DPView: Differentially Private Data Synthesis Through Domain Size Information

Chih-Hsun Lin, Chia-Mu Yu, *Senior Member, IEEE,* and Chun-Ying Huang, *Member, IEEE*

Abstract—**The use of differentially private synthetic data has been adopted as a common security measure for the public release of sensitive data. However, the existing solutions either suffer from serious privacy budget splitting or fail to fully automate the generation procedures. In this study, we propose an automated system for synthesizing differentially private synthetic tabular data, called DPView. Our key insight is that high-dimensional data synthesis can be accomplished by utilizing the domain sizes of attributes, which are public information, whereas identifying the correlation among attributes is necessary but leads to severe privacy budget splitting. In addition, we analytically optimize both the privacy budget allocation and consistency procedures of the proposed method through mathematical programming. We further propose two novel methods, including iterative non-negativity and consistency-aware normalization, to post-process the synthetic data. An extensive set of experimental results demonstrates the superior utility of DPView.**

Index Terms—**Differential Privacy, Synthetic Dataset.**

I. Introduction

Differential privacy (DP) [17], [22], [38] has been accepted as the *de facto* standard for data privacy. Companies and government agencies commonly conduct privacy-preserving data analysis using DP. For example, Google publishes an early release of their COVID-19 Community Mobility Reports with DP. Uber and LinkedIn apply DP to their in-house data warehouse, while startups such as LeapYear and Privitar offer DP approaches for regulatory compliance, de-identification, and anti-money laundering compliance. The US Census Bureau has also released the 2020 census statistics with DP.

Most of the previous work on DP has focused on the design of algorithms for specific data analysis tasks. However, this narrow view may involve three limitations, which are labeled as follows. **(L1)** The data owner should be assumed to be aware of the specific data analysis task to be performed by an analyst. Otherwise, the resulting DP dataset may involve a potential privacy violation. **(L2)** The design and deployment of DP algorithms requires expert knowledge of the specific task to be performed. For example, the SQL engine requires patching to perform DP-SQL queries, while stochastic gradient descent (SGD) methods must be modified to train a deep neural network in a DP manner. **(L3)** Thus, a data analyst might be required to change their operating procedures to adopt DP. While data analysts commonly train to apply conventional analysis tools, this additional complexity may present a challenge to DP adoption.

Chih-Hsun Lin and Chun-Ying Huang are with Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan. Chia-Mu Yu (corresponding author, chiamuyu@nctu.edu.tw) is with Department of Information Management and Finance, National Yang Ming Chiao Tung University. Chih-Hsun Lin and Chia-Mu Yu were supported by MOST 110-2636-E-009-018, and we also thank National Center for High-performance Computing (NCHC) of National Applied Research Laboratories (NARLabs) in Taiwan for providing computational and storage resources. Chun-Ying Huang was supported by MOST 110-2628-E-A49-011 and MOST 110-2218-E-A49-011-MBK.

Differentially Private Synthetic Dataset. One promising solution to address the above limitations is to generate a differentially private synthetic dataset (DPSD). As the DPSD is statistically similar to the original dataset, all analytical conclusions derived from DPSD will be similar to those from the original dataset, overcoming **(L1)**. As the analyst's operations can be understood as post-processing and therefore do not violate the privacy of the data, analysts can apply their conventional data analysis algorithms to DPSDs, overcoming **(L2)** and **(L3)**.

The current solutions for DPSD generation can largely be categorized into two types, including parametric and non-parametric [9]. The former (e.g., PrivBayes [59], and PATE-GAN [36]) use Bayesian networks, generative adversarial networks (GAN), or other generative models to learn a data generating distribution and then create a synthetic dataset from the distribution through the sampling. The latter (e.g., DPSyn [41] and PrivSyn [62]) work mainly on the empirical distribution by decomposing a high-dimensional dataset into a number of noisy marginals and generating the synthetic dataset from these noisy marginals. Parametric approaches are difficult to train and often do not have a clear privacy-utility trade-off [9]. In contrast, non-parametric approaches tend to be more efficient and their utility increases with increasing privacy budget.

Design Challenge. However, most of the current DPSD generation algorithms face the following three challenges.

- **(C1)** As all of the non-parametric approaches approximate a high-dimensional joint data distribution through a number of low-dimensional marginals, a data-dependent algorithm is required to choose the highly correlated attributes. Due to the high number of dimensions in the underlying dataset, the above data-dependency incurs severe privacy budget splitting, with a negative effect on utility.
- **(C2)** Although metrics such as mutual information are popular for measuring the correlation between attributes, they have high sensitivity, magnifying the scale of noise in the data. Moreover, it is difficult to determine a clear threshold separating high and low correlation. Many independent attributes might be included in the case of a low threshold, leading to a more severe data sparsity. In the case of a high threshold, correlation information from the data distribution might be lost. Both problems greatly diminish the utility of schemes that employ them.
- **(C3)** Lastly, most of the current DPSD generation algorithms require extensive manual effort [9] and cannot automatically generate a DPSD. For example, despite winning an award in an NIST Challenge, PGM [44] required the manual construction of graph networks. Similarly, the data utility of DP-GAN [60], [3], [11] has been shown to be sensitive to the settings of parameters and hyper-parameters, which are left to be determined by the user.

Contributions. We propose DPView as a non-parametric DPSD generation algorithm. DPView achieves pure DP, as opposed to the approximate DP of most prior works. Our key contributions can be summarized as follows. First, we propose a novel domain size-aware marginal selection method that does *not* consume the privacy budget. In contrast to prior view-based approaches that split the privacy budget to select proper marginals, our proposed method performs *data-independent* marginal selection by taking advantage of domain sizes of attributes' public information. Compared with the existing work, our unique feature of data independence significantly alleviates the budget splitting problem, overcoming **(C1)** and **(C2)**. Moreover, the configuration of system parameters can also be guided by the domain size information, enabling automated DPSD generation and overcoming **(C3)**.

Second, as the marginal with a larger domain size may lead to a higher sparsity resulting in lower noise tolerance, room for improvement remains in terms of utility if the privacy budget is evenly allocated to all marginals. Here, we formulate the privacy budget allocation as an optimization problem. We then analytically derive the optimum privacy budget allocation among marginals. Third, we propose three novel post-processing techniques to improve utility. In particular, we formulate a consistency constraint as an optimization to achieve the optimum weight distribution. Our proposed *iterative non-negativity*, compared to the prior non-negativity which is not guaranteed to stop with an indeterminate threshold, has greater efficiency and retains the noise scale better. Our proposed *consistency-aware normalization* also applies and propagates a normalization from a carefully selected view to all the other views without destroying the consistency of the data.

II. Related Work

Differential privacy (DP) [17], [22], [38] has widely been considered a gold standard in data privacy. Many DP algorithms have been proposed for specific tasks such as mining frequent graph patterns [47], mining frequent itemsets [39], and training neural networks [2]. However, a number of researchers have sought to generate general-purpose differentially private synthetic datasets (DPSD); i.e., although the optimal utility would not be expected for a given specific task, it should be possible to perform any algorithm on a DPSD and still obtains an analytical result with a reasonable utility. Many DPSD generation algorithms may be found in the literature. These can be divided into two categories, including parametric and non-parametric. Some theoretical treatments have also been devoted to DPSD generation. These are summarized below.

Parametric Approaches. Parametric approaches are designed to learn a data distribution from the original dataset in a DP manner, and then perform the sampling from the distribution to synthesize the data. PrivBayes [59] and BSG [10] approximated the data distribution using a Bayesian network. They suffered from the high sensitivity of their correlation functions, which reduces the accuracy of their network structure. KAMINO [29] follows a similar strategy by decomposing the joint probability of the original dataset into a chain of conditional probabilities, and estimates a privacy-preserving data distribution using tuple embedding [54] and an attention mechanism [6].

In contrast, JTree [16] and PGM [44] approximated the data distribution using a Markov Random Field (MRF). JTree was proposed to construct a dependency graph using the sparse vector technique (SVT); however, Lyu et al. [40] found that JTree applied SVT in a problematic way. From a set of manually chosen low-dimensional marginals, PGM estimated an MRF that best fit the marginals.

As a result of recent advance of deep learning, deep generative models have also been used in DPSD generation. Driven by the success of generative adversarial networks (GAN), the majority of such efforts [60], [3], [11], [24], [52], [49], [55], [5] have been devoted to the development of DP-GANs. Using a training phase that may access the original dataset, the DP-SGD strategy [2] is a popular choice to obfuscate gradients in DP-GANs. On the basis of DP-SGD, Zhang et al. [60] proposed clustering the weights of neural networks and warm starting to reduce the noise scale. The PATE-GAN [36] framework performed private aggregation of teacher ensembles (PATE) to tightly bound the influence of any individual sample on the model. Despite the popularity of DP-GANs, variational autoencoders (VAEs), used by P3GM [50] and DP-VAE [15], have also been alternatively used for DPSD generation.

Non-Parametric Approaches. Non-parametric approaches differ from parametric methods in that the former functions mainly on the empirical distribution of the original dataset. However, they still share the aim of reducing the noise scale by approximating a high-dimensional distribution using a number of low-dimensional distributions. To further mitigate noise accumulation, non-parametric approaches can further reduce the number of cell counts using methods such as maintaining only highly correlated marginals and applying the privacy budget asymmetrically across cells.

The publication of differentially private histograms was an early non-parametric approach [58], [57]. However, it only applied to low-dimensional datasets. For high-dimensional datasets, PriView [46] partitioned a full-dimensional marginal into a number of low-dimensional marginals by leveraging covering design, reducing the data sparsity. While PriView performed better for binary data, DPSyn [41] extended a similar idea, enabling the corresponding DPSD generation of high-utility non-binary data. Very recently, PrivSyn [62] proposed a surrogate function with much lower sensitivity for measuring dependency, resulting in a much lower noise scale and thereby improving the overall utility.

Theoretical Results. Some DPSD algorithms have been proposed that do not fall into either category. Based on a combination of the multiplicative weights update rule [33], [28] with an exponential mechanism [45], MWEM [32] initialized and refined a random dataset, ensuring that its statistical properties were consistent with those of the original dataset. Based on the multiplicative weights update method as an algorithmic technique used in game theory, Dual Query [27] modeled data synthesis as a query release game. Thaler et al. [51] sanitized and released a dataset by leveraging Chebyshev polynomials.

Negative results on the DPSD have been reported [7] [21]. However, their query sets were much broader than the natural set of queries in which a database user or maintainer would be interested. Moreover, their results were all derived in an asymptotic sense. Hence, these negative results do not rule out DPSD algorithms in practice.

III. Preliminaries

A. Problem Statement

We consider the problem as follows. A data owner has an original dataset O with sensitive information. Instead of releasing O, the data owner generates a corresponding DPSD

$\hat{O}$. $\hat{O}$ and O share similar statistical information; however, the release of $\hat{O}$ will not leak the individual information of O due to DP. Here, we assume that both O and $\hat{O}$ have n independent records, each of which has m attributes. Each cell of O and $\hat{O}$ can be either numeric or categorical. We also use *partial record* to refer to a k-dimensional record with $k < m$ from a dataset. For example, both $[3,5]$ and $[5,7,9]$ are partial records if the record $[3,5,7,9]$ exists in a dataset with $m = 4$.

More specifically, consider a dataset O with attributes $\mathbb{A} = \{a_1, a_2, \ldots, a_m\}$. $[m]$ denotes an interval of integers from 1 to m. The function $dom(a_i) = s_i$ calculates that the domain size of a_i is s_i. In other words, $dom(a_i) = s_i$ means that attribute a_i has s_i value possibilities after bucketization (See Section 4).

Contingency tables can be seen as histograms, in which each *cell* counts the number of occurrences for each tuple. Depending on context, *k-way marginal* either represents a probability distribution of k attributes or serves the same role as contingency table. *View* v is a (sub)set of attributes, and *view size*, $|v|$, is the corresponding number of attributes. When there are multiple views v_i, we use v_i-marginal to refer to the marginal with attributes in v_i. The domain size of a view/contingency table/marginal refers to the total number of value possibilities after bucketization. For example, the domain size of a view v consisting of a_1 and a_3, is $dom(v) = dom(a_1) \times dom(a_3) = s_1 \times s_3$. For notational brevity, we use $T_{v_i}(c)$ to retrieve the count of the cell c from the v_i-marginal. Moreover, we can also calculate the count $T_v((\beta_1, \beta_3)) = \sum_{i \in a_2} T_v((\beta_1, i, \beta_3))$ for the cell (β_1, β_3) of the v-marginal with $v = \{a_1, a_2, a_3\}$, where the term $i \in a_2$ indicates that i is a candidate value of the attribute a_2.

Privacy and utility are frequently used general metrics for evaluating the quality of DPSD. In particular, for privacy, we use ε (see Section III-B) as the standard. In contrast, as the analytical results on O are supposed to be identical to those on $\hat{O}$ in the ideal case, utility measures the similarity between $f(O)$ and $f(\hat{O})$ for all functions f. However, in practice, we cannot enumerate all f's; in this study, we use the classification, k-way marginals, and range query as representatives for f (see Section V-A2).

B. Differential Privacy

Differential privacy (DP) [17], [22], [38] is defined as mathematical privacy notion for statistical datasets. DP can be achieved by injecting noise to the query result. We provide the formal definition below.

Definition 1 (Differential Privacy): A randomized mechanism $\mathcal{M}$ satisfies (ε, δ)-DP, $\varepsilon > 0$, $\delta \geq 0$, if and only if

$$\forall \alpha \subseteq \mathrm{Range(M)}, \Pr[\mathcal{M}(\mathrm{D}) = \alpha] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(\mathrm{D}') = \alpha] + \delta, \tag{1}$$

where D and D' are two neighboring datasets.
In Definition 1, two datasets D and D' are neighboring if they differ in at most one record. Lower *privacy budget* ε implies higher privacy, while δ corresponds to the probability of failing to fulfill the DP definition. $(\varepsilon, 0)$-DP (abbreviated as ε-DP) is also called *pure DP* and (ε, δ)-DP is *approximate DP*. In this study, we consider pure DP only, unless stated otherwise. Given a $f(D)$ and a zero-mean Laplace distribution $\Pr[\mathrm{Lap}(\beta) = \mathrm{x}] = \frac{1}{2\beta} e^{-|\mathrm{x}|/\beta}$, one can derive its ε-DP version $\mathcal{M}_\varepsilon(D)$ through the Laplace mechanism, defined as follows.

Definition 2 (Laplace Mechanism): $\mathcal{M}_\varepsilon(D) = f(D) + \mathrm{Lap}(\frac{\Delta_f}{\varepsilon})$ achieves ε-DP, where $\Delta_f = max_{D,D'} |f(D) - f(D')|$.

DP also satisfies sequential composition and post-processing conditions. The former states that greater accesses to the data leads to greater privacy loss, while the latter states that any data-independent operations on the DP data do not harm the DP guarantee.

Definition 3 (Sequential Composition): Given $M_{\varepsilon_1}(D)$ satisfying ε_1-DP and $M'_{\varepsilon_2}(D)$ satisfying ε_2-DP, $M'_{\varepsilon_2}(M_{\varepsilon_1}(D))$ satisfies $(\varepsilon_1 + \varepsilon_2)$-DP.

Definition 4 (Post-Processing): Given $M_\varepsilon(D)$ satisfying ε-DP, for any data-independent algorithm M', the composition of M and M', i.e., $M'(M(D))$, satisfies ε-DP.

In DP algorithm design, the *privacy budget splitting* problem should be avoided. It may be observed from Definition 3 that given ε, compared to an algorithm "touching" the data only once, if the algorithm makes k accesses to O, each access only has a privacy budget ε/k and will be perturbed by a heavier noise $Lap(\frac{\Delta_f}{\varepsilon/k})$.

Three relaxed versions of DP, known as Concentrated DP (CDP) [23], zero CDP (zCDP) [8], Rényi DP (RDP) [42], have been proposed to achieve a tighter analysis of cumulative privacy loss. However, recently Jayaraman and Evans [35] found that relaxed definitions of DP reducing the amount of noise needed to improve utility also increased the measured privacy leakage. As a result, in this study, we focus solely on pure DP.

IV. DPView

After introducing the general framework of DPSD generation, this section describes DPView, followed by its variant DPView+.

A. A General Framework

A general framework is commonly used in non-parametric DPSD generation, consisting of five steps, including data preprocessing **(S1)**, view list generation **(S2)**, noisy marginal construction **(S3)**, post-processing **(S4)**, and data synthesis **(S5)**. More specifically, the objective of **(S1)** is to modify, transform, and clean the input dataset, ensuring that it is in a correct format for further operations. **(S2)** decomposes the high-dimensional dataset as a number of low-dimensional datasets, usually according to its attributes. Thus, this step can be considered as generating a list of proper views. After **(S3)** injects noise into the data, **(S4)** attempts to reduce the impact of noise by leveraging the inherent constraints of the dataset. Lastly, **(S5)** creates the DPSD, mainly by using a sampling-based technique.

DPView, the workflow of which is shown in Figure 1, follows the above framework and therefore also consists of five steps, which are detailed below.

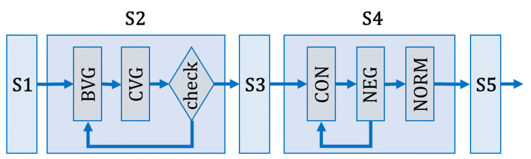


Fig. 1: Workflow of DPView. BVG and CVG denote base view generation and cross view generation, respectively, which are described in Section IV-C. CON, NEG, NORM denote consistency, non-negativity, and normalization, respectively, which are described in Section IV-E.

*B. Data Pre-Processing **(S1)***

DPView can handle only categorical data. Thus, this step performs bucketization, discretizing the numeric data. In particular, a numeric attribute is be partitioned into pre-defined buckets representing a numeric interval, and a bucket id replaces each numeric value. Thus, we can transform a numeric attribute into a categorical attribute. In this study, we use equal-size buckets, and therefore the number of buckers is the only parameter to be determined.

*C. View List Generation **(S2)***

In contrast to the existing DPSD algorithms which find highly-correlated views, DPView aims to construct views less impacted by noise by using the public information of domain size. In particular, DPView includes base and cross view types, which are formally defined below.

Definition 5: Given the view size d, the set B of base views in DPView is defined as $B = \{b_1, \ldots, b_{\lceil \frac{m}{d} \rceil} | b_i \cap b_j = \phi, i, j = 1, 2, \ldots, \lceil \frac{m}{d} \rceil, i \neq j, |b_1| = \cdots = |b_{\lceil \frac{m}{d} \rceil - 1}| = d, |b_{\lceil \frac{m}{d} \rceil}| \leq d\}$.

Definition 6: Given d and B, the set C of cross views is defined as $C = \{c_1, \ldots, c_{\lceil \frac{m}{d} \rceil - 1} | c_r \subseteq b_r \cup b_{r+1}, c_r \cap b_r \neq \emptyset, c_r \cap b_{r+1} \neq \emptyset, r = 1, \ldots, (\lceil \frac{m}{d} \rceil - 1), |c_1| = \cdots = |c_{\lceil \frac{m}{d} \rceil - 2}| = d, |c_{\lceil \frac{m}{d} \rceil - 1}| \leq d\}$.

1) Base View: Base views aim to cover all the attributes in O with the minimum number of views for a given view size d. From Definition 5, it may be observed that more base views imply a lower memory overhead and less sparsity for marginals due to the decreased domain size of each view. However, more base views indicate less correlation information among attributes preserved and more severe privacy budget splitting. To attain our objective, we need to balance the above two contradicting requirements; thus, we provide a guideline to assist in choosing a proper d in Section IV-C7, which directly leads to an adequate number of base views.

Given the number m of attributes in O and a fixed view size d, there are two cases for base view construction. In the case of $d|m$, we can generate m/d base views, each having view size d. In contrast, in the case of $d \nmid m$, the last base view $b_{\lceil \frac{m}{d} \rceil}$ will have a view size less than d. This affects how we determine the cross views, as described below.

2) Cross View: Cross views are used to bridge base views to compensate for the correlation information missing among base views. Given m, d, and B, there are three cases for cross views.

• **Type-I** ($d|m$) For every pair of base views b_i and b_{i+1}, we generate a cross view according to Definition 6. For example, in the Type-I of Figure 2, given base views $b_1 = \{a_1, a_2, a_3\}$, $b_2 = \{a_4, a_5, a_6\}$ and $b_3 = \{a_7, a_8, a_9\}$, $c_1 = \{a_2, a_3, a_4\}$ and $c_2 = \{a_5, a_7, a_8\}$ may be possible cross views.

• **Type-II** ($d \nmid m$ and $|b_{\lceil \frac{m}{d} \rceil}| > d/2$) We still generate a cross view directly according to Definition 6. For example, in the Type-II of Figure 2, given base views $b_1 = \{a_1, a_2, a_3\}$, $b_2 = \{a_4, a_5, a_6\}$ and $b_3 = \{a_7, a_8\}$, $c_1 = \{a_2, a_3, a_4\}$ and $c_2 = \{a_5, a_6, a_7\}$ may be possible cross views.

• **Type-III** ($d \nmid m$ and $|b_{\lceil \frac{m}{d} \rceil}| \leq d/2$) In this case, we select $d - |b_{\lceil \frac{m}{d} \rceil}|$ random attributes from $b_{\lceil \frac{m}{d} \rceil - 1}$ and include them in $b_{\lceil \frac{m}{d} \rceil}$ first. However, we do not generate a cross view for $b_{\lceil \frac{m}{d} \rceil - 1}$ and $b_{\lceil \frac{m}{d} \rceil}$ because the above compensation behavior for $b_{\lceil \frac{m}{d} \rceil}$ already bridges $b_{\lceil \frac{m}{d} \rceil - 1}$ and $b_{\lceil \frac{m}{d} \rceil}$. Thus, to generate an additional cross view would be superfluous and aggravate privacy budget splitting.

Two potential questions may arise during Type-III cross view construction. The first concerns the reason for only selecting attributes from $b_{\lceil \frac{m}{d} \rceil - 1}$, instead of from all the previous base views $b_1, b_2, \ldots, b_{\lceil \frac{m}{d} \rceil - 1}$, to compensate $b_{\lceil \frac{m}{d} \rceil}$. This can be attributed to the fact that, in this case, the attributes with much smaller domain size compared to all the previous attributes are highly likely to be selected due to our algorithm design (Algorithm 3, see Section IV-C). At first glance, $b_{\lceil \frac{m}{d} \rceil}$ will have a smaller domain size, which seems beneficial in terms of sparsity. However, it also leads to the loss of correlation information between $b_{\lceil \frac{m}{d} \rceil - 1}$ and $b_{\lceil \frac{m}{d} \rceil}$. Instead, after the compensation $b_{\lceil \frac{m}{d} \rceil}$ may have attributes from each previous base view, which may reduce the utility of the DPSD.

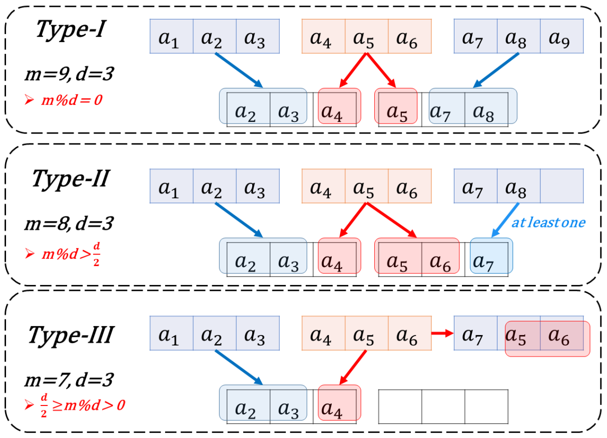


Fig. 2: Three types of cross views.

The second question concerns the reason for the necessity of compensating $b_{\lceil \frac{m}{d} \rceil}$, rather than interpreting the original $b_{\lceil \frac{m}{d} \rceil}$ as an individual base view and constructing a cross view between $b_{\lceil \frac{m}{d} \rceil - 1}$ and $b_{\lceil \frac{m}{d} \rceil}$. Without the compensation, $|B|+|C|$ views are required in total, whereas only $|B| + |C| - 1$ views are required when applying the compensation. This slightly alleviates the privacy budget splitting. Moreover, since there are only few attributes in the original $b_{\lceil \frac{m}{d} \rceil}$, if no compensation applies to $b_{\lceil \frac{m}{d} \rceil}$ and the attributes in $b_{\lceil \frac{m}{d} \rceil}$ happen to have large domain size, the attributes with larger domain size may be included in the cross view, worsening the data sparsity.

3) Formulation of Base View and Cross View Construction: Based on our observation that views with proper domain sizes may be expected to balance memory overhead, sparsity, and data synthesis accuracy, DPView aims to find base and cross views allowing the summation of domain sizes of both views to be minimized. More formally, we formulate the problem of constructing base views and cross views as an optimization as follows.

Definition 7: Given a set S of n numbers (domain sizes) and a positive integer $\lceil m/d \rceil$, the objective of the base view-cross view construction (BVCVC) problem is to find the optimum base and cross views satisfying Definitions 5 and 6 such that the summation of domain sizes of both views is minimized.

In particular, we refer to a balanced multi-way number partitioning (BMNP) problem, whose input is a set S of n numbers and a positive integer k. The objective of BMNP is to partition S into k subsets such that each subset is either $\lceil n/k \rceil$ or $\lfloor n/k \rfloor$ numbers and the difference between the maximum and minimum subset sum is minimized. Due to the similarity between BMNP and BVCVC, BVCVC can be easily proved to be NP-Hard (formal proof omitted) because BMNP has been proved NP-Hard [20].

Algorithm 1: Base View Generation (BVG)

```
1 Input: $d$, $\mathbb{A} = \{a_1, \ldots, a_m\}$
2 randomly partition $\{a_1, \ldots, a_m\}$ into a set $B$ of $\lceil \frac{m}{d} \rceil$
    base views according to Definition 5
3 sort and number views in $B$ by top-$\lfloor \frac{d}{2} \rfloor$ largest domain
    sizes in a descending order
4 return $B$
```

Using approximation or heuristic algorithms for BMNP [43], [61] to find base views first could serve as an approach to a solution of BVCVC. However, after determining base views, we still have no information to determine cross views. Worse, as BVCVC requires a joint consideration of base and cross views, the deterministic nature of its approximation and heuristic algorithms may hinder the search for base and cross views with smaller domain sizes.

4) Algorithm for Base View Generation (BVG): Despite the definition of base views, we need an algorithm for explicitly constructing base views. Here, we first present BVG (Algorithm 1) as a subroutine of VLG (Algorithm 3; see Section IV-C6). BVG can be understood largely as a simple algorithm generating random base views. However, it is featured by its ordering of base views. In particular, originally, the base views generated lack explicit order. Therefore, once base views are generated, BVG sorts and numbers them according to their respective top-$\lfloor \frac{d}{2} \rfloor$ largest domain size in descending order, where the *top-$\lfloor \frac{d}{2} \rfloor$ largest domain size of b_i* is defined as $s_{j_1} \cdots s_{j_{\lfloor \frac{d}{2} \rfloor}}$ with $a_{j_1}, \ldots, a_{j_{\lfloor \frac{d}{2} \rfloor}}$ denoting the attributes having the largest domain sizes in b_i, $j_1, \ldots, j_{\lfloor \frac{d}{2} \rfloor} \in [m]$.

For a list $[\beta_1, \ldots, \beta_{2k}]$ of the sorted numbers, the summation of products of the elements in the list $[[\beta_1, \beta_{2k}], \ldots, [\beta_k, \beta_{k+1}]]$ of 2-partitions, $\sum_{i=1}^{k} \beta_i \beta_{2k-i+1}$, can be minimized. Unfortunately, the scenario of base view construction is more complex, e.g., view size is not limited to two. Thus, the algorithm sorts and numbers the base views according to their respective top-$\lfloor \frac{d}{2} \rfloor$ largest domain sizes in a descending order in an attempt to perform a similar minimization. For example, the upper part of Figure 3 includes four base views. After the sorting and numbering, as the largest among the top-$\lfloor \frac{d}{2} \rfloor$ largest domain sizes of base views is 42, the corresponding view is numbered as b_1.

After the execution of BVG, we can make two observations. **(O1)** First, each base view is usually composed of $\lfloor \frac{d}{2} \rfloor$ (or $\lceil \frac{d}{2} \rceil$) attributes with larger domain sizes, and $\lceil \frac{d}{2} \rceil$ (or $\lfloor \frac{d}{2} \rfloor$) attributes with smaller domain sizes. **(O2)** Second, cross view generation is very likely to find a set of attributes, despite the lack of a formal guarantee; half are from b_i, another half are from b_{i+1}, and the domain size of the half from b_i is greater than that of the half from b_{i+1}. These relations are useful in determining cross views, and are further clarified in Section IV-C5.

5) Algorithm for Cross View Generation (CVG): After calculating and numbering base views through BVG, we move to the construction of cross views. As mentioned previously, for every pair of base views b_i and b_{i+1}, we generate a cross view according to Definition 6. Depending on the domain sizes of the attributes in the two base views b_i and b_{i+1}, there are four cases for cross view construction.

- (**low-to-low**) Attributes with small domain size in b_i may be paired with the attributes with small domain size in b_{i+1} (e.g., Figure 4a), so as to minimize the domain size of the generated cross view. At first glance, this may reduce the impact of noise. Nonetheless, due to the unnecessary flexibility in combining two marginals during the data synthesis (see Section IV-F), this technique may also increase the erroneous pairing of partial records, degrading the data utility of $\hat{O}$. Thus, it is necessary to avoid the low-to-low strategy.
- (**high-to-high**) Figure 4b shows the high-to-high case, where the attributes with large domain size in b_i are paired to the attributes with large domain size in b_{i+1}. This seems to imply a fine-grained record mapping, resulting in data synthesis with a better utility. Nevertheless, this technique still suffers from the problems of memory overhead and data sparsity. We often could not find a matching attribute in data synthesis, and the synthetic data generation failed as a result. Hence, we also prefer to avoid high-to-high.
- (**high-to-low** and **low-to-high**) Either high-to-low (Figure 4c) or low-to-high (Figure 4d) approaches can be used a compromise. Because the base views are combined sequentially to generate cross views, there is a subtle difference between high-to-low and low-to-high. As will be shown in the data synthesis **(S5)** (See Section IV-F), because partial records are combined from b_1 to c_1, b_2 and so on sequentially, compared to low-to-high which may be more likely to have erroneous pairings, high-to-low generation performs pairing with a more abundant amount of information on the correct linking and therefore may be more likely to generate a synthetic dataset with a better utility. In summary, we prefer using high-to-low in cross view generation.

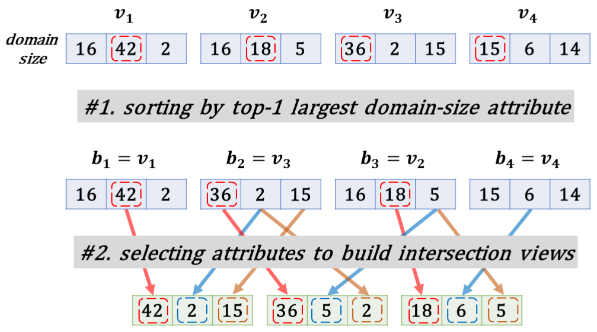

v1
v2
v3
v4
domain size
16 42 2
16 18 5
36 2 15
15 6 14
#1. sorting by top-1 largest domain-size attribute
b1 = v1
b2 = v3
b3 = v2
b4 = v4
#2. selecting attributes to build intersection views
42 2 15
36 5 2
18 6 5

Fig. 3: An illustration of invoking BVG and CVG once.

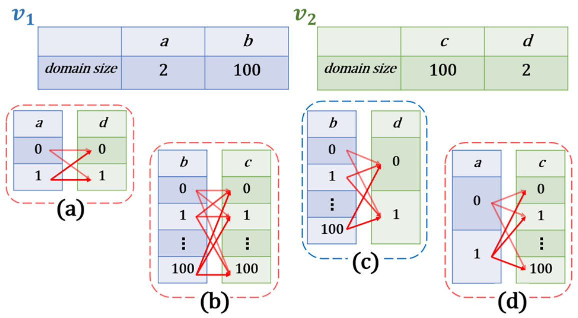


Fig. 4: Four cases for cross view generation: (a) low-to-low, (b) high-to-high, (c) high-to-low, (d) low-to-high.

As mentioned above, DPView adopts the high-to-low strategy to compose cross views. CVG (Algorithm 2) shows our algorithm for cross view construction. Here, we point out some details in implementing a high-to-low strategy.

Algorithm 2: Cross View Generation (CVG)

```
Input: $d$, $\mathbb{A} = \{a_1, \ldots, a_m\}$, $B = \{b_1, \ldots, b_{\lceil \frac{m}{d} \rceil}\}$
$C \leftarrow \emptyset$
if $d$ is even then
    for $i = 1 \sim \lceil \frac{m}{d} \rceil - 1$ do
        $set_1 \leftarrow$ select the largest $\frac{d}{2}$ attributes from $b_i$
        $set_2 \leftarrow$ select the lowest $\frac{d}{2}$ attributes from $b_{i+1}$
        $c_i \leftarrow set_1 \cup set_2$ and $C \leftarrow C \cup c_i$
else
    if $|b_i| > \lfloor \frac{d}{2} \rfloor$ then $x = \lceil \frac{m}{d} \rceil - 1$
    else $x = \lceil \frac{m}{d} \rceil - 2$
    for $i = 1 \sim x$ do
        $set_1 \leftarrow$ select the largest $\lfloor \frac{d}{2} \rfloor$ attributes from $b_i$
        $set_2 \leftarrow$ select the lowest $\lfloor \frac{d}{2} \rfloor$ attributes from $b_{i+1}$
        $c_i \leftarrow set_1 \cup set_2$
        $p \leftarrow$ select the attribute with min domain size from $b_{i+1} \setminus set_2$
        if $dom(set_1) \leq dom(set_2) \times dom(p)$ then
            $p \leftarrow$ select the attribute with min domain size from $b_i \setminus set_1$
        $c_i \leftarrow c_i \cup p$ and $C \leftarrow C \cup c_i$
return $C$
```

• For base views b_i and b_{i+1}, one can naïvely satisfy the high-to-low strategy by selecting, for example, $d-1$ attributes from b_i and one attribute from b_{i+1}. However, this imbalance may lead to poor utility of $\hat{O}$ because a partial record from b_i will be matched against another partial record from b_{i+1} by considering only one matched attribute, which is very likely to have an erroneous pairing during **(S5)**. Thus, choosing an equal number of attributes from two base views is preferrable.
• The existence of such a cross view with an equal number of attributes from two base views can be guaranteed by the sorting of base views mentioned in **(O2)**.
• Now it becomes clearer why we setup a magic number, $\lfloor \frac{d}{2} \rfloor$, in the sorting of BVG (Algorithm 1). Due to the use of high-to-low matching, a guarantee that we can find an attribute a_{j_1} from b_i and an attribute a_{j_2} from b_{i+1} satisfying $s_{j_1} \geq s_{j_2}$, where $j_1, j_2 \in [m]$ is necessary. In this sense, sorting and numbering the base views according to their respective top-$\lfloor \frac{d}{2} \rfloor$ largest domain size in a descending order attempts to ensure such a requirement. Furthermore, as each base view contributes either $\lfloor \frac{d}{2} \rfloor$ or $\lceil \frac{d}{2} \rceil$ attributes during the cross view construction, it is more stable to ensure the above requirement according to their respective top-$\lfloor \frac{d}{2} \rfloor$ largest domain size, rather than top-1 or top-d largest domain size.
• If an equal number of attributes cannot naturally be chosen from both base views (e.g., view size is odd), then we include an extra attribute from b_{i+1} in c_i (lines $13 sim 15$ in Algorithm 2) to determine whether the high-to-low strategy can be fulfilled. We accept such an arrangement if so, and change to select an extra attribute from b_i otherwise. Such a design choice can be attributed to **(O1)**; for the cross view from a pair of b_i and b_{i+1}, if we instead select an extra attribute from b_i, this extra attribute has a small domain size. Similar to the low-to-low, our experience shows that this leads to a worse utility due to the unnecessary flexibility during data synthesis.

6) Algorithm for View List Generation (VLG): We propose an iterative algorithm VLG (Algorithm 3), invoking BVG and CVG as subroutines, to generate the optimal base and cross views. In particular, VLG iterates a given number of times (e.g., x times in Algorithm 3) and returns the optimal base and cross views. On the other hand, VLG can choose to iterate an uncertain number of times with an early stop policy to find an optimal solution instead of adopting a brute force search, as prohibitively many possibilities exist for different combinations of views. For example, for a dataset with 25 attributes, to find the optimum views with $d = 5$ requires $(m \cdot d - 1)!/(d-1)! \approx 2.585 \cdot 10^{22}$ checks in a brute force manner. We note that the largest $dom(b_i)$ naturally serves as an upper bound of the domain sizes of cross views (lines $7 \sim 10$ of Algorithm 3).

Algorithm 3: View List Generation (VLG)

```
Input: $d$, $\mathbb{A} = \{a_1, \ldots, a_m\}$
Input: $x$: number of iterations that VLG executes
$D_{\text{best}} \leftarrow \infty$, $B_{\text{best}} \leftarrow \emptyset$, $C_{\text{best}} \leftarrow \emptyset$
for $run = 1 \sim x$ do
    $B \leftarrow BVG(\mathbb{A}, d)$
    $C \leftarrow CVG(\mathbb{A}, d, B)$
    for $v$ in $C$ do
        if $dom(v) >$ largest view in $B$ then
            $pass \leftarrow$ True
    if $pass$ then $continue$
    $D \leftarrow \sum_{j=1}^{|B|}(\prod_{a_i \in b_j} s_i) + \sum_{j=1}^{|C|}(\prod_{a_i \in c_j} s_i)$
    if $D < D_{best}$ then $D_{\text{best}} \leftarrow D$, $B_{\text{best}} \leftarrow B$, $C_{\text{best}} \leftarrow C$
return $B_{\text{best}}$ and $C_{\text{best}}$
```

7) Guideline for Choosing View Size d: With the observation that larger view size in a marginal implies fewer views and less splitting of privacy budget, but more severe sparsity, we know that determining the view size using only domain size information is by no means trivial. Here, we provide a guideline on how to evaluate the goodness of a specific d using only domain size information.

The total domain size D for O can be calculated as $D = s_1 \cdots s_m$. On average, each view has domain size $\frac{D \cdot d}{m}$. Here, we assume that each hypothetical marginal with d attributes is a standard Gaussian distribution whose peak probability occurs at the center area of the marginal. It may be derived that when applying a query count, a zero-mean Laplace noise will fall into $[-\rho, \rho]$ with probability γ, where $\rho = \frac{1}{\varepsilon} \ln(1-\gamma)$. Given a fixed γ, any d such that the size of the area within $(\frac{D \cdot d}{m})$-dimensional standard Gaussian distribution above $\frac{\rho}{n}$ is larger than a threshold could be a candidate for the view size d.

D. Noisy Marginal Table Construction (S3)

In the previous step, we constructed base and cross views by taking advantage of public knowledge on domain sizes of attributes. In this step, we apply the Laplace mechanism on the corresponding marginals to ensure ε-DP. This is the only step in DPView accessing O, which should be perturbed by noise.

The corresponding marginals may be easily derived given the views. Apply the Laplace mechanism to all the marginals, each of which shares the same portion of ε, is a straightforward method to ensure DP. Most of the existing DPSD generation algorithms such as DPSyn [41], PriView [46], and PrivBayes [59] followed this strategy; i.e., ε is evenly allocated to marginals. We note that though PriView proposed a notion of Expected Squared Error (ESE) to analyze noise scale, its main purpose is to simplify the view update with common attributes in the post-processing (particularly, consistency). DPSyn also follows a similar policy. However, to the best of our knowledge, DPView is the first to utilize ESE to derive an optimum budget allocation.

We begin with a brief review of ESE. Consider the construction of a DPSD from a dataset with m attributes using the flat vector approach [46], [59]. As there is only a single view, there is no need to split the privacy budget in this case. Therefore, the ESE can be calculated as $\frac{2}{\varepsilon^2}\prod_{i=1}^{m} s_i$ by summarizing the variances of all cells introduced by noises. In other words, assuming k views are used to create marginals and inject noise, the ESE of view v_i is $\frac{2}{(\varepsilon/k)^2}\prod_{a_i \in v_i} s_i$ if the privacy budget is evenly allocated.

However, equally allocating the privacy budget to all the views may not be an optimal strategy in terms of data utility. In particular, this could be beneficial for the marginal with smaller domain size. However, the marginal with larger domain size may be expected to be severely affected by the noise in this approach. Though there would be a more accurate data synthesis for the marginals with smaller domain size, the overall data synthesis still implies a likelihood of poor data utility due to the heavily perturbed marginals with larger domain size.

To remedy this problem, DPView calculates the proportion P_i of privacy budget that is to be allocated to the view v_i. Suppose there are k views $\{v_1, v_2, \dots, v_k\}$ with corresponding domain sizes $\{s_1, s_2, \dots, s_k\}$. Privacy budget allocation can be formulated as an optimization problem as follows.

$$\min \frac{s_1}{P_1^2} + \frac{s_2}{P_2^2} + \cdots + \frac{s_k}{P_k^2} \tag{2}$$

$$\text{s.t. } P_1 + P_2 + \cdots + P_k = 1 \text{ and } 0 \le P_i, \text{ for } i = 1, 2, \dots, k. \tag{3}$$

We describe the rationale behind the above optimization problem as follows. As Laplace noise needs to be added to each cell, all of the noise in a given view will be accumulated and the accumulated noise will be proportional to the number of cells. In other words, all of the noise in a given view will be accumulated and the accumulated noise will be proportional to the domain size. Thus, given k views with the same scale of noise added to each cell, to minimize the noise, we minimize $s_1 + \cdots + s_k$. However, the noise scale is controlled by Laplace noise Lap$(\frac{\Delta_f}{\varepsilon})$. Here, $\Delta_f = 1$ in DPView because Laplace noise is added to each bar (i.e., *count*) in the contingency table. Given privacy budget ε, the view v_i has $\frac{\varepsilon}{P_i}$. Moreover, as the variance of Lap$(\frac{1}{\varepsilon/P_i})$ is $\frac{2}{P_i^2}$, we need to minimize the total variance $s_1\frac{2}{P_1^2} + \cdots + s_k\frac{2}{P_k^2}$. As the constant 2 does not have impact on the result of the minimization, the above equation can be rewritten as $\frac{s_1}{P_1^2} + \frac{s_2}{P_2^2} + \cdots + \frac{s_k}{P_k^2}$.

The above optimization problem can easily find the optimum solution through the Karush-Kuhn-Tucker (KKT) condition. In particular, let $L = \frac{s_1}{P_1^2} + \frac{s_2}{P_2^2} + \cdots + \frac{s_k}{P_k^2} + \lambda(P_1 + P_2 + \cdots + P_k - 1)$, and for $i = 1, 2, \dots, k$, let $\frac{\partial L}{\partial P_i} = \frac{-2s_i}{P_i^3} + \lambda = 0$. Then, we have $P_i = (\frac{2s_i}{\lambda})^{\frac{1}{3}} > 0$, and $P_1 + P_2 + \cdots + P_k - 1 = (\frac{1}{\lambda})^{\frac{1}{3}} \cdot \sum_{i=1}^{k}(2s_i)^{\frac{1}{3}} - 1 = 0$. Thus, we can derive

$$\lambda = 2 \cdot \left(\sum_{i=1}^{k} s_i^{\frac{1}{3}}\right)^3 > 0, \text{ and thus } P_i = \left(\frac{s_i}{\left(\sum_{i=1}^{k} s_i^{\frac{1}{3}}\right)^3}\right)^{\frac{1}{3}} > 0. \tag{4}$$

Hence, the budget allocated to each view v_i would be $P_i\varepsilon$. This optimum allocation can avoid using an excessive portion of the privacy budget on the marginals with smaller domain sizes to enable a strong but useless tolerance to noise, and also allows the marginals with larger domain sizes to gain more budget to reduce the negative impact of noise.

E. Post-Processing (S4)

A post-processing step is commonly used to refine DPSDs to improve their utility. The post-processing step in DPView includes consistency, non-negativity, and normalization. We describe these in further detail below.

1) Consistency: Hay et al. [34] introduces consistency as a post-processing step. Consistency aims to reach a “consensus” of attribute values by updating attribute values through the common attributes among different views. For example, in PriView (DPSyn), if the views have common attributes, the corresponding cells are updated using the (weighted) average values. However, the consistency of PriView cannot handle non-binary data well. Simultaneously, DPSyn considers only even privacy budget allocation, which is not compatible with our configuration of the optimum privacy budget allocation in **(S3)**. Hence, based on our optimum privacy budget allocation, we formulate the process of deriving proper weights w_i in the above *weighted consistency* as an optimization problem. Then, we analytically calculate the optimum weights in the weighted consistency to improve the utility.

Weighted Consistency. Before delving into the details, we briefly overview the mechanism of weighted consistency. Let $v_1, \dots, v_x$ be x views and $v = v_1 \cap \cdots \cap v_x = \{a_{j_1}, \dots, a_{j_y}\}, j_1, \dots, j_y \in [m]$, be a view consisting of y common attributes. $T_v((a_{j_1}, \dots, a_{j_y}))$ will be updated as $T_v((a_{j_1}, \dots, a_{j_y})) = \sum_i w_i \cdot T_{v_i}((a_{j_1}, \dots, a_{j_y}))$ with weights $w_1, \dots, w_x$ in the weighted consistency. The weighted consistency degrades roughly to the standard consistency if $w_1 = \cdots = w_x = 1$. Our goal is to find an optimal set of weights such that the data can be maximized by the weighted consistency.

Formulation of Weighted Consistency. With ESE, the noise scale over v is $\sum_{C \in v} \mathrm{Var}[T_v(C)] = \sum_{C \in v}\sum_i w_i^2 \cdot \mathrm{Var}[T_{v_i}(C)] = dom(v) \cdot \sum_i w_i^2 \cdot \xi_i \cdot \mathrm{Var}_{v_i} = dom(v) \cdot \sum_i w_i^2 \cdot \xi_i \cdot (\frac{1}{P_i})^2 \cdot \mathrm{Var}_\varepsilon$, where ξ_i denotes $dom(v_i \setminus v)$, P_i denotes the proportion derived from Equation (4), Var_ε is the variance introduced by the Laplace noise Lap$(\frac{1}{\varepsilon})$ and Var_{v_i} is the variance introduced from v_i. Formally, the problem of finding the maximum weights can be formulated as

$$\min \sum_i (\frac{w_i}{P_i})^2 \cdot \xi_i \text{ subject to } \sum_i w_i = 1. \tag{5}$$

With the KKT condition, let $L = \sum_i (\frac{w_i}{P_i})^2 \cdot \xi_i + \lambda \cdot (\sum_i w_i - 1)$. For $i = 1, 2, \dots, k$, let $\frac{\partial L}{\partial w_i} = \frac{2\xi_i \cdot w_i}{P_i^2} + \lambda = 0$. We have $w_i = \frac{-\lambda P_i^2}{2\xi_i}$, and

$$\sum_i w_i = \frac{-\lambda}{2} \cdot \sum_i \frac{P_i^2}{\xi_i} = 1 \Rightarrow \lambda = \frac{-2}{\sum_i \frac{P_i^2}{\xi_i}}. \tag{6}$$

Hence, we derive $w_i = \frac{-P_i^2}{2\xi_i} \cdot (-2) / \left(\sum_j \frac{P_j^2}{\xi_j}\right) = (P_i^2) / \left(\xi_i \cdot \sum_j \frac{P_j^2}{\xi_j}\right)$, and for each C (the tuple $(a_{j_1}, \ldots, a_{j_y})$) from v-marginal, $T_v(C) = \sum_i w_i \cdot T_{v_i}(C) = \left(\sum_i [\frac{P_i^2}{\xi_i} \cdot T_{v_i}(C)]\right) / \left(\sum_j \frac{P_j^2}{\xi_j}\right)$.

Finally, each view can update its counts by $T_v(C)$ as follows. For each C' (without the loss of generality, the tuple $(a_{j_1}, \ldots, a_{j_y}, \ldots, a_{j_d})$, $y \leq d$, $j_1, \ldots, j_d \in [m]$) from v_i-marginal,

$$T_{v_i}(C') \leftarrow T_{v_i}(C') + \frac{1}{\xi_i}(T_v(C) - T_{v_i}(C)), \tag{7}$$

where C denotes the tuple $(a_{j_1}, \ldots, a_{j_y})$ from $v = v_1 \cap \cdots \cap v_x$ defined as above.

In summary, consistency can not only reduce the impact of noise by leveraging multiple noisy marginals, but also effectively reduce the possibility of erroneous pairing between views in the data synthesis (see Sections IV-E3 and IV-F).

2) Non-Negativity: Based on the observation that the count of each cell in the (noisy) marginal must be greater than or equal to 0, the cells whose counts are negative can be corrected to improve utility. Straightforward methods such as directly replacing the negative values with 0 or adding $-\mu$ to all other counts, where $-\mu$ denotes the smallest count in the marginal, do not work well because the noise scale cannot be retained. To retain the noise scale, collection of negative counts and canceling the positive counts in ascending order until depleting the negative counts has been proposed. PriView proposed ripple non-negativity, where the negative counts less than a pre-defined threshold $-\theta$, are distributed to neighboring cells.

In the present work, we propose an *iterative non-negativity* approach, shown in Figure 5. In particular, we calculate the summation q of all the negative counts and add $\frac{q}{o}$ to each of the positive counts, where o is the number of positive counts. This leads to a new collection of positive and negative counts (if any). The non-negativity stops if only positive counts remain and otherwise iterates the above process based on the collection of counts from the end of the previous iteration. Our iterative non-negativity technique retains not only the noise scale but also the shape of the distribution of the marginal.

The non-negativity techniques will destroy the consistency of the data. Therefore, after performing the non-negativity operation, we perform consistency again until the noisy marginals converge (as shown in **(S4)** in Figure 1).

3) Normalization: The total marginals may also be refined by the observation that the total counts of the noisy marginals should be n. Once the total counts of the noisy marginals are corrected to be n in a certain manner, the corrected counts could be non-integers. Each count in the contingency table should be an integer; an integralization is used to further fine-tune the noisy marginals. However, the above two corrections interfere with consistency. Thus, to better retain the noise scale and consistency, we propose *consistency-aware normalization* as follows.

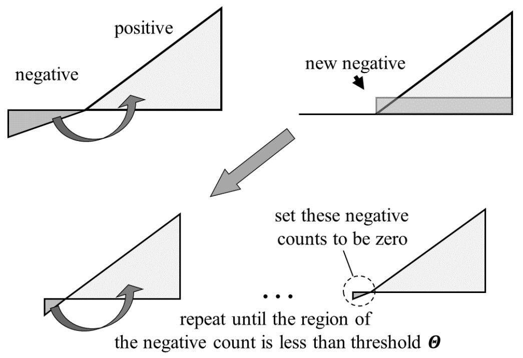

positive
negative
new negative
set these negative counts to be zero
...
repeat until the region of the negative count is less than threshold $\boldsymbol{\theta}$

Fig. 5: Iterative non-negativity in DPView.

Concretely, our proposed normalization technique divides each count into its integer and decimal parts. Extra counts are added to the corresponding cells in the integer part according to the descending order of the decimal part. This procedure simultaneously ensures the corrected marginals have only integer counts, and they sum to n. For example, consider the case that $n = 5$ and we have counts $[0.4, 3.3, 1.3]$. The integer part is $[0, 3, 1]$, which sums to 4 only, and the decimal part is $[0.4, 0.3, 0.3]$. As the first element has the largest value, we add 1 to the first element to obtain $[1, 3, 1]$[1], which sums to 5 and can be used to substitute $[0.4, 3.3, 1.3]$ thereafter.

Eliminating Negative Impact on Consistency. The above normalization also destroys the consistency among views. To eliminate the negative impact on consistency, our strategy is to choose a specific marginal to start the normalization. Given this normalized marginal, due to our design of base and cross views, we may find its *adjacent views* sharing common attributes. After fixing attribute values of the common attributes, we perform the normalization on adjacent views. We iterate the above process until all views are corrected. For example, we start from a base view b_1. On the basis of the normalization result of b_1, we fix the attribute values for the attributes from $b_1 \cap c_1$ and then perform the normalization on the marginal with attributes from c_1. By doing so, we can accomplish the normalization without compromising consistency (as shown in **(S4)** in Figure 1). We describe our method of selecting an initial view to begin and normalizing a view from a given view below.

Choosing Initial View for Normalization. The choice of the view to start the normalization has a considerable impact on utility, as its normalization result will be propagated to the other views by design. The proposed method selects the marginal with a *higher degree of concentration*. In other words, we skip the marginals with low counts in most cells. This can be attributed to the fact that such marginals are naturally more resilient to noise. Superficially, the marginal with the smallest domain size seems to act as the marginal with the highest degree of concentration. However, this is not always the case because the degree of concentration is also dependent on the data distribution. Here, similar to the case in Section IV-C7, it may be observed that a zero-mean Laplace noise will fall into $[-\rho, \rho]$ with probability γ, where $\rho = -\frac{\Delta F}{\varepsilon} \ln(1-\gamma)$. We argue that, given a confidence probability γ, the presence of many counts above ρ is an indicator of a higher degree of

[1]In the case of $n = 5$ and the counts $[0.3, 3.3, 1.3]$, we select a random position and add one to it to break a tie. For example, if we pick the second element, the result is $[0, 4, 1]$.

concentration. We then use the summation of the counts above ρ as the degree of concentration for the marginal.

Propagating Normalization to Adjacent Views. Here, we describe our proposed method to normalize v_2 given the normalized v_1 without compromising the consistency between v_1 and v_2. The counts in v_2 after the consistency-aware normalization will inevitably be modified. This count modification also guarantees that we can always generate n records during data synthesis.

Consider two adjacent views with k common attributes; without loss of generality, we consider $v_1 = \{a_1, \dots, a_d\}$ and $v_2 = \{a_{d-k+1}, \dots, a_{2d-k}\}$. For each possible value $(\alpha_1, \dots, \alpha_k)$ of attributes $a_{d-k+1}, \dots, a_d$, we have $h_{\text{target}} = T_{v_1}((a_{d-k+1}, \dots, a_d) = (\alpha_1, \dots, \alpha_k))$. Given $(\alpha_1, \dots, \alpha_k)$, for every value $(\beta_1, \dots, \beta_{d-k})$ of $a_{d+1}, \dots, a_{2d-k}$, we calculate $h_i = T_{v_2}((a_{d-k+1}, \dots, a_{2d-k}) = (\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_{d-k}))$, $1 \le i \le c_{d+1} \cdots c_{2d-k}$. Then, h_i is updated as $\frac{h_i}{h_{c_{d+1}} \cdots h_{c_{2d-k}}} \cdot h_{\text{target}}$. Consider an extreme case that h_{target} is non-zero but there is no partial record in v_2 with $(a_{d-k+1}, \dots, a_d) = (\alpha_1, \dots, \alpha_k)$; i.e., $T_{v_2}((a_{d-k+1}, \dots, a_d) = (\alpha_1, \dots, \alpha_k)) = 0$. In such case, we generate h_{target} partial records in v_2 whose values in attributes $a_{d-k+1}, \dots, a_d$ are all $\alpha_1, \dots, \alpha_k$ and whose values in attributes $a_{d+1}, \dots, a_{2d-k}$ are randomly selected. Note that $h_{c_{d+1}} \cdots h_{c_{2d-k}}$ will not be large, since most of them will be zero, due to the sparsity of the underlying marginal. As the updated h_is could be floating numbers, the previous integralization is used to correct them.

F. Data Synthesis (S5)

After post-processing, we perform data synthesis to generate a DPSD. In other words, given base and cross views, the objective of data synthesis is to generate n m-dimensional records by linking proper partial records from different views. In contrast to most of the existing DPSD generation algorithms relying on sampling-based data synthesis, we particularly note that DPView takes a fundamentally different linking-based approach. We also note that data synthesis **(S5)** is guaranteed to generate n records because the normalization in Section IV-E3 synchronized the counts of views.

view combination is a critical subroutine for data synthesis. Concretely, view combination takes as input two *adjacent views* v_1 and v_2 sharing a set A of common attributes, and returns the $|v_1|+|v_2|-|A|$-dimensional records sharing common attribute values. In general, our data synthesis starts with b_1. More specifically, we perform view combination of b_i and c_i, and of c_i and b_{i+1} alternately for $i = 1, \dots, d-1$.

1) Maximum Cardinality Matching Problem for View Combination: View combination is, in fact, an optimization problem. As a motivating example, consider the case that $v_1 = \{a_1, a_2, a_3\}$ and $v_2 = \{a_2, a_3, a_4\}$. Assume that v_1 has only single one partial record $[a_2 = 0, a_3 = 0]$ that can be linked to the partial records, $[a_4 = 3]$ and $[a_4 = 4]$. Also assume that v_1 has two partial records, $[a_2 = 0, a_3 = 1]$ and $[a_2 = 0, a_3 = 1]$ that can be linked to a partial record $[a_4 = 3]$. In such a case, if $[a_2 = 0, a_3 = 0]$ is linked to $[a_4 = 3]$, then $[a_2 = 0, a_3 = 1]$ can link to nothing. Thus, view combination of two views can be modeled as a problem of finding a maximum cardinality matching in a bipartite graph, the left side consisting of partial records from v_1 and the right side of partial records from v_2. The edge exists if two partial records share the same attribute value for the common attributes. As the maximum cardinality matching problem aims to find a matching containing as many edges as possible, the maximum cardinality matching of such a bipartite graph provides an optimum view combination result for two views.

2) Heuristic for View Combination: For a massive dataset, the state-of-the-art Hopcroft–Karp algorithm for the maximum cardinality matching problem [31] remains inefficient. Here, we propose an efficient heuristic alternative to handle the problem. Consider two adjacent views, v_1 and v_2, sharing a set $A = \{a_1, \dots, a_k\}$ of common attributes. Note that v_1 could be a view from the combination of the other views and the $|v_1| \ge d$. We sort the partial records in v_1 according to the corresponding counts in descending order. Subsequently, higher priorities are assigned to the partial records with more counts. Eventually, starting from the highest priority, each partial record in v_1 is paired with a partial record in v_2; i.e., some partial records in v_1 are prioritized to find a match in v_2 according to their counts. Consider the example in Section IV-F1. Compared to the partial record $[a_2 = 0, a_3 = 0]$ appearing only once, $[a_2 = 0, a_3 = 1]$ is assigned a higher more priority, because there are two $[a_2 = 0, a_3 = 1]$s. As a consequence, one $[a_2 = 0, a_3 = 1]$ will be paired with $[a_4 = 3]$, another $[a_2 = 0, a_3 = 1]$ will be paired with none, and $[a_2 = 0, a_3 = 0]$ will be paired with $[a_4 = 4]$. Eventually, the above generates two three-dimensional records. In the case that a partial record in v_1 has more than one choice in v_2, it is paired with a random record. This randomness incurs utility loss, but we reduce such uncertainty in Section IV-G.

G. DPView+

In this section, we propose DPView+ as a variant of DPView. DPView+ and DPView share the same general approach, with the difference being that more views are created in DPView+. In particular, in **(S2)**, the extra views, *upper cross views*, are defined and generated as follows.

Definition 8: Given d, two base views b_i and b_{i+1}, and the corresponding cross view c_i, the upper cross view u_i consists of the attributes from $b_i \cup b_{i+1} \setminus c_i$.

Similar to the three different cases in Section IV-C2, the construction of upper cross views also has three cases. Figure 6 shows an example, where the last cross view has a smaller size of **Type-II**, and there will be a vacant upper cross view of**Type-III**.

Then, Laplace noise is applied to marginals with base, cross, and upper cross views in **(S3)**, despite the increased budget splitting. In **(S4)**, the consistency and non-negativity procedures remain the same. However, to avoid the contradicting updates from two other views, upper cross views do not participate in the normalization.

As described in Section IV-F2, the upper cross views help the view combination reduce the possibility of erroneous pairings in **(S5)**. This can be attributed to the reference use of upper cross views. In particular, when performing the view combination, the existence of u_i provides extra information on which combination of partial records can be eliminated. As a concrete example, consider the views $b_1 = \{a_1, a_2, a_3\}$, $b_2 = \{a_4, a_5, a_6\}$, $c_1 = \{a_2, a_3, a_4\}$, and $u_1 = \{a_1, a_5, a_6\}$, as illustrated in Type-I of Figure 6. Consider the view combination of the view $\{a_1, a_2, a_3, a_4\}$ (the resultant view from the view combination of b_1 and c_1) and b_2. Despite the heuristic in Section IV-F2, erroneous pairings are often obtained. For example, consider $T_{\{a_1,a_2,a_3,a_4\}}((a_4 = 0)) = 2$ with 2 partial records with $a_4 = 0$ in b_2. No information is available on how to pair a specific partial record from $\{a_1, a_2, a_3, a_4\}$ with another partial record from b_2 in a correct way. Nonetheless,

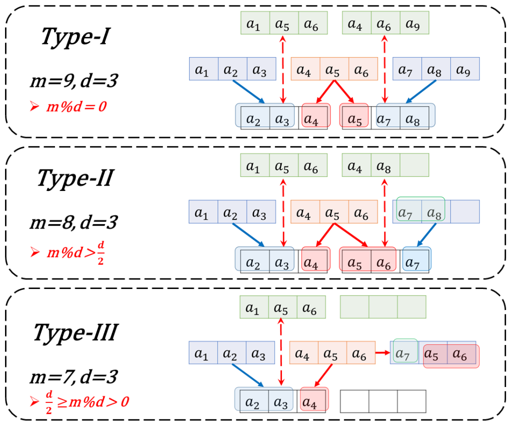


Fig. 6: Three cases of upper cross view generation.

with upper cross view $u_1 = \{a_1, a_5, a_6\}$, the possibilities can be reduced from b_2 for a specific partial record from $\{a_1, a_2, a_3, a_4\}$ to pair. For example, for a partial record $(a_1 = 4, a_2 = 9, a_3 = 4, a_4 = 0)$, if there are two partial records $(a_4 = 0, a_5 = 5, a_6 = 3)$ and $(a_4 = 0, a_5 = 2, a_6 = 1)$ in b_2 and a partial record $(a_1 = 4, a_5 = 5, a_6 = 3)$ in u_1, then we can be confident in combining $(a_1 = 4, a_2 = 9, a_3 = 4, a_4 = 0)$ and $(a_4 = 0, a_5 = 5, a_6 = 3)$.

V. Evaluation

A. *Experiment Setup*

1) Dataset Description: We use the following three datasets to evaluate the data utility of DPView.

• Adult [1]: This is a dataset widely used by machine learning practitioners to test their binary classification accuracy. In particular, Adult consists of 48842 records, each of which has six numeric attributes, eight categorical attributes, and one binary class label.

• Census-Income [12]: This dataset reports weighted census data extracted from the 1994 and 1995 current population surveys. Census-Income consists of 299285 records, each of which has 33 numeric attributes, 7 categorical attributes, and one binary class label.

• KDDCup99 [37]: This dataset, used by KDD Cup 1999, includes a wide variety of intrusions simulated in a military network environment. Census-Income consists of 494021 records, each of which has 34 numeric attributes, 8 categorical attributes, and one binary class label.

2) Tasks and Evaluation Metrics: Due to the popularity of data mining and machine applications, we evaluate the data utility of DPView according to the following data analysis tasks.

• (Classification) As classification is the most popular machine learning task, it is necessary to evaluate the data utility of DPSD in terms of classification accuracy. In particular, we use the DPSD to train the classification models and derive the accuracy as a measure of the data utility. We use 12 classification models, including AdaBoost, bagging, Bernoulli Naive Bayes, decision tree, Gaussian Naive Bayes, Linear Discriminant Analysis (LDA), Light Gradient Boosted Machine (LightGBM), logistic regression (LR), multi-layer perceptron (MLP), random forest, support vector machine (SVM), and XGBoost, to examine the data utility.

• (k-Way Marginals) k-way marginals are "the workhorse of data analysis" [4]. Thus, we compute 1-, 2-, and 3-way marginals from the DPSD and derive the average ℓ_1 error to measure data utility.

• (Range Query) Range query is the most natural type of query to retrieve the dataset. We generate 1000 random range queries, each containing 3 random attributes. We calculate $\frac{1}{|Q|}\sum_{q_i \in Q} |c_i - \hat{c}_i|$, where Q is the set of queries and c_i ($\hat{c}_i$) is the ratio of records that fall in the range of query q_i in O ($\hat{O}$), as a measure of the data utility.

3) Competitors: We consider the following competitors for the DPSD generation.

• (PGM [44]) PGM is a parametric approach and is the 1st place winner of the NIST 2018 Differential Privacy Synthetic Data Challenge. Given a set of pre-defined marginals, PGM estimates an MRF that best fits the marginals. Our implementation of PGM is adapted from the official implementation [13] and third party implementation [14], because the official implementation is customized to the NIST Challenge and thus cannot be used in our evaluation.

• (PATE-GAN [36]) PATE-GAN can be thought of as a special type of DP-GAN. Due to the lack of the official implementation, our PATE-GAN implementation is adapted from [14].

• (PrivBayes [59]) PrivBayes is a parametric approach; it approximates the high-dimensional data distribution using a number of low-dimensional marginal distributions, which will be determined by highly correlated attributes. PrivBayes is the 3rd place winner of the NIST Challenge. We implement PrivBayes from scratch.

• (PrivSyn [62]) PrivSyn is a non-parametric approach and is an extension of DPSyn, the 2nd place winner of the NIST Challenge. We implement PrivSyn from scratch.

4) Miscellaneous Setting: We carry out all the experiments on a desktop with Intel i7-6700HQ CPU 2.60GHz and 16 GB memory. All algorithms are implemented in Python 3.7. In our experiments, the view sizes for Adult and KDDCup99 are fixed to be 5 and the view size for Census-Income is fixed to be 6. Depending on the datasets under the consideration, the number of views varies. In particular, DPView has 5 views for Adult, 13 views for Census-Income, and 15 views for KDDCup99. DPView+ has 7 views for Adult, 19 views for Census-Income, and 22 views for KDDCup99.

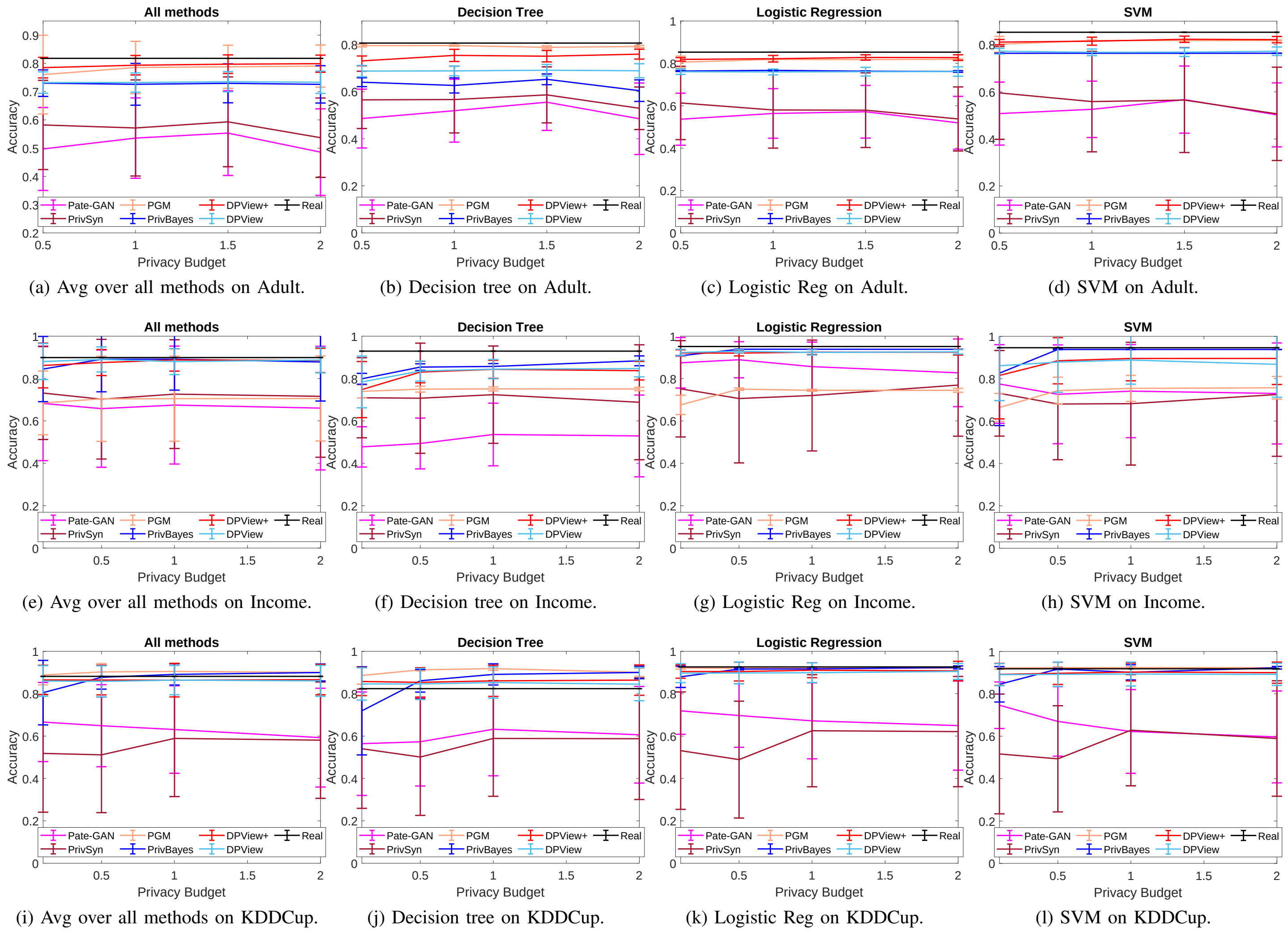
Similar to PrivSyn [62], DPView considers unbound DP [38] and has no clue on the exact number n of records in O. We thus use the average of the total counts of marginals as an approximation of n.

We implement machine learning algorithms by using scikit-learn in our experiments, where the default parameter settings are used. Each experiment result shown in the plots later is an average of three independent experiments.

B. *End-to-End Comparison*

1) Classification: As mentioned in Section V-A2, we test twelve classification algorithms. Due to the space limitation, we explicitly show the individual results for the decision tree, logistic regression, and SVM in the second, third, and fourth columns of Figure 7, respectively. However, we show the accuracy derived by averaging the classification accuracy for all the twelve classification results in the first column of Figure 7.

From the experimental results in Figure 7, we can make some observations below. First, PATE-GAN leads to an awful accuracy. According to our experience, the DP-GAN approaches can well synthesize image data, but for the tabular data, their performance degrades substantially. This can



(a) Avg over all methods on Adult. (b) Decision tree on Adult. (c) Logistic Reg on Adult. (d) SVM on Adult.

(e) Avg over all methods on Income. (f) Decision tree on Income. (g) Logistic Reg on Income. (h) SVM on Income.

(i) Avg over all methods on KDDCup. (j) Decision tree on KDDCup. (k) Logistic Reg on KDDCup. (l) SVM on KDDCup.

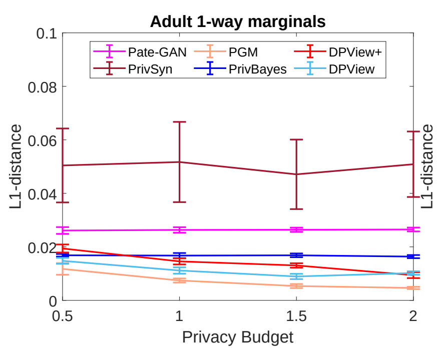
Fig. 7: Classification accuracy for different machine learning algorithms on three datasets.

be attributed to the fact that GANs may erroneously learn unnecessary or noisy correlations. Second, PrivSyn also has a pessimistic accuracy. As we already follow their suggestion on the parameter setting in PrivSyn (e.g., the parameters $\alpha = 2$, $\beta = 0.5, etc.$), we conjecture that their parameters are not universally suitable for arbitrary data. Third, DPView+ outperforms DPView in nearly all cases. This can be attributed to the fact that upper cross views whose purpose is to reduce the probability of the erroneous pairings exhibit a consistency-like behavior, compensating the utility loss due to the worse splitting of the budget. Fourth, the accuracy of DPView+, PGM, and PrivBayes is higher than that of non-private data in Figure 7j. An explanation is that though decision trees are prone to overfitting, DP can alleviate overfitting [18], [19]. This enables these DPSDs to gain better accuracy.

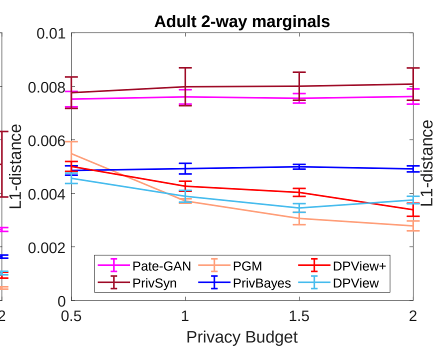
2) k-Way Marginals and Range Query: As the k-way marginals reflect the quality of low-dimensional correlation preserved in the DPSD, the range query reflects the quality of high-dimensional correlation preserved in the DPSD. Figure 8 shows that DPView effectively captures the low-dimensional correlations. This is due to the design of DPView, in which base views and cross views, despite their low dimensionality, naturally preserve the low-dimensional correlations. Figure 9 shows that DPView also captures the high-dimensional correlations. In addition to the reduced noise scale thanks to the data-independent view generation, This can partially be attributed to our design of consistency-aware normalization, which "correct" from one view to another, ensuring the capture of high-dimensional correlation.

VI. Discussion

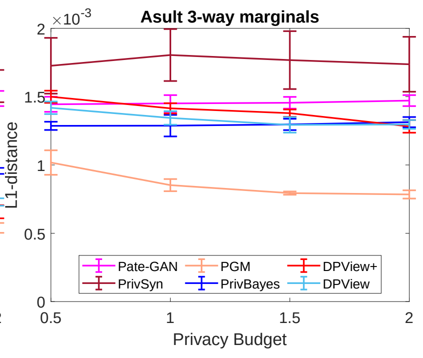
Here, we pay attention only to the automatic DPSD generation. PGM does not propose a marginal selection algorithm, and consequently, one either has to pick marginals manually or considers the other data-dependent marginal selection methods. S. Takagi et al. [50] evenly partition O into two parts and derive the necessary parameters from one part directly. However, in such a case, the privacy of half records will be compromised. PrivSyn claims to be an automatic DPSD generation system. It indeed can generate the DPSD automatically, but the resultant data utility can hardly be ensured (see Section V-B). Bowen and Snoke [9] argue that PrivBayes is an easy-to-use algorithm because it strikes a balance between the data utility and the hardness of using it. In this sense, DPView can also reach a similar level of balance compared to PrivBayes. In particular, despite the necessary parameters such



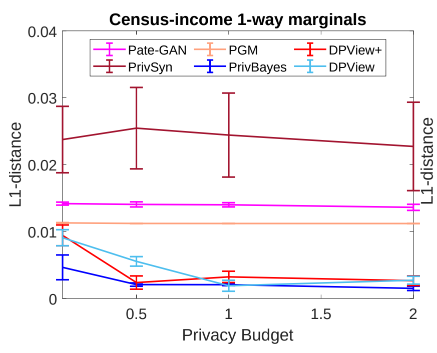
(a) 1-way marginals on Adult.



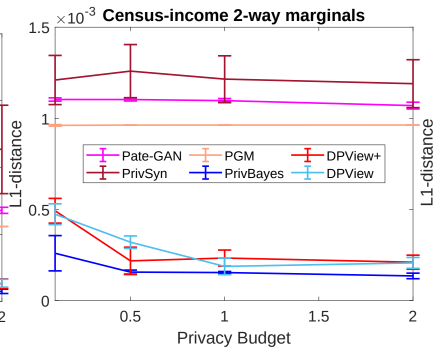
(b) 2-way marginals on Adult.



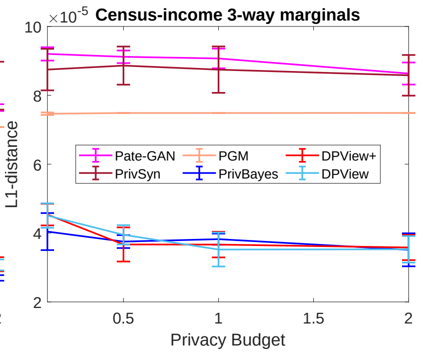
(c) 3-way marginals on Adult.



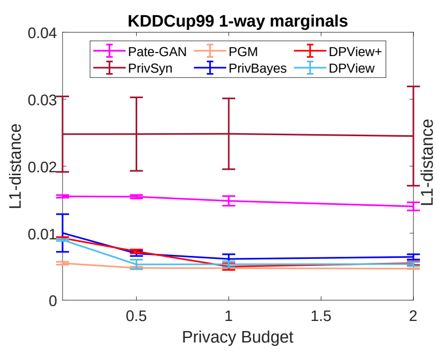
(d) 1-way marginals on Income.



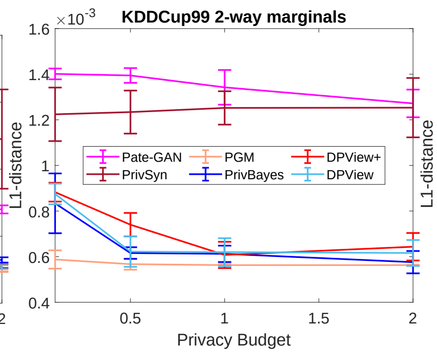
(e) 2-way marginals on Income.



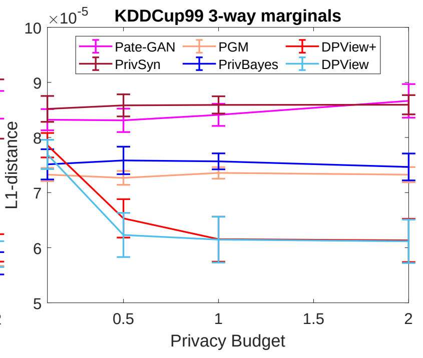
(f) 3-way marginals on Income.



(g) 1-way marginals on KDDCup.

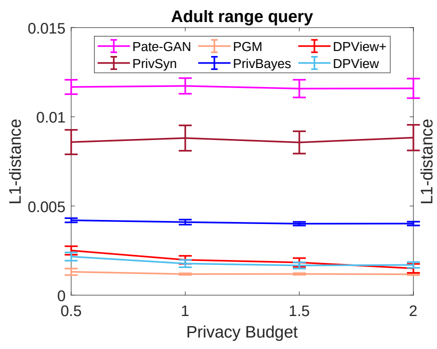


(h) 2-way marginals on KDDCup.

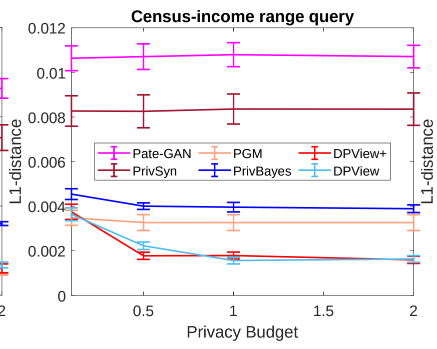


(i) 3-way marginals on KDDCup.

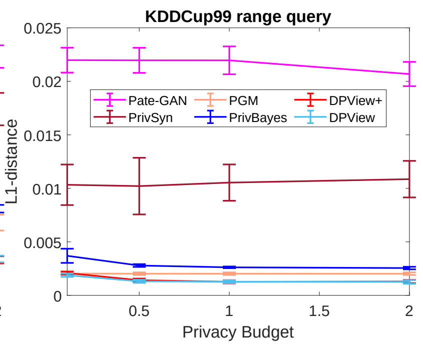
Fig. 8: *k*-way marginals on three datasets.



(a) Range query on Adult.



(b) Range query on Income.



(c) Range query on KDDCup.

Fig. 9: Range query on three datasets.

as the stopping criteria in the view generation and confidence probability in the view size determination, they do not lead to a fluctuation of data utility. As a result, DPView can be regarded as an automatic DPSD generation system.

For the design challenge **(C1)**, data-independent algorithms, Algorithms 1∼3, are proposed to preserve the correlation among attributes by taking advantage of public domain information. Thus, DPView alleviates the privacy budget problem because there is no need to spend budget on preserving attribute correlation in view list generation. DPView is conceptually similar to works [46], [41], [62], the characteristic of its data-independent view list generation in DPView contributes to the main novelty. On the other hand, the problem of **(C2)** exists mainly due to the use of data-dependant algorithms for view list generation. Therefore, DPView avoids **(C2)**. For **(C3)**, because DPView uses equal-size buckets, the number of buckets is the only parameter to be determined. In other words, the data synthesis can be nearly fully automated in DPView.

VII. Conclusion

In this paper, we present DPView as an algorithm for differentially private data synthesis. The proposed data-independent view generation algorithm significantly alleviates the problem of privacy budget splitting. We perform the budget allocation and weighted consistency in an analytically optimal manner. The proposed iterative non-negativity and consistency-aware normalization reconcile the discrepancy among noisy marginals. All of the above results in the utility improvement of the DPSD from DPView.

References

[1] https://archive.ics.uci.edu/ml/datasets/Adult

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[3] Nazmiye Ceren Abay, Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham, and Latanya Sweeney. Privacy preserving synthetic data release using deep learning. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2018.

[4] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. *ACM international conference on Management of data (SIGMOD)*, 2007.

[5] D. Boob, R. Cummings, D. Kimpara, U. Tantipongpipat, C. Waites, and K. Zimmerman. Differentially Private Synthetic Data Generation via GANs. Available at: https://www.herox.com/UnlinkableDataChallenge/round/358/entry/20533

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. *International Conference on Learning Representations (ICLR)*, 2015.

[7] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. *ACM Symposium on Theory of Computing (STOC)*, 2008.

[8] M. Bun and T. Steinke. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. *Theory of Cryptography Conference (TCC)*, 2016.

[9] C. M. Bowen and J. Snoke. Comparative Study of Differentially Private Synthetic Data Algorithms from the NIST PSCR Differential Privacy Synthetic Data Challenge. *Journal of Privacy and Confidentiality*, vol. 11, no. 1, 2021.

[10] V. Bindschaedler, R. Shokri, C. A. Gunter. Plausible Deniability for Privacy-Preserving Data Synthesis, *Proceedings of Very Large Data Bases (PVLDB)*, 2017.

[11] B. K Beaulieu-Jones, Z. S. Wu, C. Williams, R. Lee, S P Bhavnani, J. B. Byrd, and C. S. Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*, vol. 12 no. 7, 2019.

[12] https://archive.ics.uci.edu/ml/datasets/Census-Income+

[13] https://github.com/ryan112358/private-pgm

[14] https://github.com/BorealisAI/private-data-generation

[15] Q. Chen, C. Xiang, M. Xue, B. Li, N. Borisov, D. Kaafar, and H. Zhu. Differentially Private Data Generative Models, *arXiv:1812.02274*, 2018.

[16] R. Chen, Q. Xiao, Y. Zhang, and J. Xu. Differentially Private High-Dimensional Data Publication via Sampling-Based Inference. *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2015.

[17] C. Dwork. Differential Privacy. *International conference on Automata, Languages and Programming (ICALP)*, 2006.

[18] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, A. Roth. Generalization in Adaptive Data Analysis and Holdout Reuse. *Advances in neural information processing systems (NeuIPS)*, 2015.

[19] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, A. Roth. Generalization in Adaptive Data Analysis and Holdout Reuse. *ACM on Symposium on Theory of Computing (STOC)*, 2015.

[20] M. Dell'Amico and S. Martello. Bounds for the cardinality constrained $P||C_{\max}$ problem. *Journal of Scheduling*, vol. 4, no. 3, pp. 123-138, 2001.

[21] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results, *ACM Symposium on Theory of Computing (STOC)*, 2009.

[22] C. Dwork and A. Roth. The Algorithmic Foundations of Differential Privacy. Now Publishers Inc. 2014.

[23] C. Dwork and G. N. Rothblum. Concentrated Differential Privacy. *arxiv:1603.01887*, 2016.

[24] L. Frigerio, A. Santana de Oliveira, L. Gomez, and P. Duverger. Differentially private generative adversarial networks for time series, continuous, and discrete open data. *IFIP International Conference on ICT Systems Security and Privacy Protection (SEC)*, 2019.

[25] D. Gordon. Covering Designs.

[26] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *Advances in neural information processing systems (NeuIPS)*, 2017.

[27] M. Gaboardi, E. J. Gallego Arias, J. Hsu, A. Roth, and Z. Steven Wu. Dual query: Practical private query release for high dimensional data. *International Conference on Machine Learning (ICML)*, 2014.

[28] Anupam Gupta, Moritz Hardt, Aaron Roth, and Jon Ullman. Privately releasing conjunctions and the statistical query barrier. *ACM Symposium on Theory of Computing (STOC)*, 2011.

[29] C. Ge. S. Mohapatra, X. He, and I. F. Ilyas. Kamino: Constraint-Aware Differentially Private Data Synthesis. *arXiv:2012.15713*, 2020.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems (NeuIPS)*, 2014.

[31] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*. vol. 2, no. 4, pp. 225-231. 1971.

[32] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. *Advances in Neural Information Processing Systems (NeuIPS)*, 2012.

[33] Moritz Hardt and Guy Rothblum. A multiplicative weights mechanism for interactive privacy-preserving data analysis. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2010.

[34] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of Very Large Data Bases (PVLDB)*, 2010.

[35] B. Jayaraman and D. Evans. Evaluating Differentially Private Machine Learning in Practice. *USENIX Security Symposium*. 2019.

[36] J. Jordon, J. Yoon, and M. van der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. *International Conference on Learning Representations (ICLR)*, 2018.

[37] https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data

[38] N. Li, M. Lyu, D. Su, and W. Yang. Differential Privacy: From Theory to Practice. *Synthesis Lectures on Information Security, Privacy, and Trust.* Morgan Claypool, 2016.

[39] N. Li, W. Qardaji, D. Su, and J. Cao. PrivBasis: Frequent Itemset Mining with Differential Privacy, *Proceedings of Very Large Data Bases (PVLDB)*, 2012.

[40] M. Lyu, D. Su, and N. Li. Understanding the Sparse Vector Technique for Differential Privacy, *Proceedings of Very Large Data Bases (PVLDB)*, 2017.

[41] N. Li, Z. Zhang, and T. Wang. Dpsyn: Differentially private synthetic data publication. Available at: https://github.com/usnistgov/PrivacyEngCollabSpace/tree/master/tools/de-identification/Differential-Privacy-Synthetic-Data-Challenge-Algorithms/DPSyn

[42] Ilya Mironov. Rényi Differential Privacy. *IEEE Computer Security Foundations Symposium*, 2017.

[43] W. Michiels, J. H. M. Korst, E. H. L. Aarts, and J. van Leeuwen. Performance ratios for the differencing method applied to the balanced number partitioning problem. *Symposium on Theoretical Aspects of Computer Science (STACS)*, 2003.

[44] R. McKenna, D. Sheldon, and G. Miklau. Graphical-model based estimation and inference for differential privacy. *International Conference on Machine Learning (ICML)*, 2019.

[45] F. McSherry, and K. Talwar. Mechanism design via differential privacy. *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2007.

[46] W. Qardaji, W. Yang, and N. Li. Priview: practical differentially private release of marginal contingency tables. *ACM international conference on Management of data (SIGMOD)*, 2014.

[47] E. Shen and T. Yu. Mining Frequent Graph Patterns with Differential Privacy. *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2013.

[48] A. Triastcyn, and B. Faltings. Generating artificial data for private deep learning. *AAAI Spring Symposium Series*, 2019.

[49] R. Torkzadehmahani, P. Kairouz, and B. Paten. 2020. DP-CGAN: Differentially Private Synthetic Data and Label Generation. *arxiv:2001.09700*, 2020.

[50] S. Takagi, T. Takahashi, Y. Cao and M. Yoshikawa. P3GM: Private high-dimensional data release via privacy preserving phased generative model. *IEEE Conference on Data Engineering (ICDE)*, 2021.

[51] J. Thaler, J. R. Ullman, S. P. Vadhan. Algorithms for Privately Releasing Marginals. *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2012.

[52] U. Tantipongpipat, C. Waites, D. Boob, A. Ankit Siva, and R. Cummings. Differentially private mixed-type data generation for unsupervised learning. *arXiv:1912.03250*, 2019.

[53] S. Vigna. A weighted correlation index for rankings with ties. In *International Conference on World Wide Web (WWW*, 2015.

[54] Richard Wu, Aoqian Zhang, Ihab F. Ilyas, and Theodoros Rekatsinas. Attention-based Learning for Missing Data Imputation in HoloClean. *Conference on Machine Learning and Systems (MLSys)*, 2020.

[55] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou. Differentially Private Generative Adversarial Network. *arxiv:1802.06739*, 2018.

[56] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. Modeling tabular data using conditional gan. In *Advances in Neural Information Processing Systems (NeuIPS)*, 2019.

[57] X. Xiao, G. Wang, and J. Gehrke. Differential Privacy via Wavelet Transforms. *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 8, pp. 1200-1214, 2011.

[58] J. Xu, Z. Zhang, X. Xiao, Y. Yang, G. Yu. Differentially Private Histogram Publication. *IEEE International Conference on Data Engineering (ICDE)*, 2012.

[59] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Privbayes: Private data release via bayesian networks. In *ACM SIGMOD International Conference on Management of Data*, pp. 1423–1434, 2014.

[60] X. Zhang, S. Ji, and T. Wang. Differentially private releasing via deep generative model. *arXiv:1801.01594*, 2018.

[61] J. Zhang, K. Mouratidis, and H. Pang. Heuristic Algorithms for Balanced Multi-way Number Partitioning. *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2011.

[62] Z. Zhang, T. Wang, N. Li, J. Honorio, M. Backes, S. He, J. Chen, and Y. Zhang. PrivSyn: Differentially Private Data Synthesis. *USENIX Security Symposium*, 2021.

Chih-Hsun Lin is a Ph.D. student at Department of Computer Science, National Yang Ming Chiao Tung University. His research interests include network security and data privacy.

Chia-Mu Yu is currently an associate professor at National Yang Ming Chiao Tung University, Taiwan. He had academic visits at IBM Thomas J. Watson research center, Harvard University, Imperial College London, University of Padova, and the University of Illinois at Chicago. He received Hwa Tse Roger Liang Junior Chair Professor, MOST Yong Scholar Fellowship, ACM/IICM K. T. Li Young Researcher Award, Observational Research Scholarship from Pan Wen Yuan Foundation, and MOST Project for Excellent Junior Research Investigators, Taiwan. He serves as an Associate Editor for IEEE Internet of Things Journal. His research interests include differentially private mechanism design, cloud storage security, and IoT security.

Chun-Ying Huang joined National Yang Ming Chiao Tung University as an Associate Professor in 2016, where he has been a Professor since 2018 and is currently a Professor with the Department of Computer Science. His research interests include system security, multimedia networking, and mobile computing. He is a member of ACM. He was a recipient of the 2014 ACM Taipei/Taiwan Chapter K. T. Li Young Researcher Award.