$\Pr(I_{A_i,A_j}\geq\ell)\geq 1-\alpha. \tag{17}$$

Combining Equations 16 and 17, we can calculate ℓ as follows:

$$\ell = \tau_{A_i,A_j} - \Delta I\left(\left(\frac{2}{n}\ln\frac{2^{|\Omega_{A_i}||\Omega_{A_j}|}-2}{2\cdot\alpha}\right)^{\frac{1}{2}}\right). \tag{18}$$

If $I^n_{A_i,A_j}\geq\ell$, it means that A_i and A_j are strongly correlated with high probability, we keep edge (A_i,A_j) in the graph. Otherwise, we prune this edge.

Note that, according to Equation 18, ℓ will gradually increase with the increase of n, i.e., the relaxed threshold becomes closer to the actual threshold.

5.2.4 Details of Incremental Learning-based Method.

Based on the proposed data accumulation and edge selection techniques, we introduce the incremental learning-based dependency graph construction method. Algorithm 1 gives the details of this method.

In the beginning of the algorithm (Line 1), we initialize a dependency graph G with d vertices and let any two vertices have an edge connected. Let E be the edge set of G (i.e., the set of attribute pairs). As a next step, the algorithm partition the users $\widetilde{N}$ into T parts and each part $\widetilde{N}_t$ is used for the t-th iteration (Line 3). The rest of the algorithm consists of T iterations (Lines 4-14). In each iteration, we collect some new group of data for each remaining attribute pair in E and re-estimate the correlations to prune the attribute pairs with weak correlations. In particular, we judge whether to prune the edge $e=(i,j)\in E$ in t-th iteration following the steps below: (i) estimate $P_t(A_i,A_j)$ over $\widetilde{N}_{t,e}$ users (Lines 7-9), and then (ii) accumulate $P_t(A_i,A_j)$ to the result that obtained in the previous rounds (discussed in section 6.1.2) and recalculate the mutual information I_{A_i,A_j} (Lines 10-11), after that (iii) we calculate relaxed threshold ℓ (discussed in section 6.1.1) and decide whether to prune the edge e (Lines 13-14). Once the structure of the dependency graph G is decided, the algorithm terminates and returns the dependency graph G (Line 15).

Algorithm 1: Incremental Learning-based Method for Constructing Dependency Graph

```
Input: Dataset $\mathcal{D}$ with $d$ attributes, number of users $\widetilde{N}$, privacy
       budget $\epsilon$, dependency degree $\phi$, number of iterations $T$,
       significance level $\alpha$
Output: Dependency graph $G$
1  Initialize graph $G$ with $d$ nodes and the edge set
   $E = \{(i, j) \mid 0 \le i \ne j < d\}$;
2  Initialize $\mathcal{F}$ to store the computed 2-way marginals, and matrix $n$
   to record the number of users used in each edge;
3  Partition $\widetilde{N} = \sum_{t=1}^{T} \widetilde{N}_t$;
4  for the $t$-th iteration where $t \in [1, T]$ do
5      Partition $\widetilde{N}_t = \sum_{e \in E} \widetilde{N}_{t,e}$, where
       $\widetilde{N}_{t,e} = \frac{|\Omega_e|}{\sum_{(i,j)\in E} |\Omega_i| \times |\Omega_j|} \cdot \widetilde{N}_t$;
6      for each edge $e = (i, j)$ in $E$ do
7          $D_{t,e} \leftarrow$ Randomly select $\widetilde{N}_{t,e}$ users from Dataset $\mathcal{D}$;
8          $P_t(A_i, A_j) \leftarrow$ Estimate by VOUE$(D_{t,e}, e, \epsilon)$;
9          $n[i][j] += \widetilde{N}_{t,e}$;
10         $\mathcal{F}[e]$ = DataAccumulation$(\mathcal{F}[e], P_t(A_i, A_j))$;
11         Calculate mutual information $I_{A_i,A_j}$ by Equation 14
           with $\mathcal{F}[e]$;
12         Calculate threshold $\tau_{A_i,A_j}$ by Equation 15 with $\phi$;
13         Calculate relaxed threshold $\ell$ by Equation 18 with
           $\left(\tau_{A_i,A_j}, n[i][j], \alpha\right)$;
14         If $I_{A_i,A_j} < \ell$, remove edge $(i, j)$ from $E$;
15 return $G$;
```

5.3 Estimate the Joint Distribution of the Cliques

After constructing the dependency graph G by Algorithm 1, the aggregator transforms the dependency graph G into a junction tree $\mathcal{T}$ with clique set $C = \{C_1, \ldots, C_k\}$. To generate a synthetic dataset, we need to estimate the joint distribution of each clique in C. It is relatively straightforward to calculate the joint distribution of the small cliques, but it is highly nontrivial to calculate the joint distribution of the large cliques due to the curse of dimensionality. Recently, Cai et al. [8] propose PrivMRF, which calculates the joint distribution of the large cliques in the centralized setting. However, extending this method to the local setting will incur a high communication complexity. Thus, it urgently needs to design a new method for the local setting.

To solve this problem, based on joint distribution decomposition and redundancy elimination, we propose a novel marginal calculation method for the large cliques in the context of LDP. In particular, given a clique C contains $\hat{d}$ attributes $\left\{A_1, \ldots, A_{\hat{d}}\right\}$, we first factorize the full joint distribution into the product of multiple conditional probabilities, e.g.,

$$\begin{aligned}\Pr\left(A_1, \ldots, A_{\hat{d}}\right) &= \prod_{i=1}^{\hat{d}} \Pr\left(A_i | A_1, \ldots, A_{i-1}\right) \\ &= \prod_{i=1}^{\hat{d}} \Pr\left(A_i | \Pi_i\right),\end{aligned} \tag{19}$$

where Π_i denotes the set of conditions of A_i, i.e., $A_1, \ldots, A_{i-1}$. There may exist some redundancy in Π_i, i.e., given some attributes in Π_i, the other attributes in Π_i and A_i are conditionally independent. Then, we eliminate the redundancy in Π_i to get $\hat{\Pi}_i$. In practice, the size of $\hat{\Pi}_i$ is usually much smaller than that of Π_i, especially in the last few terms of the factorization, where

Algorithm 2: The Joint Distribution Estimation Method for Large Cliques

```
Input: A large clique $C$ with $\hat{d}$ attributes, number of users $\widetilde{N}$,
       the mutual information matrix $I$, privacy budget $\epsilon$, clique
       size threshold $\sigma$
Output: The joint distribution of the large clique $C$
1  Initialize $Q = \{A_1, \ldots, A_{\hat{d}}\}$, $\Psi$ to store the factorization results,
   and $\mathcal{M}$ be the set of factors needed to estimate the distribution;
2  for $i = \hat{d}$ to 1 do
3      if $|Q| < \sigma$ then
4          Random select an attribute from $Q$ as $A_h$, and let
           $S_{A_h} = Q \setminus A_h$;
5      else
6          for each attribute $A_j \in Q$ do
7              $S_{A_j} \leftarrow$ RedundancyEliminate$(A_j, Q \setminus A_j, I)$;
8              Var$(A_j) \leftarrow |S_{A_j} \cup A_j|$;
9          Let $A_h$ be the attribute in $Q$ that minimize Var$(A_h)$;
10     Insert $(A_h, S_{A_h})$ into $\Psi$, insert $S_{A_h} \cup A_h$ into $\mathcal{M}$;
11     Remove $A_h$ from $Q$;
12 Estimate the distribution of items in $\mathcal{M}$ with $\widetilde{N}$;
13 $P(C) = \prod_{(k_i, v_i) \in \Psi} P(k_i | v_i)$, where $P(k_i | v_i)$ is derived from
   $P(k_i, v_i)$;
14 return $P(C)$;
```

$\hat{\Pi}_i$ contains multiple attributes. Finally, calculate the joint distribution $\Pr(A_i, \hat{\Pi}_i)$ by collecting the perturbed data of remaining users and derive $\Pr(A_i | \hat{\Pi}_i)$ to achieve a more accurate distribution of C. The keys of this method are how to determine an optimal factorization order and make full use of the aggregations obtained in the graph construction phase to eliminate the redundancy without invoking additional users' data. However, finding the optimal factorization result is NP-hard, which requires us to choose the optimal solution from all possible factorization results, leading to substantial computational complexity. Thus, we design a heuristic algorithm to achieve a balance between accuracy and efficiency. In this algorithm, based on the observation that more redundancy will generally exist in the factorization items containing more attributes, we consider performing the forward selection strategy to achieve a better result (i.e., eliminate more redundancy). The details are shown in Algorithm 2.

In the beginning, we initialize the candidate attribute set Q to be $\left\{A_1, \ldots, A_{\hat{d}}\right\}$ and list Φ and $\mathcal{M}$ to be empty (Line 1). Φ and $\mathcal{M}$ are used to store the factorization results and the factors needed to estimate the distribution, respectively. Next, we employ a forward-selection strategy to find an approximate factorization result (i.e., we first determine the last item of the factorization $\Pr\left(x_{\hat{d}} | x_{\hat{d}-1}, \ldots, x_1\right)$, $x_{\hat{d}} \in Q$, and then determine the forward items in turn). More specifically, the strategy consists of $\hat{d}$ iterations (Lines 2-11), in each of which we divide into two cases according to $|Q| = \prod_{A_j \in Q} \left|\Omega_{A_j}\right|$ which is the domain size of Q:

(1) If $|Q| > \sigma$, for each attribute $A_j \in Q$, let A_j be the target attribute and use the minimal-redundancy-maximal-relevance feature selection method [27] to eliminate redundancy from $Q \setminus A_j$ (Line 7). The result of redundancy elimination is S_{A_j}. After that, we can obtain $|Q|$ results $\left\{S_{A_1}, \ldots, S_{A_{|Q|}}\right\}$. Let $A_h \in Q$ which brings the minimum variance $\left|S_{A_h} \cup A_h\right|$ be the selected attribute (Line 9).

Algorithm 3: PrivIncr

```
Input: Number of users $N$, privacy budget $\epsilon$, partition
       coefficient $\omega$
Output: A synthetic dataset $D^*$
1 Partition the users: $\widetilde{N}_1 = \omega \cdot N$, and $\widetilde{N}_2 = (1-\omega) \cdot N$;
2 $(G, \mathcal{F}, I) \leftarrow$ Construct the dependency graph, obtain any
    attribute pairs' distribution and mutual information by
    Algorithm 1 in Subsection 5.2 with $\widetilde{N}_1$ users;
3 Transform $G$ to junction tree and obtain clique set $\mathcal{C}$;
4 for each small clique $C \in \mathcal{C}$ do
5     $\Pr(C) \leftarrow$ Obtain the distribution from $\mathcal{F}$;
6 for each large clique $C \in \mathcal{C}$ do
7     $\Pr(C) \leftarrow$ Estimate by Algorithm 2 in Subsection 5.3 with
          $(\widetilde{N}_2, I)$;
8 Generating the synthetic datasets $D^*$ according to all cliques'
    distribution;
9 return $D^*$;
```

(2) Otherwise, we randomly select an attribute $A_h \in Q$ as the selected attribute and let $S_{A_h} = Q \setminus A_h$ (Line 4).

Then, add (A_h, S_{A_h}) into Ψ and $S_{A_h} \cup A_h$ into $\mathcal{M}$, and remove A_h from Q (Lines 10-11). Repeat the above steps until Q is empty, we obtain the factorization results Φ. Finally, we estimate the distribution of each item (i.e., factor) in $\mathcal{M}$ (Line 12) and then calculate the overall distribution of clique C by multiplying the factorization results in Φ (Line 13).

We can get an approximate factorization result with Algorithm 2 without bringing substantial computational complexity. Furthermore, we can also try to find the optimal factorization results by searching all possible factorization results when the clique has an acceptable dimension.

5.4 Putting Things Together: PrivIncr

Algorithm 3 illustrates the overall workflow of our solution, which can be referred to as PrivIncr. In the first phase (Line 1), we divide all the users into two groups $\widetilde{N}_1$ and $\widetilde{N}_2$ with the partition coefficient ω, where $\widetilde{N}_1$ is used to construct the dependency graph G and $\widetilde{N}_2$ is used to estimate the marginals of all cliques. The setting of ω will be discussed in Section 6.2.1. In the second phase, we construct the dependency graph G by algorithm 1 with $\widetilde{N}_1$ users (Line 2), and then transform the dependency graph G into a junction tree $\mathcal{T}$ composed of multiple cliques $\mathcal{C}$ (Line 3). In the third phase (Lines 4-7), we estimate the distribution $\Pr(C)$ of each clique $C \in \mathcal{C}$, respectively. Finally, according to the structure of the junction tree and all cliques' distribution, we can generate the synthetic dataset D^* (Line 8).

5.5 Privacy and Complexity Analysis

5.5.1 Privacy Guarantee.

In PrivIncr, each participant uploads their records only once under ϵ-local differential privacy protection. Thus, the whole solution satisfies ϵ-local differential privacy.

THEOREM 5.3. *PrivIncr satisfies ϵ-local differential privacy.*

5.5.2 Complexity Analysis.

We analyze the time complexity of our solution by discussing the time complexity of the key steps of our solutions as follows. We first suppose that (i) the average attribute domain size $|\Omega_i|$ as v, (ii) the average number of users used to re-estimate the distribute for each edge in each iteration as n_e, (iii) the average number of attributes in large cliques as $\hat{d}$, and (iv) the average number of users used to estimate the distribution of each item in the factorization result of the large clique as n_c.

Table 1: Dataset information description

Dataset	Original Tuples	Attributes	Domain Size	Sampled Tuples
Adult	45222	15	$\approx 2^{52}$	1.5×10^6
TPC-E	40000	24	$\approx 2^{77}$	2×10^6

THEOREM 5.4. *The worst-case time complexity of constructing the dependency graph (i.e., Algorithm 1) is $O\left(Td^2\left(n_e v^2 + v^2\right)\right)$.*

PROOF. According to Algorithm 1, the time overhead for reselect one edge in each iteration consists of four parts, (i) collect more data for estimating the distribution, (ii) accumulate with previous data, (iii) recalculate the mutual information, and (iv) calculate the relaxed threshold. The corresponding time overhead are $O\left(n_e v^2\right)$, $O\left(v^2\right)$, $O\left(v^2\right)$ and $O(1)$, respectively. There exist at most $\frac{d(d-1)}{2}$ edges and T iterations. Thus, in the worst case, the time complexity (i.e., no edges are pruned in each iteration) is $O\left(Td^2\left(n_e v^2 + v^2\right)\right)$.

In fact, in the process of incremental learning-based graph construction, many edges are pruned. Thus, the complexity will be significantly reduced.

THEOREM 5.5. *The worst-case time complexity of estimating the joint distribution of one large clique (i.e., Algorithm 2 in Subsection 5.3) is $O\left(\hat{d}^2 + n_c \cdot \sum_{i=1}^{\hat{d}} v^i + \hat{d}v^{\hat{d}}\right)$.*

PROOF. According to Algorithm 2, the time overhead for estimating the distribution of a large clique consists of three parts, (i) determine the factorization result, (ii) estimate the distributions of each item in factorization result, and (iii) calculate the distribution of the clique. The corresponding time overhead are $O\left(\hat{d}^2\right)$, $O\left(n_c \cdot \sum_{i=1}^{\hat{d}} v^i\right)$ and $O\left(\hat{d}v^{\hat{d}}\right)$, respectively. Thus, in the worst-case the time complexity is $O\left(\hat{d}^2 + n_c \cdot \sum_{i=1}^{\hat{d}} v^i + \hat{d}v^{\hat{d}}\right)$.

In practice, by employing redundancy elimination, the size of the factorization results can be significantly reduced. Thus, Algorithm 2 can always achieve high efficiency.

6 EXPERIMENTAL EVALUATION

6.1 Experimental Setup

Datasets. In our experiments, we use two real datasets: Adult [22], which contains census data from the 1994 US census of 1.5 million users and 15 attributes; TPC-E[*] contains information of "Trade", "Security", "Security status" and "Trade type" tables in the TPC-E benchmark, around two million records, and 23 attributes. Note that both Adult and TPC-E contain continuous and categorical attributes, so we discretized the continuous attributes to categorical attributes. For each continuous attribute, we discretize its domain into a fixed number b of equi-width ranges (we use $b = 16$). Meanwhile, to better evaluate the performance of methods, we increase the amount of Adult and TPC-E by sampling from their true distribution. Table 1 illustrates the properties of the datasets.

[*] TPC-E is available at http://www.tpc.org/.

Tasks and Metrics. Following [9, 43], we evaluate the performance of the published synthetic dataset on two types of tasks, the linear query task (e.g. k-way marginals) and the non-linear analysis task (e.g. SVM classification). For the first task, we evaluate the accuracy of k-way marginals of the synthetic dataset. Identical to [9, 33, 43], we use the total variant distance (referred to as TVD) to measure the accuracy between two distributions, which is as follows:

$$\|P - Q\|_{TVD} = \frac{1}{2} \sum_{\omega \in \Omega} |P(\omega) - Q(\omega)|, \tag{20}$$

where P denotes one k-way marginal over the synthetic dataset and Q denotes its real distribution. We report the average of all total variant distances as the final metric. The second task is to use the synthetic dataset to train the SVM classification models and use the classification rate to measure the performance. For each task, we repeat the experiment 100 times and report the average.

Competitors. We evaluate our PrivIncr by comparing it with the following methods:

- **LoPub** [33]: It is the existing work for multi-dimensional data publication with LDP, and its implementation details and limitations have been described in Section 4.
- **NoIncremental**: It is a baseline method. This approach constructs the probabilistic graph model without employing the incremental learning-based method. Specifically, NoIncremental directly selects edges from all $\binom{d}{2}$ edges at once to construct the dependency graph. Note that, NoIncremental plays two roles in the following experiments. On the one hand, it is the no-incremental version of PrivIncr. This helps us to understand the effectiveness of our incremental-based approach. On the other hand, it is an improved version of LoPub with two main modifications: (i) replacing the privacy budget split strategy with the data split strategy to improve the result utility. As described in [3], it has been proved that the latter strategy can often provide higher data utility than the former. (ii) changing the data perturbation method, i.e., replacing the combination of RAPPOR [16] and EM [17] algorithm with the OUE [37] method, to improve the accuracy and efficiency of the distribution estimation. By comparing NoIncremental (as the improved version of LoPub) with our solution PrivIncr, it helps us to demonstrate the superiority of our solution PrivIncr over the related work LoPub.
- **NoPrivJTree**: We directly construct the junction tree and calculate the distribution of cliques on the original dataset to generate a synthetic dataset without privacy protection, i.e., it does not enforce ϵ-local differential privacy. This can help us better understand how much inaccuracy is inherent in the underlying Junction tree model.
- **CALM** [44] and **FT** [10]: These are two approaches to construct k-way marginals for multi-dimensional attributes under LDP. CALM first partitions the attribute set with the optimal CoverDesign parameters and then reconstructs any k-way marginals from these marginals of the partitioned attribute set. FT is a Fourier-Transformation-based method. Note that FT is unable to deal with the non-binary attributes. Following [44], we encode each non-binary attribute into several binary attributes to implement the non-binary version of FT.

Parameters. For PrivIncr and NoIncremental, we set $\phi = 0.3$ to control the dependency level between pairwise attributes. Meanwhile, we allocate half of the dataset on the construction of PGM model and the other half on estimating the ditributions of all cliques in junction tree (i.e., the data partition coefficient ω in algorithm 3 is 0.5). In particular, we set the number of iterations $T = 6$ for PrivIncr in algorithm 1. These values are chosen based on our experimental results in Subsection 6.2.1.

Computing Environment Setup. All algorithms are implemented in Python 3.6, and all the experiments are run on a server with Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz and 251GB memory and Nvidia RTX 2080Ti GPU.

6.2 Experimental Results

6.2.1 Parameter Tuning.

In our first set of experiments, we evaluate the performance of PrivIncr on Adult while varying its two internal parameters: (i) the number of iterations T used in Algorithm 1; (ii) the data partition coefficient ω in Algorithm 3; (iii) the dependency degree ϕ in Algorithm 1.

Figure 4(a) shows the average total variation distance of the 3-way marginals generated from PrivIncr's output as T and ϵ varying. We observe that the average variation distance decreases with the increase of T. Meanwhile, when $T > 4$, the performance improves slowly. Therefore, we infer that an appropriate value for T should be in the range of $[4, 8]$, so we set $T = 6$ for PrivIncr to balance the performance in the following experiments.

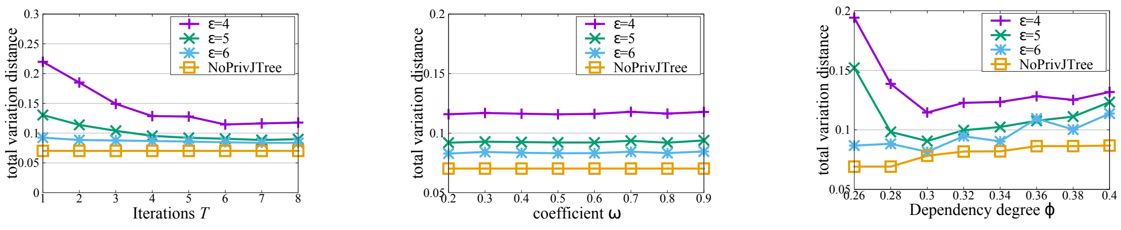
Figure 4(b) illustrates the performance of PrivIncr for 3-way marginal, when ω and ϵ varies. We can observe that changing the data partition coefficient ω almost does not affect the performance. This is because PrivIncr can already select the correct edges accurately under a small ω. Recall that the coefficient ω controls the number of users assigned to participate in the construction of the dependency graph and the estimation of the distribution of the cliques. On the one hand, when ω is small, a small number of users are used to construct the dependency graph, which compromises its ability to identify the correlations. On the other hand, when ω is large, although we can accurately model the dependency graph, in which case the estimation of the cliques' distributions will become error-prone and degrade the utility of the synthetic dataset. Based on these results, we set $\omega = 0.5$ as the default for all subsequent experiments.

Figure 4(c) plots the total variant distance of 3-way marginals when ϕ and ϵ vary. Observe that the overall performance of PrivIncr is optimized when ϕ is not too large or too small. This lies in that, in the construction of the dependency graph, a large ϕ means stricter correlation requirements, which makes some important correlation information lost. As the decrease of ϕ, these information losses are recaptured, and the performance is improved. However, when ϕ is further reduced, a lot of unnecessary information will be captured, and more noise needs to be injected to protect this information to meet privacy protection requirements. This results in performance degradation. Based on these observations, we set $\phi = 0.3$ for PrivIncr to achieve high performance.

6.2.2 Comparison with LoPub.

In the second set of experiments, we compare PrivIncr with LoPub on Adult by evaluating the accuracy of 2-way marginals of their synthetic datasets.

Table 2 illustrate the comparison results. We observe that the performance of LoPub is significantly worse than that of our



(a) Performance of PrivIncr vs. T (b) Performance of PrivIncr vs. ω (c) Performance of PrivIncr vs. ϕ

Figure 4: Parameter tuning on Adult

Table 2: The Comparison with LoPub

TVD		Privacy budget ε					
		1	4	8	30	165	442
Methods	LoPub	0.721	0.68	0.633	0.43	0.17	0.065
	NoIncremental	0.46	0.166	0.044	0.036	0.034	0.033
	PrivIncr	0.162	0.073	0.04	0.033	0.033	0.033

solution PrivIncr. Specifically, when the privacy budget is within a reasonable range, i.e., $\epsilon \in [1, 8]$, our method can achieve high performance while the error of LoPub is substantial. For LoPub, even with a privacy budget of 8, the error is still 0.633, which cannot meet the requirements of data utility for practical applications. To make LoPub achieve better performance, the privacy budget is set according to the privacy budget settings in [32] (i.e., $\epsilon = 30, 165, 442$, the corresponding perturbation probability f of each attribute is equal to 0.1, 0.5, and 0.9, respectively). At this time, the error of our solution PrivIncr can achieve 0.033, which is close to the error of the methods without considering privacy. While the error of LoPub can only achieve 0.065. Seriously, for LoPub, such large privacy budgets can hardly meet the privacy requirements of practical applications. For the poor performance of LoPub, besides the need to calculate marginal distributions of a large number of attribute pairs and large cliques will bring a large amount of noise (these challenges are also the focus of our work), there are also the following two reasons: (i) adopt the privacy budget split strategy. (ii) use the combination of RAPPOR [16] and EM [17] algorithm to perturb the data and estimate the distributions. These result in a large perturbation for each attribute (the details have been discussed in Section 4).

To better demonstrate the superiority of our solution, we design a new baseline method NoIncremental and compare it with our solution PrivIncr in the following experiments. NoIncremental can be seen as an improved version of LoPub with two main modifications: (i) replacing the privacy budget split strategy with the data split strategy. (ii) replacing the combination of RAPPOR and EM algorithm with the OUE [37] method. Please see more details in Section 6.1. Even so, our solution PrivIncr can consistently outperform NoIncremental.

6.2.3 *Results of k-way Marginals.*

In the third set of experiments, we compare PrivIncr with NoIncremental, NoPrivJTree, CALM and, FT on both two datasets.

Figure 5 shows the average total variation distance of each method for all k-way marginals under different privacy budgets ϵ. As can be observed, PrivIncr consistently outperforms other methods in almost all cases, and the difference becomes more apparent as the privacy budget decreases. Obverse that when $\epsilon = 4$, the average total variation distance of PrivIncr is half of that of NoIncremental (even one-third on TPC-E). Especially, when $\epsilon < 4$, NoIncremnetal cannot provide valid data accuracy. The reason is that the low privacy budget leads NoIncremnetal to calculate the pairwise correlations with a large amount of noise and select some wrong edges in the dependency graph. However, PrivIncr can prune most useless edges and allocate more data and privacy budget to the useful edges, improving the accuracy of the calculated correlations. With the increase of the privacy budget, the accuracy of PrivIncr is close to that of NoPrivJTree. The bad performance of FT is due to the binary encoding dramatically increasing the number of Fourier coefficients required to reconstruct marginals, and the results are consistent with the conclusion in [44].

6.2.4 *Results of SVM Classification.*

In the fourth set of experiments, we compare PrivIncr with NoIncremental, NoPrivJTree, and NoPrivacy for the SVM classification rate. NoPrivacy constructs classifiers directly on the input data without any privacy protection (let 80% of the records be the training set, and the remaining 20% are the testing set). We report the experimental results on Adult with three different attributes (two binary attributes and one Non-binary attribute) as labels: (i) salary, (ii) gender, and (iii) education. Meanwhile, on TPC-E, we construct three classifiers to predict (i) the trade type, (ii) the transaction type, and (iii) the security status type, respectively.

Figure 6 illustrates the SVM classification rate in each experiment. We can observe that PrivIncr consistently outperforms NoIncremental on both two datasets. Due to the small privacy budget will make it difficult for NoIncrement to identify the strong correlations, the performance of NoIncrement is much lower than PrivIncr. As the privacy budget grows, the accuracy of PrivIncr is close to that of NoPrivJTree. This is consistent with the results in Figure 5. Note that the accuracy of PrivIncr is better than that of NoPrivJTree when $\epsilon \geq 7$ in Figure 6(e), we interpret this phenomenon as reasonable because it is robust against noise injection when the number of tuples in the training set is large.

6.2.5 *Results of Computation Cost.*

In the last set of experiments, we evaluate the computation cost of PrivIncr, NoIncremental, and LoPub on both two datasets.

LoPub employs the EM algorithm to estimate the marginal distributions, and there exist some large cliques in the constructed dependency graph, which incurs a tremendous amount of computation cost. The running time of LoPub will be at least 20 hours. While PrivIncr and NoIncremental take no more than 30 minutes in all tested cases. The running time of LoPub is not in the same order of magnitude as the overhead of PrivIncr and NoIncremental. To more intuitively display the difference in the running time of PrivIncr and NoIncremental, we didn't show the results of LoPub in Figure 7.

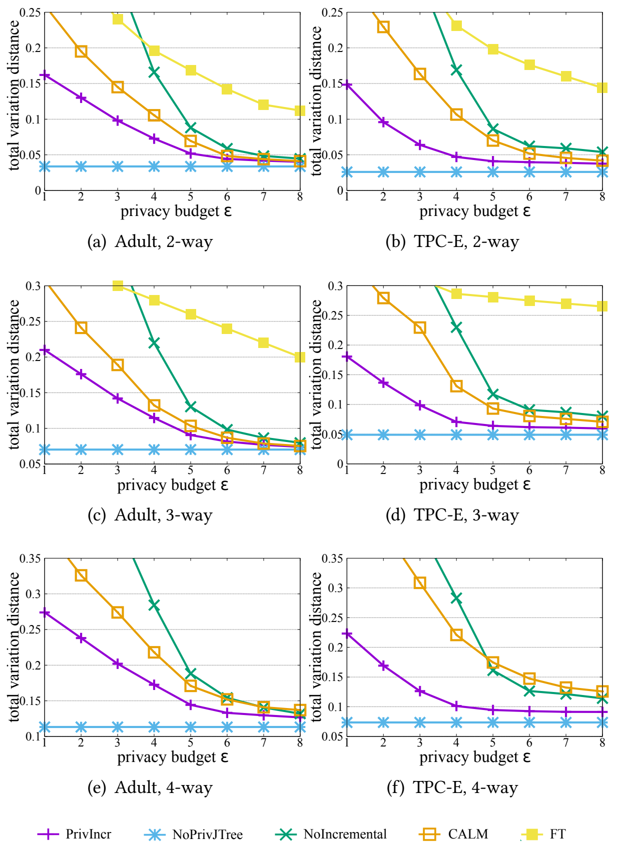


Figure 5: Average total variation distance of k-way marginals

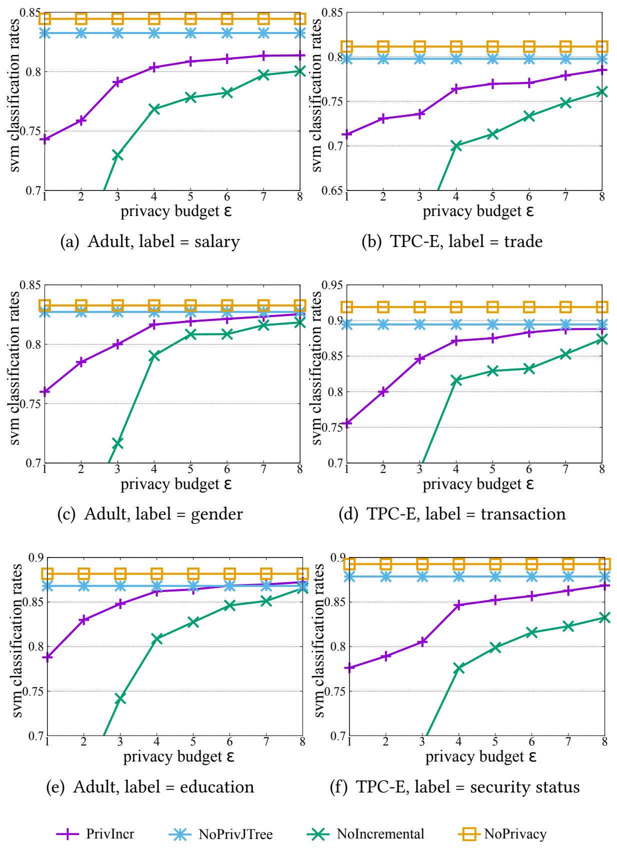


Figure 6: SVM classification rates

Figure 7(a) shows the running time of PrivIncr and NoIncremental in seconds under different numbers of iterations T, where the privacy budget $\epsilon = 4$ and the dependency degree $\phi = 0.3$. Because there is no parameter T in NoIncremental, its running time remains the same when T varies. We can observe that the running time of PrivIncr becomes smaller as the number of iterations increases and is always smaller than NoIncremental when the number of iterations is within a reasonable range. But excessive iterations will lead to increased runtime. The reason is as follows. In the incremental learning-based method PrivIncr, on the one hand, attribute pair (edge) pruning can reduce the computational complexity. On the other hand, data accumulation increases computational complexity. To reduce the computational complexity caused by the latter, we propose an efficient data accumulation method based on the divisibility of local differential privacy. Specifically, according to Equation 13 (please see more details in Section 5.2.3), we can calculate the results of the first T rounds based on the calculation results of the first $(T-1)$ rounds with a low computation cost. In this case, when the number of iterations is within a reasonable range, the benefits brought by pruning are more significant. However, when T is too large, the last few pruning rounds have little effect and only bring accumulation computation cost.

Figure 7(b) shows the running time of PrivIncr and Incremental in seconds under different parameters ϕ, where the privacy budget $\epsilon = 4$ and number of iterations $T = 6$. Generally speaking, the running time tends to decrease when the parameter ϕ increases in both of PrivIncr and NoIncremental. The reason lies

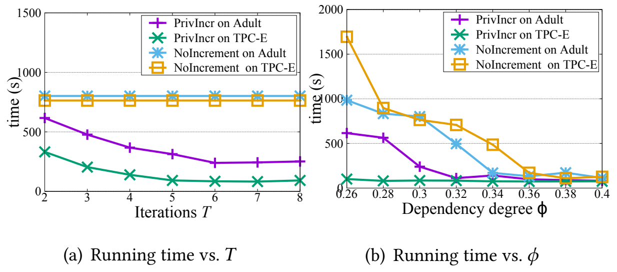


Figure 7: Computation Cost of PrivIncr and NoIncremental

in that, the parameter ϕ controls the threshold τ that is used to determine whether a pair of attributes have a strong correlation. According to Equation 15, the larger ϕ is, the larger τ is. Then, fewer edges are selected and contained in the constructed dependency graph. Such that, when calculating the marginal distribution of the clique, the amount of calculation is smaller. However, to achieve a high data utility of the synthetic dataset, we usually set $\phi = 0.3$.

7 CONCLUSION AND FUTURE WORK

In this paper, we studied the problem of multi-dimensional data publishing with local differential privacy (LDP). Based on the sparseness of PGM and the divisibility of LDP, we proposed an incremental learning-based method for constructing the junction tree. In addition, based on joint distribution decomposition and associated attribute selection, we proposed a marginal distribution calculation method for the large cliques in the PGM in the context of LDP. We presented a practical solution for multi-dimensional data publishing under LDP with the above techniques. Extensive experiments on real datasets demonstrated that our solution offers desirable data utility.

For future work, we plan to investigate the following two aspects. On the one hand, we will consider solving the problem of multiple dimensional data publishing with local differential privacy under the case where some tuples have missing values on some attributes. A feasible solution is to learn prior knowledge with the tuples that have no missing values to make inferences for the missing values, and then use these padded tuples to learn more accurate distribution information. On the other hand, our incremental learning-based method is a generic framework and of independent interest for other applications in the context of local differential privacy. We will consider using this method to solve the problems of extracting data features from a large number of candidates while guaranteeing local differential privacy.

ACKNOWLEDGMENTS

We thank the reviewers for their valuable comments which significantly improved this paper. The work was supported by National Natural Science Foundation of China under Grant No. 62002203, No. 61872105, No. 62072136, Shandong Provincial Natural Science Foundation No. ZR2020QF045, No. ZR2020MF055, No. ZR2021LZH007, No. ZR2020LZH002, the New Engineering Disciplines Research and Practice Project under Grant No. E-JSJRJ20201314, and Young Scholars Program of Shandong University.

REFERENCES

[1] Gergely Acs, Luca Melis, Claude Castelluccia, and Emiliano De Cristofaro. 2018. Differentially private mixture of generative neural networks. *TKDE* 31, 6 (2018), 1109–1121.

[2] Moustafa Alzantot and Mani Srivastava. 2019. Differentially Private Dataset Release using Wasserstein GANs. https://github.com/nesl/nist_differential_privacy_synthetic_data_challenge

[3] Héber H Arcolezi, Jean-François Couchot, Bechara Al Bouna, and Xiaokui Xiao. 2021. Random Sampling Plus Fake Data: Multidimensional Frequency Estimates With Local Differential Privacy. In *CIKM*. 47–57.

[4] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. 2007. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS*. 273–282.

[5] David Barber. 2004. Probabilistic modelling and reasoning: The junction tree algorithm. *Course notes* (2004).

[6] Raef Bassily and Adam Smith. 2015. Local, private, efficient protocols for succinct histograms. In *STOC*. 127–135.

[7] Brett K Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P Bhavnani, James Brian Byrd, and Casey S Greene. 2019. Privacy-preserving generative deep neural networks support clinical data sharing. *CIRC-CARDIOVASC QUAL* 12, 7 (2019), e005122.

[8] Kuntai Cai, Xiaoyu Lei, Jianxin Wei, and Xiaokui Xiao. 2021. Data Synthesis via Differentially Private Markov Random Fields. *VLDB* 14, 11 (2021), 2190–2202.

[9] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. 2015. Differentially private high-dimensional data publication via sampling-based inference. In *SIGKDD*. 129–138.

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

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

[12] 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 *CCS*. 1266–1288.

[13] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2018. Minimax optimal procedures for locally private estimation. *JASA* 113, 521 (2018), 182–201.

[14] Cynthia Dwork. 2006. Differential privacy. In *ICALP*. 1–12.

[15] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *TCC*. 265–284.

[16] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*. 1054–1067.

[17] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. 2015. Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries. *arXiv:1503.01214* (2015).

[18] Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. 2019. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *IFIP SEC*. 151–164.

[19] Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. 2014. Dual query: Practical private query release for high dimensional data. In *ICML*. 1170–1178.

[20] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2010. A simple and practical algorithm for differentially private data release. *arXiv:1012.4763* (2010).

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

[22] K.Bache and M.Lichman. 2013. UCI machine learning repository. (2013).

[23] Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques.*

[24] Haoran Li, Li Xiong, and Xiaoqian Jiang. 2014. Differentially private synthesization of multi-dimensional data using copula functions. In *EDBT*, Vol. 2014. 475.

[25] Lichun Li, Rongxing Lu, Kim-Kwang Raymond Choo, Anwitaman Datta, and Jun Shao. 2016. Privacy-preserving-outsourced association rule mining on vertically partitioned databases. *TIFS* 11, 8 (2016), 1847–1861.

[26] Zitao Li, Tianhao Wang, Milan Lopuhaä-Zwakenberg, Ninghui Li, and Boris Škoric. 2020. Estimating numerical distributions under local differential privacy. In *SIGMOD*. 621–635.

[27] Hanchuan Peng and et al. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *TPAMI* 27, 8 (2005), 1226–1238.

[28] Hanchuan Peng, Fuhui Long, and Chris Ding. 2019. Graphical-model based estimation and inference for differential privacy. In *ICML*. 4435–4444.

[29] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2014. Priview: practical differentially private release of marginal contingency tables. In *SIGMOD*. 1435–1446.

[30] Zhan Qin and et al. 2016. Heavy hitter estimation over set-valued data with local differential privacy. In *CCS*. 192–203.

[31] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2021. Frequency Estimation under Local Differential Privacy [Experiments, Analysis and Benchmarks]. *arXiv:2103.16640* (2021).

[32] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, and Julie McCann. 2016. High-dimensional crowdsourced data distribution estimation with local privacy. In *CIT*. IEEE, 226–233.

[33] 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. *TIFS* 13, 9 (2018), 2151–2166.

[34] Arno G Stefani, Johannes B Huber, Christophe Jardin, and Heinrich Sticht. 2014. Confidence intervals for the mutual information. *IJMISSP* 1, 3 (2014), 201–214.

[35] Giuseppe Vietri, Grace Tian, Mark Bun, Thomas Steinke, and Steven Wu. 2020. New oracle-efficient algorithms for private synthetic data release. In *ICML*. 9765–9774.

[36] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and analyzing multidimensional data with local differential privacy. In *ICDE*. 638–649.

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

[38] 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 *SIGMOD*. 159–176.

[39] Tianhao Wang, Ninghui Li, and Somesh Jha. 2018. Locally differentially private frequent itemset mining. In *S&P*. 127–143.

[40] Tianhao Wang, Ninghui Li, and Somesh Jha. 2019. Locally differentially private heavy hitter identification. *TDSC* (2019).

[41] Teng Wang, Xinyu Yang, Xuebin Ren, Wei Yu, and Shusen Yang. 2019. Locally private high-dimensional crowdsourced data release based on copula functions. *TCS* (2019).

[42] Jianyu Yang, Tianhao Wang, Ninghui Li, Xiang Cheng, and Sen Su. 2020. Answering multi-dimensional range queries under local differential privacy. *arXiv:2009.06538* (2020).

[43] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. Privbayes: Private data release via bayesian networks. *TODS* 42, 4 (2017), 1–41.

[44] 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 *CCS*. 212–229.

[45] Zhikun Zhang, Tianhao Wang, Ninghui Li, Jean Honorio, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. 2021. Privsyn: Differentially private data synthesis. In *USENIX Security*.