PrivBayes: Private Data Release via Bayesian Networks

JUN ZHANG, Google Inc., China
GRAHAM CORMODE, University of Warwick, UK
CECILIA M. PROCOPIUC, Google Inc., USA
DIVESH SRIVASTAVA, AT&T Labs–Research, USA
XIAOKUI XIAO, Nanyang Technological University, Singapore

Privacy-preserving data publishing is an important problem that has been the focus of extensive study. The state-of-the-art solution for this problem is differential privacy, which offers a strong degree of privacy protection without making restrictive assumptions about the adversary. Existing techniques using differential privacy, however, cannot effectively handle the publication of high-dimensional data. In particular, when the input dataset contains a large number of attributes, existing methods require injecting a prohibitive amount of noise compared to the signal in the data, which renders the published data next to useless.

To address the deficiency of the existing methods, this paper presents PrivBayes, a differentially private method for releasing high-dimensional data. Given a dataset D, PrivBayes first constructs a Bayesian network $\mathcal{N}$, which (i) provides a succinct model of the correlations among the attributes in D and (ii) allows us to approximate the distribution of data in D using a set $\mathcal{P}$ of low-dimensional marginals of D. After that, PrivBayes injects noise into each marginal in $\mathcal{P}$ to ensure differential privacy and then uses the noisy marginals and the Bayesian network to construct an approximation of the data distribution in D. Finally, PrivBayes samples tuples from the approximate distribution to construct a synthetic dataset, and then releases the synthetic data. Intuitively, PrivBayes circumvents the curse of dimensionality, as it injects noise into the low-dimensional marginals in $\mathcal{P}$ instead of the high-dimensional dataset D. Private construction of Bayesian networks turns out to be significantly challenging, and we introduce a novel approach that uses a surrogate function for mutual information to build the model more accurately. We experimentally evaluate PrivBayes on real data and demonstrate that it significantly outperforms existing solutions in terms of accuracy.

CCS Concepts: • **Security and privacy → Data anonymization and sanitization**;

Additional Key Words and Phrases: Differential privacy, synthetic data generation, bayesian network

ACM Reference format:
Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. PrivBayes: Private Data Release via Bayesian Networks. *ACM Trans. Database Syst.* 42, 4, Article 25 (October 2017), 41 pages.
https://doi.org/10.1145/3134428

25

This work was supported in part by European Commission Marie Curie Integration Grant PCIG13-GA-2013-61820, by the DSAIR center at Nanyang Technological University, by Microsoft Research Asia under a gift grant, by the Ministry of Education (Singapore) under Grant ARC19/14, and by AT&T under an unrestricted gift grant.
Authors' addresses: J. Zhang, Google Shanghai, 100 Century Avenue, Shanghai 200120, China; email: junzha@google.com; G. Cormode, Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK; email: g.cormode@warwick.ac.uk; C. M. Procopiuc, Google New York, 111 8th Ave, New York, NY 10011, USA; email: mpro@google.com; D. Srivastava, AT&T Labs - Research, 1 AT&T Way, Bedminster, NJ 07921, USA; email: divesh@research.att.com, X. Xiao, School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798; email: xkxiao@ntu.edu.sg.

ACM 0362-5915/2017/10-ART25 $15.00
https://doi.org/10.1145/3134428

1 INTRODUCTION

The problem of privacy-preserving data publishing (PPDP) has become increasingly important in recent years. Often, we encounter situations where a data owner wishes to make available a data set without revealing private, sensitive information. For example, this arises in the case of revealing detailed demographic information about citizens (census), patients (health data), investors (financial data), and so on. In each example, there are many potential uses and users for the data: for social analysis, for medical research, for freedom-of-information, and other legal disclosure reasons. The canonical case in PPDP is when the database can be modeled as a table, where each row may contain information about an individual (say, details of their medical status or employment information). Then, the aim is to release some manipulated version of this information so this can still be used for the intended purpose, but the privacy of individuals in the database is preserved.

Following many attempts to formally define the requirements of privacy, the current state-of-the-art solution is to seek the differential privacy guarantee [18]. Informally, this model requires that what can be learned from the released data is (approximately) the same, whether or not any particular individual was included in the input database. This model offers strong privacy protection and does not make any limiting assumptions about the power of the notional adversary: it remains a strong model even in the face of an adversary with much background knowledge and reasoning power.

Putting differential privacy into practice remains a challenging problem. Since its proposal, there have been many efforts to develop mechanisms and processes for data release for different kinds of input databases and for different objectives for the end use. However, it seems that all existing techniques encounter difficulties when trying to release even moderately high-dimensional data—that is, an input table with half a dozen columns or more. The reasons for these problems are twofold:

Output Scalability: Most algorithms (see, e.g., Reference [46]) either explicitly or implicitly represent the database as a histogram (i.e., a vector x) of size equal to the *domain size*, that is, the product of cardinalities of the attributes. For many natural data sets, the domain size m is orders of magnitude larger than the data size n [15]. Hence, these algorithms become inapplicable for any realistic dataset with a moderate-to-high number of attributes. For example, a million row table with ten attributes, each of which has 20 possible values, results in a domain size (and hence an output size) of $m = 20^{10} \approx 10$TB, which is very unwieldy and slow to use compared to the input that can be measured in megabytes.

Signal-to-Noise Ratio: When the high-dimensional database is represented as a vector x, the average count in each entry, given by n/m, is typically very small. Once noise is added to x (or some transformation of it) to obtain another vector x^*, the noise completely dominates the original signal, making the published vector x^* next to useless. For example, if the table above has size $n = 1$M, the average entry count is $n/m = 10^{-7}$. By contrast, the average noise injected to achieve, for example, differential privacy with parameter $\varepsilon = 0.1$ has expected magnitude around 10. Even if the data is skewed in the domain space, that is, there are some entries $x[i]$ with high counts, such peaks are infrequent, and so the vast majority of published values $x^*[i]$ are useless.

1.1 Related Work

A full survey of methods to realize differential privacy is beyond the scope of this work. Here, we identify the most related efforts and discuss why they cannot fully solve the problems above.

Initial efforts released projections of the data on subsets of dimensions, via Fourier decompositions [2]. This reduces the impact of higher dimensionality but requires the data owner to

determine (somehow) which set of dimensions are of interest and for the data to be mapped into a space where a Fourier decomposition can be computed. Subsequent work followed this template, for example, by searching for meaningful "subcubes" of the datacube representation to release privately [17]. These can be aggregated to answer lower-dimensional cube queries, where the signal-to-noise ratio in each cell is significantly higher than in the original domain. A major limitation is that the running time is exponential in the dimensionality of the domain, making it inapplicable for all but small sets. Accuracy is improved by additional post-processing of the output to restore "consistency" of the counts mandated by the structure (e.g., using the fact that counts in a hierarchy should sum to the count of an ancestor) [2, 17, 27]; however, this improvement does not overcome the inherent error incurred in high dimensions. Recently, Chen et al. [9] also proposed to utilize probabilistic graphical models in modelling sensitive high-dimensional datasets. They introduced the novel sparse vector technique [20, 25] to the private learning of graphical models, resulting in a significant improvement in utility over previous methods. However, this technique is shown to be flawed [10], as its privacy analysis is erroneous. This invalidates the solution proposed in Reference [9], and we have not seen any easy way to address this issue.

Another approach is to use data reduction techniques to avoid dimensionality issues. For example, Cormode et al. [15] propose various sampling mechanisms to reduce the size of (and time to produce) the output x^*, while approximately preserving the accuracy of subset-sum queries. However, the accuracy guarantee is in terms of using the entire vector x^*, rather than the original vector x, which degrades rapidly with data dimensionality. The approach in Reference [14] tries to keep the domain size of the output small, and the density of data within the new domain high, by adaptively grouping some of the attribute values. In the example above, suppose that on each of the ten attributes, we grouped the 20 values into four groups. Thus, we reduce the output size from 20^{10} to $4^{10} \approx 1$MB. This coarser grid representation of the data loses precision, and in this example still leads to small average counts of the order of 1. Cormode et al. [14] address the latter problem by using spatial decompositions to define irregular grids over the domain of x, such that the count $x[i]$ in each grid cell is sufficiently large. This makes sense only for range queries and requires all attributes to have an ordering.

Other approaches find other ways to recast the input data and release it, so the noise from the privacy transformation has less impact on certain statistics. In particular, Xiao et al. [46] make use of the wavelet transformation of data. This addresses range queries, and means that the noise incurred by a range query scales proportionately to the logarithm of its length, rather than to its length directly. The value of this is confined primarily to low-dimensional databases where range queries are anticipated. More generally, the matrix mechanism of Li and Miklau [32, 33] and subsequent related approaches [23, 26, 47, 48] take in a query workload (expressed in terms of a weighted set of inner-product queries), and seek to release a version of the database that minimizes the noise for these queries. The cost of this optimization can be high and critically assumes the foreknowledge of the query distribution.

The above discussion focuses on methods that produce an output in a general form that can be used flexibly for a variety of subsequent analyses. There is also a large class of results that instead generate a description of the output of a specific algorithm computed under privacy, for example, the result of building a classifier for the data. Prominent examples include the work of McSherry and Mironov [38] to mask the preferences of individual raters in recommendation systems; Rastogi and Nath [42] to release time-series data based on leading Fourier coefficients; and McSherry and Mahajan [37] for addressing common queries over network trace data. Differential privacy has also been applied to other sophisticated data analysis tasks, for example, coresets for summarizing geometric data [21], building classifiers in the form of decision trees [22] or support vector machines [43], and mining the occurrence of frequent patterns [4, 34].

Several frameworks have been proposed to solve specific classes of optimization problems. For example, Chaudhuri and Monteleoni [7], Chaudhuri et al. [8], and Kifer et al. [29] consider differentially private convex empirical risk minimization, which can be applied to a wide range of optimization problems (e.g., logistic regression and support vector machines). Zhang et al. [50] propose the *PrivGene* framework, which is a combination of genetic algorithms and an enhanced version of the *exponential mechanism* for differentially private model fitting. Nissim et al. [41], Smith [45], and Mohan et al. [40], respectively, present and implement the *sample and aggregate* framework that can be used for any analysis task whose results are not affected by the number of records in the database. While this class of methods obtains generally good results for the target problem, it requires fixing this objective at the time of data release, and limits the applicability of the output for other uses.

1.2 Our Contributions

In this article, we propose a powerful solution to the problem of publishing differentially private high-dimensional data. Unlike the bulk of prior work, which focuses on optimizing the output for specific workloads (e.g., range queries, cube queries), we aim to approximate the high-dimensional distribution of the original data with a data-dependent set of well-chosen low-dimensional distributions, in the belief that, for a sufficiently accurate approximation, the resulting data will maintain high accuracy for almost any type of (linear or non-linear) query. The result is an output data set that is drawn from the obtained model while obeying the same schema and format of the original input. Since our approach is query-independent, many different queries can be evaluated (accurately) on the same set of released data. Query-independence means that our approach may be weaker than approaches that target a particular query set; however, we show empirically that the gap is small or non-existent in many natural cases. By working in low-dimensional spaces, we avoid the signal-to-noise problem. Although we compare to the full-dimensional distribution for evaluation purposes, our approach never needs to compute this explicitly, thus avoiding the scalability problem.

To achieve this goal, we start from the well-known *Bayesian network* model, which is widely studied in the statistical and machine learning communities [30]. Bayesian networks combine low-dimensional distributions to approximate the full-dimensional distribution of a data set, and are a simple but powerful example of a graphical model. Our algorithm, dubbed PrivBayes, consists of the following steps:

1. (*Network learning*) We describe how to compute a differentially private Bayesian network that approximates the full-dimensional distribution via the exponential mechanism (EM). This step requires new theoretical insights, which are described in Section 3. We improve on this basic approach by defining a new score function for use in EM, which results in significantly better networks being found. The definition and computation of this function are one of our main technical contributions; see Section 4.

2. (*Distribution learning*) We explain how to compute the necessary differentially private distributions of the data in the subspaces of the Bayesian network, via the Laplace Mechanism.

3. (*Data synthesis*) We show how to generate synthetic data from the differentially private Bayesian network, without explicitly materializing the full-dimensional distribution.

In Section 6, we provide an extensive experimental evaluation of the accuracy of the synthetic datasets generated above, over workloads of linear and non-linear queries. In each case, we compare to prior methods specifically designed to optimize the accuracy for that type of workload. Our experiments show that PrivBayes is often *more* accurate than any prior method, even though

it is not optimized for any specific type of query. When PrivBayes is less accurate than some prior method, the accuracy loss is small and, we believe, an acceptable tradeoff, since PrivBayes offers a generic solution that does not require prior knowledge of the workload and works well on many different types of queries.

1.3 Relation to the Preliminary Version

A preliminary version of this article appeared in Reference [49]. Compared to the preliminary version [49], the current version contains three significant extensions. First, the solution in the preliminary version requires each attribute in the input data be transformed into a set of *binary* attributes (i.e., the domain of each attribute is $\{0, 1\}$), which destroys the semantics of natural attributes in the dataset and degrades the utility of the output data; in contrast, the current version presents several advanced techniques that enable PrivBayes to handle attributes with general domains without requiring the binary transformation (see Section 5). Second, the current version presents extensive experiments that evaluate the extended PrivBayes and demonstrate that, in terms of data utility, it significantly outperforms the preliminary version of PrivBayes that *binarizes* attributes. The current version also includes new experiments on the impact of PrivBayes's internal parameters β and θ (see Section 6). Finally, the current version includes more detailed explanations and proofs that were omitted in the preliminary version (much of this additional detail appears in the Appendix).

2 PRELIMINARIES

This section reviews two concepts closely related to our work, namely, differential privacy and Bayesian networks.

2.1 Differential Privacy

Let D be a sensitive dataset to be published. Differential privacy requires that any release of information about D should be done via a randomized algorithm G, such that the output of G does not reveal much information about any particular tuple in D. The formal definition of differential privacy is as follows:

Definition 2.1 (ε-Differential Privacy [19]). A randomized algorithm G satisfies ε-differential privacy, if for any two datasets D_1 and D_2 that differ only in one tuple, and for any possible output O of G, we have

$$\Pr[G(D_1) = O] \leq e^{\varepsilon} \cdot \Pr[G(D_2) = O], \tag{1}$$

where $\Pr[\cdot]$ denotes the probability of an event.

In what follows, we say that two datasets are neighboring if they differ in only one tuple, that is, the values of one tuple are changed while the rest are identical. While there are many approaches to achieving differential privacy, we rely on the two best known and most widely used, namely, the *Laplace mechanism* [19] and the *exponential mechanism* [39].

The Laplace mechanism releases the result of a function F that takes as input a dataset and outputs a set of numeric values. Given F, the Laplace mechanism transforms F into a differentially private algorithm, by adding i.i.d. noise (denoted as η) into each output value of F. The noise η is sampled from a *Laplace distribution* $\text{Lap}(\lambda)$ with the following pdf: $\Pr[\eta = x] = \frac{1}{2\lambda} e^{-|x|/\lambda}$. Dwork et al. [19] prove that the Laplace mechanism ensures ε-differential privacy if $\lambda \geq S(F)/\varepsilon$, where $S(F)$ is the *sensitivity* of F:

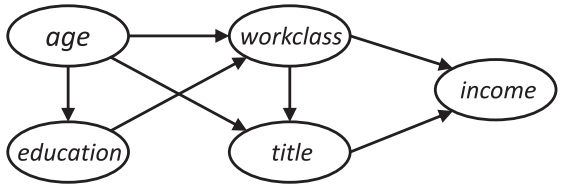


Fig. 1. A Bayesian network $\mathcal{N}_1$ over five attributes.

Definition 2.2 (Sensitivity [19]). Let F be a function that maps a dataset into a fixed-size vector of real numbers. The *sensitivity* of F is defined as

$$S(F) = \max_{D_1, D_2} \|F(D_1) - F(D_2)\|_1 , \tag{2}$$

where $\|\cdot\|_1$ denotes the L_1 norm, and D_1 and D_2 are any two neighboring datasets.

Intuitively, $S(F)$ measures the maximum possible change in F's output when we modify one arbitrary tuple in F's input.

When F's output is categorical instead of numeric, the Laplace mechanism does not apply, but the exponential mechanism [39] can be used instead. The exponential mechanism releases a differentially private version of F, by sampling from F's output domain Ω. The sampling probability for each $\omega \in \Omega$ is determined based on a user-specified *score function* f_s, which takes as input any dataset D and any element $\omega \in \Omega$, and outputs a numeric score $f_s(D, \omega)$ that measures the quality of ω: a larger score indicates that ω is a better output with respect to D. More specifically, given a dataset D, the exponential mechanism samples $\omega \in \Omega$ with a probability proportional to $\exp(f_s(D, \omega)/2\Delta)$, where Δ is a scaling factor that controls the degree of privacy protection. McSherry and Talwar [39] show that the exponential mechanism achieves ε-differential privacy if $\Delta \geq S(f_s)/\varepsilon$, where $S(f_s)$ is defined as

$$S(f_s) = \max_{D_1, D_2, \omega'} |f_s(D_1, \omega') - f_s(D_2, \omega')| , \tag{3}$$

for D_1 and D_2 any two neighboring datasets, and ω' any element in Ω. For convenience, we also refer to $S(f_s)$ as the *sensitivity* of f_s, as it is similar in form to sensitivity as defined above.

Both mechanisms can be applied quite generally; however, to be effective, we seek to ensure that the noise introduced does not outweigh the signal in the data, and that it is computationally efficient to apply the mechanism. This requires a careful design of what functions to use in the mechanisms.

2.2 Bayesian Network

Let $\mathcal{A}$ be the set of attributes on a dataset D. We regard D as a joint probability distribution over the cross-product of $\mathcal{A}$'s attribute domains. A Bayesian network on $\mathcal{A}$ is a way to compactly describe this distribution, by specifying conditional independence among certain attributes in $\mathcal{A}$. In particular, a Bayesian network is a directed acyclic graph (DAG) that (i) represents each attribute in $\mathcal{A}$ as a node, and (ii) models conditional independence among attributes in $\mathcal{A}$ using directed edges. As an example, Figure 1 shows a Bayesian network over a set $\mathcal{A}$ of five attributes, namely, *age*, *education*, *workclass*, *title*, and *income*. For any two attributes $X, Y \in \mathcal{A}$, there exist three possibilities for the relationship between X and Y:

Case 1: Direct Dependence. There is an edge between X and Y, say, from Y to X. This indicates that for any tuple in D, its distribution on X is determined (in part) by its value on Y. We define Y as a *parent* of X, and refer to the set of all parents of X as its *parent set*. For example, in Figure 1,

Table 1. The Attribute-parent Pairs in $\mathcal{N}_1$

i	X_i	Π_i
1	*age*	$\emptyset$
2	*education*	{*age*}
3	*workclass*	{*age*, *education*}
4	*title*	{*age*, *workclass*}
5	*income*	{*workclass*, *title*}

the edge from *workclass* to *income* indicates that the income distribution depends on the type of job (and also on title).

Case 2: Weak Conditional Independence. There is a path (but no edge) between Y and X. Assume without loss of generality that the path goes from Y to X. Then, X and Y are *conditionally independent* given X's parent set. For instance, in Figure 1, there is a two-hop path from *age* to *income*, and the parent set of *income* is {*workclass*, *title*}. This indicates that, given workclass and job title of an individual, her income and age are conditionally independent.

Case 3: Strong Conditional Independence. There is no path between Y and X. Then, X and Y are conditionally independent given any of X's and Y's parent sets.

Let d denote the size of $\mathcal{A}$. Formally, a Bayesian network $\mathcal{N}$ over $\mathcal{A}$ is defined as a set of d *attribute-parent (AP) pairs*, $(X_1, \Pi_1), \ldots, (X_d, \Pi_d)$, such that

1. Each X_i is a unique attribute in $\mathcal{A}$;
2. Each Π_i is a subset of the attributes in $\mathcal{A} \setminus \{X_i\}$. We say that Π_i is the parent set of X_i in $\mathcal{N}$;
3. For any $1 \leq i < j \leq d$, we have $X_j \notin \Pi_i$, that is, there is no edge from X_j to X_i in $\mathcal{N}$. This ensures that the network is acyclic, namely, it is a DAG.

We define the *degree* of $\mathcal{N}$ as the maximum size of any parent set Π_i in $\mathcal{N}$. For example, Table 1 shows the AP pairs in the Bayesian network $\mathcal{N}_1$ in Figure 1; $\mathcal{N}_1$'s degree equals 2, since the parent set of any attribute in $\mathcal{N}_1$ has a size at most two.

Let $\Pr[\mathcal{A}]$ denote the full distribution of tuples in database D. The d AP pairs in $\mathcal{N}$ essentially define a way to approximate $\Pr[\mathcal{A}]$ with d conditional distributions $\Pr[X_1 \mid \Pi_1], \Pr[X_2 \mid \Pi_2], \ldots, \Pr[X_d \mid \Pi_d]$. In particular, under the assumption that any X_i and any $X_j \notin \Pi_i$ are conditionally independent given Π_i, we have

$$\begin{aligned}\Pr[\mathcal{A}] &= \Pr[X_1, X_2, \ldots, X_d] \\ &= \Pr[X_1] \cdot \Pr[X_2 \mid X_1] \cdot \Pr[X_3 \mid X_1, X_2] \ldots \Pr[X_d \mid X_1, \ldots X_{d-1}] \\ &= \prod_{i=1}^{d} \Pr[X_i \mid \Pi_i]. \end{aligned} \tag{4}$$

Let $\Pr_{\mathcal{N}}[\mathcal{A}] = \prod_{i=1}^{d} \Pr[X_i \mid \Pi_i]$ be the above approximation of $\Pr[\mathcal{A}]$ defined by $\mathcal{N}$. Intuitively, if $\mathcal{N}$ accurately captures the conditional independence among the attributes in $\mathcal{A}$, then $\Pr_{\mathcal{N}}[\mathcal{A}]$ would be a good approximation of $\Pr[\mathcal{A}]$. In addition, if the degree of $\mathcal{N}$ is small, then the computation of $\Pr_{\mathcal{N}}[\mathcal{A}]$ is relatively simple as it requires only d low-dimensional distributions $\Pr[X_1 \mid \Pi_1], \Pr[X_2 \mid \Pi_2], \ldots, \Pr[X_d \mid \Pi_d]$. Low-degree Bayesian networks are the core of our solution to release high-dimensional data. Table 2 shows notations that will be frequently used in this article.

Table 2. Table of Notations

Notation	Description
D	A sensitive dataset to be published
n	The number of tuples in D
$\mathcal{A}$	The set of attributes in D
d	The number of attributes in $\mathcal{A}$
$\mathcal{N}$	A Bayesian network over $\mathcal{A}$
$\Pr[\mathcal{A}]$	The distribution of tuples in D
$\Pr_{\mathcal{N}}[\mathcal{A}]$	An approximation of $\Pr[\mathcal{A}]$ defined by $\mathcal{N}$
$\text{dom}(X)$	The domain of random variable X

3 SOLUTION OVERVIEW

This section presents an overview of PrivBayes, our solution for releasing a high-dimensional dataset D in a differentially private manner. PrivBayes runs in three phases:

1. Construct a k-degree Bayesian network $\mathcal{N}$ over the attributes in D, using an ε_1-differentially private method. (k is a small value that can be chosen automatically by PrivBayes.)
2. Use an ε_2-differentially private algorithm to generate a set of *conditional distributions* of D, such that for each AP pair (X_i, Π_i) in $\mathcal{N}$, we have a noisy version of the conditional distribution $\Pr[X_i \mid \Pi_i]$. (We denote this noisy distribution as $\Pr^*[X_i \mid \Pi_i]$.)
3. Use the Bayesian network $\mathcal{N}$ (constructed in the first phase) and the d noisy conditional distributions (constructed in the second phase) to derive an approximate distribution of the tuples in D, and then sample tuples from the approximate distribution to generate a synthetic dataset D^*.

In short, PrivBayes utilizes a low-degree Bayesian network $\mathcal{N}$ to generate a synthetic dataset D^* that approximates the high dimensional input data D. The construction of $\mathcal{N}$ is highly non-trivial, as it requires carefully selecting AP pairs and the value of k to derive a close approximation of D without violating differential privacy. By contrast, the second and third phases of PrivBayes are relatively straightforward. In the following, we will clarify the details of these latter phases, and prove the privacy guarantee of PrivBayes; the algorithm for PrivBayes's first phase will be elaborated in Section 4.

Generation of Noisy Conditional Distributions. Suppose that we are given a k-degree Bayesian network $\mathcal{N}$. To construct the approximate distribution $\Pr_{\mathcal{N}}[\mathcal{A}]$, we need d conditional distributions $\Pr[X_i \mid \Pi_i]$ ($i \in [1, d]$), as shown in Equation (4). Algorithm 1 illustrates how the distributions specified by our algorithm can be derived in a differentially private manner. In particular, for any $i \in [k+1, d]$, the algorithm first materializes the joint distribution $\Pr[X_i, \Pi_i]$ (Line 3), and then injects Laplace noise into $\Pr[X_i, \Pi_i]$ to obtain a noisy distribution $\Pr^*[X_i, \Pi_i]$ (Lines 4 and 5). To enforce the fact that these are probability distributions, all negative numbers in $\Pr^*[X_i, \Pi_i]$ are set to zero, then all values are normalized to maintain a total probability mass of 1 (Line 5).[1] After that, based on $\Pr^*[X_i, \Pi_i]$, the algorithm derives a noisy version of the conditional distribution $\Pr[X_i \mid \Pi_i]$, denoted as $\Pr^*[X_i \mid \Pi_i]$ (Line 6). The scale of the Laplace noise

[1] More generally, we could apply additional post-processing of distributions, in the spirit of References [2, 17, 27], to reflect the fact that lower degree distributions should be consistent. For simplicity and brevity, we omit such optimizations from this presentation.

ALGORITHM 1: NoisyConditionals $(D, \mathcal{N}, k)$: returns $\mathcal{P}^*$

1 initialize $\mathcal{P}^* = \emptyset$;
2 **for** $i = k + 1$ **to** d **do**
3 materialize the joint distribution $\Pr[X_i, \Pi_i]$;
4 generate differentially private $\Pr^*[X_i, \Pi_i]$ by adding noise $\mathrm{Lap}(\frac{2(d-k)}{n\varepsilon_2})$;
5 set negative values in $\Pr^*[X_i, \Pi_i]$ to 0 and normalize;
6 derive $\Pr^*[X_i \mid \Pi_i]$ from $\Pr^*[X_i, \Pi_i]$; add it to $\mathcal{P}^*$;
7 **for** $i = 1$ **to** k **do**
8 derive $\Pr^*[X_i \mid \Pi_i]$ from $\Pr^*[X_{k+1}, \Pi_{k+1}]$; add it to $\mathcal{P}^*$;
9 **return** $\mathcal{P}^*$;

added to $\Pr[X_i, \Pi_i]$ is set to $2(d-k)/n\varepsilon_2$, which ensures that the generation of $\Pr^*[X_i, \Pi_i]$ satisfies $(\varepsilon_2/(d-k))$-differential privacy, since $\Pr[X_i, \Pi_i]$ has sensitivity $2/n$. Meanwhile, the derivation of $\Pr^*[X_i \mid \Pi_i]$ from $\Pr^*[X_i, \Pi_i]$ does not incur any privacy cost, as it only relies on $\Pr^*[X_i, \Pi_i]$ instead of the input data D.

Overall, Lines 2–6 of Algorithm 1 construct $d-k$ noisy conditional distributions $\Pr^*[X_i \mid \Pi_i]$ $(i \in [k+1, d])$, and they satisfy ε_2-differential privacy, since each $\Pr^*[X_i \mid \Pi_i]$ is $(\varepsilon_2/(d-k))$-differentially private. This is due to the composability property of differential privacy [18]. In particular, composability indicates that when a set of m algorithms satisfy differential privacy with parameters $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m$, respectively, the set of algorithms as a whole satisfies $(\sum_i \varepsilon_i)$-differential privacy.

After $\Pr^*[X_{k+1} \mid \Pi_{k+1}], \ldots, \Pr^*[X_d \mid \Pi_d]$ are constructed, Algorithm 1 proceeds to generate $\Pr^*[X_i \mid \Pi_i]$ $(i \in [1, k])$. This generation, however, does not require any additional information from the input data D. Instead, we derive $\Pr^*[X_i \mid \Pi_i]$ $(i \in [1, k])$ directly from $\Pr^*[X_{k+1}, \Pi_{k+1}]$, which has been computed in Lines 2–6 of Algorithm 1. Such derivation is feasible, since our algorithm for constructing the Bayesian network $\mathcal{N}$ (to be clarified in Section 4) ensures that $X_i \in \Pi_{k+1}$ and $\Pi_i \subset \Pi_{k+1}$ for any $i \in [1, k]$. Since each $\Pr^*[X_i \mid \Pi_i]$ $(i \in [1, k])$ is derived from $\Pr^*[X_{k+1}, \Pi_{k+1}]$ without inspecting D, the construction of $\Pr^*[X_i \mid \Pi_i]$ does not incur any privacy overhead. Therefore, Algorithm 1 as a whole is ε_2-differentially private. Example 3.1 illustrates Algorithm 1.

Example 3.1. Given the Bayesian network $\mathcal{N}_1$ in Figure 1, Algorithm 1 constructs three noisy joint distributions $\Pr^*[a, e, w]$, $\Pr^*[a, w, t]$ and $\Pr^*[w, t, i]$ (attributes are denoted by their initials). Based on $\Pr^*[a, w, t]$ and $\Pr^*[w, t, i]$, Algorithm 1 derives noisy conditional distributions $\Pr^*[t \mid a, w]$ and $\Pr^*[i \mid w, t]$, respectively. In addition, the algorithm uses $\Pr^*[a, e, w]$ to derive three other conditional distributions $\Pr^*[a]$, $\Pr^*[e \mid a]$, and $\Pr^*[w \mid a, e]$. Given these five conditional distributions, the input tuple distribution is approximated as

$$\Pr^*_{\mathcal{N}_1}[a, e, w, t, i] = \Pr^*[a] \cdot \Pr^*[e \mid a] \cdot \Pr^*[w \mid a, e] \cdot \Pr^*[t \mid a, w] \cdot \Pr^*[i \mid w, t].$$

Generation of synthetic data. Even with the simple closed-form expression in Equation (4), it is still time and space consuming to directly sample from $\Pr^*_{\mathcal{N}}[\mathcal{A}]$ by computing the probability for each element in the domain of $\mathcal{A}$. Fortunately, the Bayesian network $\mathcal{N}$ provides a means to perform sampling efficiently without materializing $\Pr^*_{\mathcal{N}}[\mathcal{A}]$. As shown in Equation (4), we can sample each X_i from the conditional distribution $\Pr^*[X_i \mid \Pi_i]$ independently, without considering any attribute not in $\Pi_i \cup \{X_i\}$. Furthermore, the properties of $\mathcal{N}$ (discussed in Section 2.2) ensure that $X_j \notin \Pi_i$ for any $j > i$. Therefore, if we sample X_i $(i \in [1, d])$ in increasing order of i, then by

the time X_j ($j \in [2, d]$) is to be sampled, we must have sampled all attributes in Π_j, that is, we will be able to sample X_j from $\Pr^*[X_j \mid \Pi_j]$ given the previously sampled attributes. That is to say, the sampling of X_j does not require the full distribution $\Pr^*_{\mathcal{N}}[\mathcal{A}]$.

With the above sampling approach, we can generate an arbitrary number of tuples from $\Pr^*_{\mathcal{N}}[\mathcal{A}]$ to construct a synthetic database D^*. In this article, we consider the size of D^* is set to n, that is, the same as the number of tuples in the input data D. The dataset D^* can then be used for analysis; clearly, it may be better suited for some tasks than others, which we study in our experimental evaluation. Sampling the same number of tuples means that we have a synthetic database that is directly comparable with the input. If the modeling assumption is valid (i.e., the data is well-modeled by a Bayesian network), then we can imagine that the original input D is also a sample of n tuples from a Bayesian model, so the choice to sample this many tuples is appropriate. For some applications, it may be possible to answer queries directly from the model, or results may be improved by drawing a larger sample. For simplicity, however, we adopt the sample of size n for our experimental study.

Privacy Guarantee. The correctness of PrivBayes directly follows the composability property of differential privacy [18]. In particular, the first and second phases of PrivBayes require direct access to the input database, and each consumes ε_1 and ε_2 privacy budget, respectively. No access to the original database is invoked during the third (sampling) phase. The results of first two steps, that is, the Bayesian network $\mathcal{N}$ and the set of noisy conditional distributions, are sufficient to generate the synthetic database D^*. Therefore, we have the following theorem.

Theorem 3.2. PrivBayes *satisfies* $(\varepsilon_1 + \varepsilon_2)$*-differential privacy.*

This theorem implies a partitioning of the total privacy budget ε for end-to-end ε-differential privacy into ε_1 and ε_2. To recast in these terms, we add a parameter β to PrivBayes, which assigns $\varepsilon_1 = \beta\varepsilon$ and $\varepsilon_2 = (1 - \beta)\varepsilon$. Intuitively, β should balance the quality of network learning and distribution learning phases in PrivBayes; therefore, we might believe that an even split would be a good starting point. We refine this view by providing an empirical analysis of suitable choices of β in Section 6.4.

4 PRIVATE BAYESIAN NETWORKS

This section presents our solution for constructing differentially private Bayesian networks. We will first introduce a non-private algorithm for Bayesian network construction (in Section 4.1), and then explain how the algorithm can be converted into a differentially private solution (in Sections 4.2 and 4.3).

4.1 Non-Private Methods

Suppose that we aim to construct a k-degree Bayesian network $\mathcal{N}$ on a dataset D containing a set $\mathcal{A}$ of attributes. Ideally, $\mathcal{N}$ should provide an accurate approximation of the tuple distribution in D, that is, $\Pr_{\mathcal{N}}[\mathcal{A}]$ should be close to $\Pr[\mathcal{A}]$. A natural question is under what condition will $\Pr_{\mathcal{N}}[\mathcal{A}]$ closely approximate $\Pr[\mathcal{A}]$? We make use of standard notions from information theory to measure this. The *entropy* of a random variable X over its domain $\mathrm{dom}(X)$ is denoted by $H(X) = -\sum_{x \in \mathrm{dom}(X)} \Pr[X = x] \log \Pr[X = x]$,[2] and $I(\cdot, \cdot)$ denotes the *mutual information* between two variables as

$$I(X, \Pi) = \sum_{x \in \mathrm{dom}(X)} \sum_{\pi \in \mathrm{dom}(\Pi)} \Pr[X = x, \Pi = \pi] \log \frac{\Pr[X = x, \Pi = \pi]}{\Pr[X = x] \Pr[\Pi = \pi]}, \tag{5}$$

[2] All logarithms used in this article are to the base 2.

ALGORITHM 2: GreedyBayes (D, k): returns $\mathcal{N}$

1 initialize $\mathcal{N} = \emptyset$ and $V = \emptyset$;
2 randomly select an attribute X_1 from $\mathcal{A}$; add $(X_1, \emptyset)$ to $\mathcal{N}$; add X_1 to V;
3 **for** $i = 2$ **to** d **do**
4 initialize $\Omega = \emptyset$;
5 for each $X \in \mathcal{A} \backslash V$ and each $\Pi \in \binom{V}{k}$, add (X, Π) to Ω;
6 select a pair (X_i, Π_i) from Ω with the maximal mutual information $I(X_i, \Pi_i)$;
7 add (X_i, Π_i) to $\mathcal{N}$; add X_i to V;
8 **return** $\mathcal{N}$;

where $\Pr[X, \Pi]$ is the joint distribution of X and Π, and $\Pr[X]$ and $\Pr[\Pi]$ are the marginal distributions of X and Π, respectively. The *KL-divergence* [16] from $\Pr_{\mathcal{N}}[\mathcal{A}]$ to $\Pr[\mathcal{A}]$ measures the difference between the two probability distributions, and is defined by

$$D_{KL}(\Pr[\mathcal{A}] \parallel \Pr_{\mathcal{N}}[\mathcal{A}]) = -\sum_{i=1}^{d} I(X_i, \Pi_i) + \sum_{i=1}^{d} H(X_i) - H(\mathcal{A}). \tag{6}$$

We seek a Bayesian network representation so the approximate distribution has a small KL-divergence to the original distribution. In Equation (6), the term $\sum_{i=1}^{d} H(X_i) - H(\mathcal{A})$ is solely decided by $\Pr[\mathcal{A}]$, which is fixed once the input database D is given. Hence, the KL-divergence from $\Pr_{\mathcal{N}}[\mathcal{A}]$ to $\Pr[\mathcal{A}]$ is small (in which case they closely approximate each other), if and only if $\sum_{i=1}^{d} I(X_i, \Pi_i)$ is maximized. Therefore, the construction of $\mathcal{N}$ can be modeled as an optimization problem, where we aim to choose a parent set Π_i for each attribute X_i in D to maximize $\sum_{i=1}^{d} I(X_i, \Pi_i)$.

For the case when $k = 1$, Chow and Liu show that greedily picking the next edge based on the maximum mutual information is optimal, leading to the celebrated notion of Chow-Liu trees [12]. However, as shown in Reference [11], this optimization problem is NP-hard when $k > 1$. For this reason, heuristic algorithms (e.g., hill-climbing, genetic algorithms, and simulated annealing) are often employed in practice [36]. In the context of differential privacy, however, a different calculus applies: these methods incur a high cost in terms of sensitivity and so incur a large amount of noise. That is, these algorithms make many queries to the data, so making them differentially private entails large perturbations that lead to poor overall accuracy. Therefore, we seek a new method that will imply less noise, and so give a better overall approximation when the noise is added. Thus, we propose a greedy algorithm that makes fewer probes to the data, by extending the Chow-Liu approach to higher degrees, described in Algorithm 2.

In the beginning of the algorithm (Line 1), we initialize the Bayesian network $\mathcal{N}$ to an empty list of AP pairs. Let V be a set that contains all attributes whose parent sets have been fixed in the partial construction of $\mathcal{N}$. As a next step, the algorithm randomly selects an attribute (denoted as X_1) from $\mathcal{A}$, and sets its parent set Π_1 to $\emptyset$ (Line 2). The rest of the algorithm consists of $d - 1$ iterations (Lines 3–7), in each of which we greedily add into $\mathcal{N}$ an AP pair with a large mutual information. Specifically, the AP pair in each iteration is selected from a candidate set Ω that contains every AP pair (X, Π) satisfying two requirements:

1. $|\Pi| \leq k$, which ensures that $\mathcal{N}$ is a k-degree Bayesian network. This is ensured by choosing Π only from $\binom{V}{k}$, where $\binom{V}{k}$ denotes the set of all subsets of V with size $\min(k, |V|)$ (Lines 5 and 6).

2. $\mathcal{N}$ contains no edge from X_i to X_j for any $j < i$, which guarantees that $\mathcal{N}$ is a DAG. We ensure this condition by requiring that in the beginning of any iteration, V only contains the attributes whose parent sets have been decided in the previous iterations (Line 7). In other words, the parent set of X_i can only be a subset of $\{X_1, X_2, \ldots, X_{i-1}\}$, as a consequence of which $\mathcal{N}$ cannot contain any edge from X_i to X_j for any $j < i$.

Once the parent set of each attribute is decided, the algorithm terminates and returns the Bayesian network $\mathcal{N}$ (Line 8). The number of pairs considered in iteration i is $(d-i)\binom{i}{k}$, so summing over all iterations the cost is bounded by $d\sum_{i=1}^{d}\binom{i}{k} = d\binom{d+1}{k+1}$. This determines the asymptotic cost of the procedure. Note that when $k = 1$, the above algorithm is equivalent to Chow and Liu's method [12] for constructing optimal 1-degree Bayesian networks.

4.2 A First-Cut Solution

Observe that in Algorithm 2, there is only one place where we interact directly with the input dataset D, namely, the greedy selection of an AP pair (X_i, Π_i) in each iteration of the algorithm (Line 6). Therefore, if we are to make Algorithm 2 differentially private, we only need to replace Line 6 of Algorithm 2 with a procedure that selects (X_i, Π_i) from Ω in a private manner. Such a procedure can be implemented with the exponential mechanism outlined in Section 2.1, using the mutual information function I as the score function. Specifically, we first inspect each AP pair $(X, \Pi) \in \Omega$, and calculate the mutual information $I(X, \Pi)$ between X and Π. After that, we sample an AP pair from Ω, such that the sampling probability of any pair (X, Π) is proportional to $\exp(I(X, \Pi)/2\Delta)$, where Δ is a scaling factor.

The value of Δ is set as follows. As mentioned in Section 3, PrivBayes requires that the construction of the Bayesian network $\mathcal{N}$ should satisfy ε_1-differential privacy. Accordingly, we set $\Delta = (d-1)S(I)/\varepsilon_1$, where $S(I)$ denotes the sensitivity of the mutual information function I (see Equation (3)). This ensures that each invocation of the exponential mechanism satisfies $(\varepsilon_1/(d-1))$-differential privacy. Given the composability property of differential privacy [18] and the fact that we only invoke the exponential mechanism $d-1$ times during the construction of $\mathcal{N}$, it can be verified that the overall process of constructing $\mathcal{N}$ is ε_1-differentially private.

Last, we calculate the sensitivity, $S(I)$.

Lemma 4.1.

$$S(I(X,\Pi)) = \begin{cases} \frac{1}{n}\log(n) + \frac{n-1}{n}\log\left(\frac{n}{n-1}\right), & \text{if } X \text{ or } \Pi \text{ is binary;} \\ \frac{2}{n}\log\left(\frac{n+1}{2}\right) + \frac{n-1}{n}\log\left(\frac{n+1}{n-1}\right), & \text{otherwise,} \end{cases}$$

where n is the number of tuples in D.

This can be shown by considering the maximum change in mutual information based on its definition Equation (5), as the various probabilities are changed by the alteration of one tuple. We defer the full proof to the appendix for brevity, but the maximum difference in mutual information between binary variables is achieved by the following example:

$X\backslash\Pi$	0	1
0	0	0
1	0	1

$X\backslash\Pi$	0	1
0	$\frac{1}{n}$	0
1	0	$\frac{n-1}{n}$

The mutual information of the left distribution is 0, and that of the right one is $\frac{1}{n}\log(n) + \frac{n-1}{n}\log(\frac{n}{n-1})$.

4.3 An Improved Solution

The method in Section 4.2 is simple and intuitive, but may not achieve the best results: Observe that $S(I) > \frac{1}{n}\log(n)$; this can be large compared to the range of I. For example, range$(I) = 1$ for binary distributions. As a consequence, the scaling factor $\Delta = (d-1)S(I)/\varepsilon_1$ tends to be large, and so the exponential mechanism is still quite likely to sample (from Ω) an AP pair with a small mutual information. In that case, the Bayesian network $\mathcal{N}$ constructed using the exponential mechanism will offer a weak approximation of $\Pr[\mathcal{A}]$, resulting in a low-quality output from PrivBayes. To improve over this solution, we propose to avoid using I as the score function in the exponential mechanism. Instead, we define a novel function F that maps each AP pair $(X, \Pi) \in \Omega$ to a score, such that

1. F's sensitivity is small (with respect to the range of F).
2. If $F(X, \Pi)$ is large, then $I(X, \Pi)$ tends to be large.

The rationale is that since $S(F)$ is small with respect to range(F), the scaling factor $\Delta = (d-1)S(F)/\varepsilon_1$ will also be small, and hence, the exponential mechanism has a high probability to select an AP pair (X, Π) with a large $F(X, \Pi)$. In turn, such an AP pair tends to have a large mutual information between X and Π, which helps improve the quality of the Bayesian network $\mathcal{N}$.

In what follows, we will clarify our construction of F. To achieve property 2 above, we set F to its maximum value (i.e., 0) when I is greatest. To achieve property 1, we make $F(X, \Pi)$ decrease linearly in proportion to the L_1 distance from $\Pr[X, \Pi]$ to a distribution that maximizes F, since linear functions ensure that the sensitivity is controlled: the function does not change sharply anywhere in its domain. We first introduce the concept of *maximum joint distribution*, which will be used to define the peaks of F, and then characterize such distributions:

Definition 4.2 (Maximum Joint Distribution). Given an AP pair (X, Π), a maximum joint distribution $\Pr^\diamond[X, \Pi]$ for X and Π is one that maximizes the mutual information between X and Π.

Lemma 4.3. *Assume that* $|\mathrm{dom}(X)| \le |\mathrm{dom}(\Pi)|$. *A distribution* $\Pr^\diamond[X, \Pi]$ *is a maximum joint distribution if and only if*

(1) $\Pr^\diamond[X = x] = 1/|\mathrm{dom}(X)|$, *for any* $x \in \mathrm{dom}(X)$;
(2) For any $\pi \in \mathrm{dom}(\Pi)$, *there is at most one* $x \in \mathrm{dom}(X)$ *with* $\Pr^\diamond[X = x, \Pi = \pi] > 0$.

Proofs in this section are deferred to the appendix. We illustrate Definition 4.2 and Lemma 4.3 with an example:

Example 4.4. Consider a binary variable X with $\mathrm{dom}(X) = \{0, 1\}$ and a variable Π with $\mathrm{dom}(\Pi) = \{a, b, c\}$. Consider two joint distributions between X and Π as follows:

.

$X\backslash\Pi$	a	b	c
0	0.5	0	0
1	0	0.5	0

$X\backslash\Pi$	a	b	c
0	0	0.2	0.3
1	0.5	0	0

By Lemma 4.3, both of the above distributions are maximum joint distributions, with $I(X, \Pi) = 1$.

Let (X, Π) be an AP pair, and $\mathcal{P}^\diamond[X, \Pi]$ be the set of all maximum joint distributions for X and Π. Our score function F (for evaluating the quality of (X, Π)) is defined as

$$F(X, \Pi) = -\frac{1}{2} \min_{\Pr^\diamond \in \mathcal{P}^\diamond} \left\| \Pr[X, \Pi] - \Pr^\diamond[X, \Pi] \right\|_1. \tag{7}$$

Table 3. An Example of Joint Distributions

$X\backslash\Pi$	00	01	10	11
0	0.6	0	0	0
1	0.1	0.1	0.1	0.1

(a) input joint distribution

$X\backslash\Pi$	00	01	10	11
0	0.5	0	0	0
1	0	0.3	0.1	0.1

(b) maximum joint distribution

If $F(X, \Pi)$ is large, then $\Pr[X, \Pi]$ must have a small L_1 distance to one of the maximum joint distributions in $\mathcal{P}^\diamond[X, \Pi]$, and vice-versa. In turn, if $\Pr[X, \Pi]$ is close to a maximum joint distribution in $\mathcal{P}^\diamond[X, \Pi]$, then intuitively, $\Pr[X, \Pi]$ is likely to give a large mutual information between X and Π. In other words, the value of $F(X, \Pi)$ tends to be positively correlated with $I(X, \Pi)$. This explains why F could be a good score function to replace I. In addition, F has a much smaller sensitivity than I, as shown in the following theorem:

Theorem 4.5. $S(F) = 1/n$.

This follows immediately from considering the L_1 distance between neighboring distributions. Observe that $S(F) < S(I)/\log n$, where n is the number of tuples in the input data. Meanwhile, the ranges of F and I are comparable; for example, $\text{range}(I) = 1$ and $\text{range}(F) = 0.5$ for binary domains. Therefore, when n is large (as is often the case), the sensitivity-to-range ratio of F is significantly smaller than that of I, which makes F a favorable score function over I for selecting AP pairs in the Bayesian network $\mathcal{N}$.

4.4 Computation of F

While (7) defines the function F, it still remains unclear how we can calculate $F(X, \Pi)$ given $\Pr[X, \Pi]$. In this subsection, we use dynamic programming to solve the problem for the case when all attributes in $\Pi \cup \{X\}$ have binary domains; we address the case of non-binary domains in Section 5.

Let (X, Π) be an AP pair where $|\Pi| = k$. Then, the joint distribution $\Pr[X, \Pi]$ can be represented by a 2×2^k matrix where the sum of all elements is 1. For example, Table 3(a) illustrates a joint distribution $\Pr[X, \Pi]$ with $|\Pi| = 2$. To compute $F(X, \Pi)$, we need to identify the minimum L_1 distance between $\Pr[X, \Pi]$ and a maximum joint distribution $\Pr^\diamond[X, \Pi] \in \mathcal{P}^\diamond[X, \Pi]$. Table 3(b) illustrates one such maximum joint distribution, whose L_1 distance to the distribution in Table 3(a) equals 0.4. To derive the minimum L_1 distance, a naive approach is to enumerate all maximum joint distributions in $\mathcal{P}^\diamond[X, \Pi]$; however, since $\mathcal{P}^\diamond[X, \Pi]$ may contain an infinite number of maximum joint distributions, a brute-force enumeration of $\mathcal{P}^\diamond$ is infeasible. To address this issue, we will first introduce an exponential-time algorithm for computing $F(X, \Pi)$, which will serve as the basis of our dynamic programming solution.

The basic idea of our exponential-time algorithm is to (i) partition the distributions in $\mathcal{P}^\diamond$ into a finite number of equivalence classes, and then (ii) compute $F(X, \Pi)$ by processing each equivalence class individually. By Lemma 4.3, any maximum joint distribution $\Pr^\diamond[X, \Pi]$ has the following property: for any $\pi \in \text{dom}(\Pi)$, either $\Pr^\diamond[X = 0, \Pi = \pi] = 0$ or $\Pr^\diamond[X = 1, \Pi = \pi] = 0$. In other words, for each column in the matrix representation of $\Pr^\diamond[X, \Pi]$ (where a column corresponds to a value in $\text{dom}(\Pi)$), there should be at most one non-zero entry. For example, the gray cells in Table 3(b) indicate the positions of non-zeros in the given maximum joint distribution.

For any two distributions in $\mathcal{P}^\diamond$, we say that they are *equivalent* if (i) their matrix representations have the same number of non-zero entries, and (ii) the positions of the non-zero entries are the same in the two matrices. Suppose that we divide the distributions in $\mathcal{P}^\diamond$ into equivalence

classes, each of which contains a maximal subset of equivalent distributions. Then, there are $O(3^{2^k})$ equivalent classes. As we show below, one can easily calculate the minimum L_1 distance from $\Pr[X, \Pi]$ to each equivalence class.

To explain, consider a particular equivalence class E. Let Z^- be the set of pairs (x, π), such that $\Pr^\diamond[X = x, \Pi = \pi] = 0$ for any $\Pr^\diamond[X, \Pi] \in E$. That is, Z^- captures the positions of all zero entries in the matrix representation of $\Pr^\diamond[X = x, \Pi = \pi]$. Similarly, we define the sets of non-zero entries in row $X = 0$ and $X = 1$ as

$$Z_0^+ = \{(0, \pi) \mid \Pr^\diamond[X = 0, \Pi = \pi] > 0\} \text{ and } Z_1^+ = \{(1, \pi) \mid \Pr^\diamond[X = 1, \Pi = \pi] > 0\}. \tag{8}$$

For convenience, we also abuse notation and define

$$\Pr[Z^-] = \sum\nolimits_{(x,\pi)\in Z^-} \Pr[X = x, \Pi = \pi],$$
$$\Pr[Z_0^+] = \sum\nolimits_{(x,\pi)\in Z_0^+} \Pr[X = x, \Pi = \pi],$$
$$\Pr[Z_1^+] = \sum\nolimits_{(x,\pi)\in Z_1^+} \Pr[X = x, \Pi = \pi].$$

By Lemma 4.3, we have $\Pr^\diamond[Z^-] = 0$, $\Pr^\diamond[Z_0^+] = 1/2$, and $\Pr^\diamond[Z_1^+] = 1/2$ for any $\Pr^\diamond[X, \Pi] \in E$. Then, for any $\Pr[X, \Pi]$, its L_1 distance to a distribution $\Pr^\diamond[X, \Pi] \in E$ is bounded by

$$\left\| \Pr[X, \Pi] - \Pr^\diamond[X, \Pi] \right\|_1 \geq \Pr[Z^-] + \left| \Pr[Z_0^+] - \frac{1}{2} \right| + \left| \Pr[Z_1^+] - \frac{1}{2} \right|.$$

To simplify the expressions, we make use of the notation $(x)_+$ to denote $\max(0, x)$. Given that $\Pr[Z^-] + \Pr[Z_0^+] + \Pr[Z_1^+] = 1$, the above inequality can be simplified to

$$\left\| \Pr[X, \Pi] - \Pr^\diamond[X, \Pi] \right\|_1 \geq 2 \cdot \left[\left(\frac{1}{2} - \Pr[Z_0^+] \right)_+ + \left(\frac{1}{2} - \Pr[Z_1^+] \right)_+ \right]. \tag{9}$$

Furthermore, there always exists a $\Pr^\diamond[X, \Pi] \in E$ that makes the equality hold. In other words, once the positions of the non-zero entries in $\Pr^\diamond[X, \Pi]$ are fixed, we can use Equation (9) to derive the minimum L_1 distance from any $\Pr[X, \Pi]$ to E, with a linear scan of the entries in the matrix representation of $\Pr[X, \Pi]$. By enumerating all $O(3^{2^k})$ equivalence classes of $\mathcal{P}^\diamond$, we can then derive $F(X, \Pi)$.

The above procedure for calculating F is impractical when $k \geq 4$, as the exhaustive search over all possible equivalence classes of $\mathcal{P}^\diamond$ is prohibitive. To tackle this problem, we propose a dynamic-programming-based optimization that reduces computation costs by taking advantage of the fact that the distributions are induced by n items.

Based on Equation (9), our target is to find a combination of Z_0^+ and Z_1^+ (which therefore determine Z^-) that minimizes

$$\left(\frac{1}{2} - \Pr[Z_0^+] \right)_+ + \left(\frac{1}{2} - \Pr[Z_1^+] \right)_+.$$

We define the probability mass associated with Z_0^+ and Z_1^+ as K_0 and K_1, respectively. Initially, $K_0 = K_1 = 0$. For each $\pi \in \mathrm{dom}(\Pi)$, we can either increase K_0 by $\Pr[X = 0, \Pi = \pi]$ (by assigning $(0, \pi)$ to Z_0^+) or increase K_1 by $\Pr[X = 1, \Pi = \pi]$ (by assigning $(1, \pi)$ to Z_1^+). We index $\pi \in \mathrm{dom}(\Pi)$ as $\pi_1, \pi_2, \ldots, \pi_{2^k}$. We use $C(i, a, b)$ to indicate if $K_0 = a/n$ and $K_1 = b/n$ is reachable by using the first i π's, that is, $\pi_1, \pi_2, \ldots, \pi_i$. It can be verified that (i) $C(i, a, b) = \text{true}$ if $i = a = b = 0$, (ii) $C(i, a, b) = \text{false}$ if $i < 0$ or $a < 0$ or $b < 0$, and (iii) otherwise,

$$C(i, a, b) = C(i - 1, a - n \Pr[X = 0, \Pi = \pi_i], b) \vee C(i - 1, a, b - n \Pr[X = 1, \Pi = \pi_i]). \tag{10}$$

Given an input dataset D with n tuples, each cell in $\Pr[X, \Pi]$ must be a multiple of $1/n$. Thus, we only consider the case when a and b are integers in the range $[0, n]$. Thus, the total number

of states $C(i, a, b)$ is $n^2 2^k$. A direct traversal of all states takes $O(n^2 2^k)$ time. To reduce this time complexity, we introduce the following concept:

Definition 4.6 (Dominated State). A state $C(i, a_1, b_1)$ is *dominated* by $C(i, a_2, b_2)$ if and only if $a_1 \leq a_2$ and $b_1 \leq b_2$.

Note that a dominated state can always be ignored without affecting the correctness of final result. Consequently, we maintain the set of at most n non-dominated reachable states for each $i \in [1, 2^k]$. The function F can be calculated by

$$F(X, \Pi) = - \min_{C(2^k, a, b)=\text{true}} \left(\frac{1}{2} - \frac{a}{n}\right)_+ + \left(\frac{1}{2} - \frac{b}{n}\right)_+.$$

As such, the total number of states that need to be traversed is $n2^k$, and thus the complexity of the algorithm is reduced to $O(n2^k)$. Note that k is small in practice, since we only need to consider low-degree Bayesian networks. If we treat k as a (small) constant, then the running time is linear in n.

4.5 Choice of k and θ-Usefulness

We have discussed how to build a k-degree Bayesian network under differential privacy, where k is considered as a given input to the algorithm. However, k is usually unknown in real applications and should be chosen carefully. The choice of k is non-trivial for PrivBayes. Intuitively, a Bayesian network with a larger k keeps more information from the full dimensional distribution $\Pr[\mathcal{A}]$, for example, a $(d-1)$-degree Bayesian network approximates $\Pr[\mathcal{A}]$ perfectly without having any information loss. On the other hand, the downside of using large k is that it forces PrivBayes to anonymize a set of high-dimensional marginal distributions in the second phase, which are very vulnerable to noise due to their domains of large size. These noisy distributions are less useful after anonymization especially when the privacy budget ε is small, leading to a synthetic database full of random perturbation. With very small values of ϵ, the best choice may be to pick $k = 0$, that is, to model all attributes as independent. Hence, the choice of k should balance the informativeness of a Bayesian network and the robustness of marginal distributions. This balancing act is affected by three parameters: the total privacy budget ε, the total number of tuples n in database, and usefulness θ of each noisy marginal distribution in the second phase. We quantify this in the following definition.

Definition 4.7 (θ-usefulness). A noisy distribution is θ-useful if the ratio of average scale of information to average scale of noise is no less than θ.

Lemma 4.8. *The noisy distributions in Algorithm 1 are* $(\frac{n \cdot \varepsilon_2}{(d-k) \cdot 2^{k+2}})$*-useful.*

Proof. In Algorithm 1, where k is a fixed parameter, each marginal distribution is $(k+1)$-dimensional with a total domain size 2^{k+1}. Therefore, the average scale of information in each cell is $1/2^{k+1}$ (making a simplifying uniformity assumption for this calculation).

For the scale of noise, we have $(d-k)$ marginal distributions to be anonymized and each of them consumes an equal fraction of the privacy budget for this task, that is, $\varepsilon_2/(d-k)$. The sensitivity of each marginal distribution is $2/n$, that is, (working with probabilities) the maximum contribution of changing any individual's information is to add $1/n$ to one probability corresponding to their new values and to subtract $1/n$ from their old probability. When using the Laplace mechanism, the Laplace noise η injected to each cell is drawn from distribution $\text{Lap}\,(2(d-k)/n\varepsilon_2)$ where the average scale of noise is $\mathrm{E}(|\eta|) = 2(d-k)/n\varepsilon_2$. As a consequence, we obtain that the ratio of information to noise behaves as $(1/2^{k+1})/(2(d-k)/n\varepsilon_2) = n\varepsilon_2/(d-k)2^{k+2}$ as claimed. □

The notion of θ-usefulness provides a more intuitive way to choose k automatically without closely studying the specific instance of the input database. Generally speaking, we believe a 0.5-useful noisy distribution is not good, because the scale of noise is twice that of information, while a 5-useful one is more reliable due to its large information to noise ratio. In practice, we set up a threshold θ, then choose the largest positive integer k that guarantees θ-usefulness in distribution learning (note, this is independent of the data, as it depends only on the non-private values ϵ, θ, n and d). If such a k does not exist, then k is set to the minimum value, 0. In the experimental section, we will show that the performance of PrivBayes is not sensitive to the choice of θ; therefore, we have a wide range of values to select from.

5 EXTENSIONS TO GENERAL DOMAINS

The dynamic programming approach in Section 4.4 and the notion of θ-usefulness in Section 4.5 both assume that all attributes in the input data D are binary. In this section, we extend our solution to the case when D contains non-binary attributes.

5.1 Encodings of Non-binary Attributes

First, we study how to represent non-binary attributes in PrivBayes. We present four different approaches, starting with the application of a standard encoding idea.

Binary Encoding. Following the common practice in the literature [47], our first approach to handling non-binary attributes is to convert each of them into a set of binary attributes. In particular, for each categorical attribute X whose domain size equals ℓ, we first encode each value in X's domain into a binary representation with $\lceil \log \ell \rceil$ bits; after that, we convert X into $\lceil \log \ell \rceil$ binary attributes $X_1, X_2, \ldots, X_{\lceil \log \ell \rceil}$, such that X_i corresponds to the ith bit in the binary representation. Meanwhile, for each continuous attribute Y, we first discretize the domain of Y into a fixed number b of equi-width bins,[3] and then convert Y into $\lceil \log b \rceil$ binary attributes, using a similar approach to the transformation of X. Figures 2 and 3 give examples of binary encoding on continuous and categorical attributes, respectively. Figure 2 shows how we might encode the attribute "age." First, it is broken into $b = 8$ (for the purpose of illustration) ranges of ten years each. Then the index for each range is represented as three binary bits (i.e., $\log_2 8$), 000_2 for the first, 111_2 for the last. For the categorical attribute in Figure 3, any linearization of the attribute values to order them can be used before the binary encoding is applied. After the transformation, D can be encoded to form a new database D_b in the binary domain. Then, we apply PrivBayes on D_b to generate a synthetic dataset D_b^*, then decode it to get D^* in the original domain.

The strength of this approach is that it allows the direct use of θ-usefulness and the advanced score function F. Moreover, the binary decomposition of attributes provides a high level of flexibility in constructing Bayesian networks. For instance, assume that the value of a binary attribute "is retired" is true, if and only if the value of "age" (as in Figure 2) is larger than 60. To fully preserve this correlation, PrivBayes with binary encoding requires to use the binary attribute "is retired" and merely the first two bits of "age." The reason is that the first two bits actually discretize the domain of "age" into four bins, that is, (0, 20], (20, 40], (40, 60], and (60, 80], which is sufficient for the purpose. Data utility is then improved, as the corresponding joint distribution of "is retired" and "age" contains only 8 cells, instead of 16 if all bits are included; thus, it is more robust against noise.

Although the binary encoding provides high flexibility, it comes at the cost of some redundancy. The semantics of the new binary attributes in isolation is not always very clear. For example, the second bit of "age" in Figure 2 represents whether a person's age is in $(20, 40] \cup (60, 80]$. This alone

[3]We use $b = 16$ in experiments and $b = 8$ in the example in Figure 2 for simplicity of presentation.

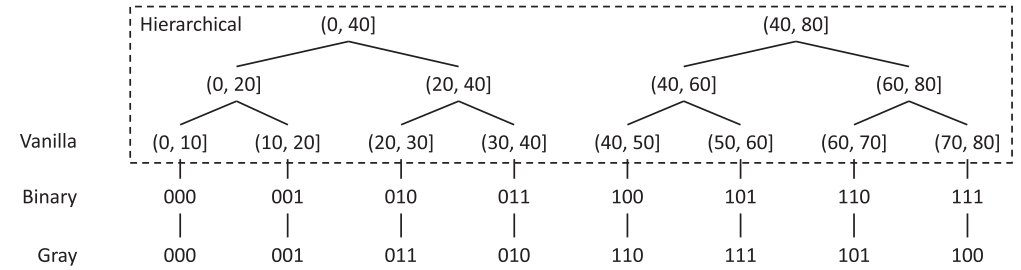


Fig. 2. Four encodings on a continuous attribute "age."

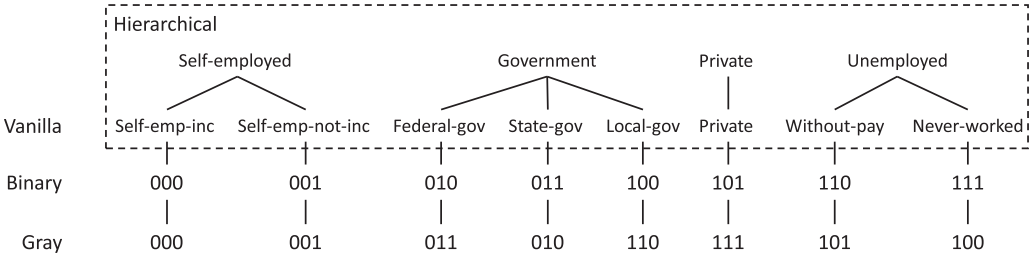


Fig. 3. Four encodings on a categorical attribute "workclass."

does not make too much sense, until the most significant bit is taken into account as well. As for the categorical attribute "workclass" (in Figure 3), each bit alone eventually splits all values in the domain into two subsets. But there is no semantics behind those partitions, unless all three bits (and partitions) are combined together. In the construction of a Bayesian network, however, all these bits are treated as natural attributes of independent interest, leading to the consideration of a large number of redundant AP pairs with artificial meanings. This would hurt the quality of Bayesian networks, if any redundant AP pair is selected, which may often be the case when ε is small.

Gray Encoding. Our second approach is a variant of binary encoding, namely, Gray encoding [24]. It is a binary encoding system where two successive values differ in only one bit. See examples in Figures 2 and 3: the same set of binary identifiers are generated, but in a different order compared to the sequence of the attribute values. When combined with PrivBayes, it inherits some of the pros and cons of the natural binary encoding, but potentially brings some advantages. As successive values share most bits, Gray encoding can be more robust to noise. Take the value $(30, 40]$ of attribute "age" as an example (Figure 2). If the perturbation in PrivBayes happens to change the first or the last bit of its code 010_2, then the noisy code (i.e., 110_2 or 011_2) will correspond to a value adjacent to the original one. Thus, the distortion of information is reduced. In contrast, the natural binary encoding does not have this property. Therefore, Gray encoding is a competitive alternative to the natural binary encoding.

Vanilla Encoding. To circumvent the redundancy problem of binary encodings, we propose another approach that keeps all attributes intact during the construction of Bayesian networks. So, for an attribute that takes on ℓ different values, instead of trying to encode it with $\lceil \log_2 \ell \rceil$ bits, we directly represent it as a discrete variable with ℓ possible values. Figures 2 and 3 present two examples of vanilla encoding. Under vanilla encoding, the domain of each attribute is considered to be indivisible. Intuitively, this preserves the semantics of non-binary attributes and avoids generating

extra attributes with artificial meaning, which help reduce the uncertainty in learning Bayesian networks.

However, applying vanilla encoding on PrivBayes turns out to be challenging. First, we need to revise our notion of θ-usefulness principle and modify the definition of score function F to attributes with general domains. In the following Sections 5.2 and 5.3, we will clarify how each component of PrivBayes can be updated or redesigned to fit in with non-binary attributes. These effects make the use of non-binary encoding possible within the framework of PrivBayes. Another issue of vanilla encoding is lack of flexibility: attribute inclusion is an "all or nothing" decision. As the domain of each attribute can not be partially encoded, preserving a correlation with a high cardinality can be costly. Recall the "is retired" example in binary encoding. PrivBayes with vanilla encoding has to build a joint distribution of full size 16, because all values of "age" must be present. Due to the θ-usefulness criterion, PrivBayes may choose to discard this distribution of large size when the privacy budget ε is small.

Hierarchical Encoding. Last, we present hierarchical encoding, a flexible encoding scheme aware of the semantics of attributes. The main idea is to generalize the domain of each attribute using a taxonomy tree. In Figures 2 and 3, we illustrate taxonomy trees (roots are omitted) for continuous and categorical attributes, respectively. To each continuous attribute whose domain is partitioned into b bins, we build a binary tree of height $\lceil \log b \rceil$ (excluding the root), where each intermediate node represents the concatenation of bins in its leaves. For each categorical attribute, we require a specific taxonomy tree based on domain knowledge. The tree is built based only on knowledge of the relevant domain; in many cases, there are natural hierarchies based on organizations (e.g., city, region, country, continent) that are inherent to the attribute, or can be easily specified. It is common in prior work on data anonymization to assume the existence of such a hierarchy, for example, efforts on k-anonymization [3, 28]. The leaf nodes describe all the possible specific values in the domain, which are then recursively generalized at each subsequent level of the tree. For concreteness, take attribute "workclass" in Figure 3 as an example. Working for federal, state, or local government are three values in the original domain, which can be generalized to working for government at the upper level of the taxonomy tree. Similarly, the attribute "country" can be generalized to geographical regions, then to continents, according to the CIA World Factbook [13]. It is also worth mentioning that vanilla encoding can be seen as a special case of hierarchical encoding, where each taxonomy tree consists of leaf nodes only.

To apply hierarchical encoding on PrivBayes, we first formalize the concept of generalized attribute. Let height(X) denote the height of attribute X's taxonomy tree. For each integer $i \in [0, \text{height}(X))$, the nodes at level i (leaves at level 0) of the taxonomy tree define the domain of a generalized version of X, denoted as $X^{(i)}$. The larger i is, the more generalized the attribute will be. Note that $X = X^{(0)}$. With hierarchical encoding, each attribute can provide multiple choices in the construction of Bayesian networks. The tradeoff of attribute generalization is that a less generalized attribute is more informative, yet the more generalized one is more flexible due to its domain of small size. To balance this tradeoff, we resort to the θ-usefulness principle. Following the intuitions in Section 4.5, we always prefer the less generalized attributes under the condition that θ-usefulness is guaranteed. In the next section, we will specify how to integrate θ-usefulness with the hierarchical encoding. To avoid a combinatorial explosion of possibilities, we restrict to considering generalizations where we pick a single level i to apply across one attribute, that is, we do not allow generalizations where different branches of the generalization tree are represented at different levels. Removing this assumption is feasible within our framework, but entails a much larger set of possibilities to search, with a comensurate drain on the privacy budget. Consequently, we choose to make this restriction.

5.2 Extension of θ-usefulness

In this subsection, we extend the notion of θ-usefulness to the non-binary encodings, that is, vanilla and hierarchical encodings. This requires us to handle non-binary attributes (both encodings) and generalized attributes (hierarchical encoding only). For easy understanding, we will first present how each component of PrivBayes can be updated to fit in with non-binary attributes, under the criterion of θ-usefulness; the algorithms for generalized attributes will be elaborated at the end of this subsection.

Non-binary Attributes. Recall that, in the proof of Lemma 4.8, all $(k+1)$-dimensional marginals have the same domain size 2^{k+1}, such that we can bound the average scale of information in each cell by simply limiting k. However, that is no longer the case when the domain sizes of attributes vary. In particular, marginal distributions of the same dimensionality may have very different sizes; for example, the joint distribution of two binary attributes has 4 cells, while that of two attributes with cardinality four each has 16. Therefore, a single k is not sufficient to guarantee θ-usefulness anymore: we should also take the domain size of each attribute into account. Toward this end, we revise Algorithms 1 and 2 as follows.

ALGORITHM 3: NoisyConditionals $(D, \mathcal{N})$: returns $\mathcal{P}^*$

initialize $\mathcal{P}^* = \emptyset$;
for $i = 1$ **to** d **do**
  materialize the joint distribution $\Pr[X_i, \Pi_i]$;
  generate differentially private $\Pr^*[X_i, \Pi_i]$ by adding noise $\mathrm{Lap}(\frac{2d}{n\varepsilon_2})$;
  set negative values in $\Pr^*[X_i, \Pi_i]$ to 0 and normalize;
  derive $\Pr^*[X_i \mid \Pi_i]$ from $\Pr^*[X_i, \Pi_i]$; add it to $\mathcal{P}^*$;
return $\mathcal{P}^*$;

ALGORITHM 4: GreedyBayes (D, θ): returns $\mathcal{N}$

initialize $\mathcal{N} = \emptyset$ and $V = \emptyset$;
randomly select an attribute X_1 from $\mathcal{A}$; add $(X_1, \emptyset)$ to $\mathcal{N}$; add X_1 to V;
for $i = 2$ **to** d **do**
  initialize $\Omega = \emptyset$;
  foreach $X \in \mathcal{A} \backslash V$ **do**
    find all maximal parent sets of X, that is,
    $\top(X) = \texttt{MaximalParentSets}(V, \frac{n\varepsilon_2}{2d\theta|\mathrm{dom}(X)|})$;
    if $\top(X) = \emptyset$ **then**
      add $(X, \emptyset)$ to Ω;
    else
      foreach $\Pi \in \top(X)$ **do** add (X, Π) to Ω;
  select (X_i, Π_i) from Ω, using the exponential mechanism with privacy budget $\varepsilon_1/(d-1)$;
  add (X_i, Π_i) to $\mathcal{N}$; add X_i to V;
return $\mathcal{N}$;

ALGORITHM 5: MaximalParentSets (V, τ): returns $\mathcal{S}$

```
1 if τ < 1 then return ∅;
2 if V = ∅ then return {∅};
3 pick an arbitrary attribute X from V;
4 initialize S = MaximalParentSets(V \ {X}, τ);
5 foreach Z ∈ MaximalParentSets(V \ {X}, τ / |dom(X)|) do
6     if Z ∈ S then remove Z from S;
7     add Z ∪ {X} to S;
8 return S;
```

First, in Algorithm 1, we set $k = 0$, that is, we materialize all d marginal distributions associated with the Bayesian network $\mathcal{N}$. The privacy budget of this process (i.e., ε_2) is evenly divided into d portions, each of which is spent on a marginal. In other words, the Laplace noise added to each cell of each marginal is $\mathrm{Lap}(\frac{2d}{n\varepsilon_2})$. Algorithm 3 presents the revised version of Algorithm 1, with the places that have changed underlined to show the difference.

Now consider Algorithm 2. Observe that, in Algorithm 2, Line 5 is the only place where we interact directly with k, and it generates a candidate set Ω, which contains all eligible AP pairs for the next round of selection. Specifically, for each attribute $X \in \mathcal{A} \backslash V$, the intention of Line 5 is to help identify every subset Π of V that satisfies the following two requirements:

(i) $\Pr[X, \Pi]$ is θ-useful. The motivation of this requirement is to ensure that the information in $\Pr[X, \Pi]$ will not be overwhelmed by the noise introduced in the distribution learning phase, as we explain in Section 4.5;

(ii) Π is maximal, that is, there is no subset Π' of V such that $\Pr[X, \Pi']$ is θ-useful and $\Pi \subset \Pi'$. The requirement of maximality of Π relies on the monotonicity of the mutual information I. Given two sets Π and Π' such that $\Pi \subseteq \Pi'$, we always have $I(X, \Pi) \leq I(X, \Pi')$ for any attribute X. Therefore, with respect to the informativeness of the Bayesian network, a larger Π is always preferred.

We refer to a subset Π satisfying the above two conditions as a *maximal parent set* of X. Finding all maximal parent sets of a particular attribute is simple when all attributes are binary. In particular, with Lemma 4.8, one can easily find the appropriate value of k such that all subsets of V with size $\min(k, |V|)$ meet the requirements.[4]

However, when some attributes are non-binary, a different calculus applies. Given an AP pair (X, Π) with m cells in $\Pr[X, \Pi]$, the average scale of information in each cell is m^{-1}. On the other hand, the average scale of noise is $2d/n\varepsilon_2$ with our updates in Algorithm 3. Therefore, $\Pr[X, \Pi]$ is θ-useful only if $m \leq n\varepsilon_2/2d\theta$. In other words, given an attribute X, we are only interested in those maximal subsets of V whose domain sizes are no greater than $\frac{n\varepsilon_2}{2d\theta|\mathrm{dom}(X)|}$.

Based on the above discussion, we present Algorithm 4, which is a revised version of Algorithm 2. The initialization and outer loop stay the same, but compared to Algorithm 2, it adopts a new procedure to generate the candidate set Ω (Lines 5–10). Specifically, for each attribute $X \in \mathcal{A} \backslash V$, we first invoke the MaximalParentSets algorithm (Line 6), which identifies a set $\top(X)$ that contains all maximal parent sets of X. (We will elaborate MaximalParentSets shortly.) After constructing $\top(X)$, we add AP pair (X, Π) to Ω for each $\Pi \in \top(X)$ (Line 10). In Lines 7 and 8,

[4]When $k > |V|$, the only maximal parent set is V itself.

ALGORITHM 6: MaximalParentSets* (V, τ): returns $\mathcal{S}$

```
1  if τ < 1 then return ∅;
2  if V = ∅ then return {∅};
3  pick an arbitrary attribute X from V;
4  initialize S = ∅, U = ∅;
5  for i = 0 to height(X) − 1 do
6      foreach Z ∈ MaximalParentSets*(V \ {X}, τ / |dom(X^(i))|) do
7          if Z ∈ U then continue;
8          add Z to U; add Z ∪ {X^(i)} to S;
9  foreach Z ∈ MaximalParentSets*(V \ {X}, τ) do
10     if Z ∈ U then continue;
11     add Z to S;
12 return S;
```

we also consider an extreme case that $\top(X)$ is empty, which implies that even $\Pr[X]$ violates θ-usefulness. Following the common practice discussed in Section 4.5, we add $(X, \emptyset)$ to Ω to ensure that every attribute is modeled (at least as an independent attribute) in the Bayesian network. Once the candidate set Ω is ready to be selected from, we invoke the exponential mechanism to privately choose an AP pair, then add it to $\mathcal{N}$ (Lines 11 and 12).

We now clarify the details of the MaximalParentSets method. Algorithm 5 shows the pseudocode of MaximalParentSets. It takes as input a set of attributes V and an upper bound of domain size τ, and outputs a set $\mathcal{S}$ that contains all maximal subsets of V with domain sizes no larger than τ. The main idea of the algorithm is to recursively build two sets of maximal subsets, with and without a particular attribute $X \in V$, respectively, and then merge them to get the final result. More specifically, we first initialize $\mathcal{S}$ as the set of maximal subsets without attribute X (Line 4). To construct the ones with X, the algorithm utilizes the attribute set $V \backslash X$ and the upper bound $\tau / |\operatorname{dom}(X)|$, to ensure that adding X to each returned subset still guarantees the maximality without violating the restriction on domain sizes. In Lines 6 and 7, the algorithm updates $\mathcal{S}$ by removing the non-maximal duplications (as Z is a subset of $Z \cup \{X\}$) and then adding the maximal subsets that include the attribute X. Finally, the algorithm has two terminating conditions:

(i) $\tau < 1$. Given that the minimum size of a domain is $\operatorname{dom}(\emptyset) = 1$, this condition indicates that no subset of V meets the restriction on domain size. In that case, the algorithm simply returns the empty set;
(ii) $V = \emptyset$. Given that $\tau \geq 1$ and V is empty, $\emptyset$ is the only subset of V that satisfies the requirement on the domain size. Therefore, the algorithm returns a singleton set $\{\emptyset\}$.

Generalized Attributes. Last, we show how to utilize generalized attributes in the framework of PrivBayes. Recall that the intuition of attribute generalization is to provide high flexibility in the construction of Bayesian networks. In particular, given an AP pair (X, Π) whose joint distribution $\Pr[X, \Pi]$ violates θ-usefulness, we aim to generalize some attributes in Π to satisfy the restriction on domain size, while preserving as much information between X and Π as possible. Toward this end, we extend the definition of maximal parent sets to allow generalized attributes.

First, we define a *generalized subset* as a subset in which attributes are generalized. Note that the original attribute can be seen as a generalized attribute of itself. Then, for each attribute $X \in \mathcal{A} \backslash V$,

the maximal parent set Π is a generalized subset of V that satisfies: (i) $\Pr[X, \Pi]$ is θ-useful; (ii) Π is maximal, that is, there is no generalized subset Π' of V such that $\Pr[X, \Pi']$ is θ-useful and Π' contains any extra attribute not in Π, or any shared attribute but with a lower generalization level.

By this definition, we update the MaximalParentSets method as in Algorithm 6. In Lines 6–8, the algorithm generates maximal parent sets that contain attribute $X^{(i)}$. By utilizing the attribute set $V \backslash X$ and the upper bound $\tau / |\operatorname{dom}(X^{(i)})|$, we ensure that the union of $X^{(i)}$ and each returned subset Z satisfies the requirement on domain size. Before adding $Z \cup \{X^{(i)}\}$ to $\mathcal{S}$, we check its maximality with set $\mathcal{U}$, as $Z \in \mathcal{U}$ implies that the union of Z and a less generalized X is already included in $\mathcal{S}$; therefore, we have to discard the non-maximal $Z \cup \{X^{(i)}\}$. By enumerating generalization levels of X (Line 5), we construct all maximal parent sets with the particular attribute X. In the end, we process the remaining cases in which X is excluded (Lines 9–11).

Although the construction of maximal parent sets is complicated, applying generalized attributes to other parts of PrivBayes is surprisingly simple. No additional change is required in Algorithms 3 and 4. In both algorithms, the only place where we interact with generalized attributes is to materialize the non-private joint distribution (to inject noise and compute score function, respectively). This can be done by generalizing tuples in the input data accordingly. As for data synthesis, the properties of Bayesian networks ensure that each attribute is sampled before appearing in any parent set. Therefore, to sample X from the conditional distribution $\Pr^*[X \mid \Pi]$, all we need is to make sure that the previously sampled attributes in Π are generalized properly.

5.3 Alternative Score Function

To facilitate the above extension of θ-usefulness to non-binary encodings, there is one missing piece of the puzzle: we need to identify a score function to be used within the exponential mechanism to select AP pairs. One natural choice, as in the case of binary domains (see Section 4.2), is the mutual information function I in Equation (5). Observe that I does not have any restriction on the domain sizes of attributes; therefore, it can be directly applied on general domains. As for the more advanced score function F defined in Section 4.3, however, it is not immediately clear how we may extend F to general domains, since the dynamic programming algorithm for computing F in Section 4.4 requires that all input domains are binary. In what follows, we will prove that the extension of F is infeasible, and an alternative score function R will be proposed to work on general domains instead.

First, we observe that computing F is hard in general (proof provided in the Appendix).

Theorem 5.1. *The problem of determining whether $F(X, \Pi) = t$ is NP-hard, given a joint distribution $\Pr[X, \Pi]$ and a real number t.*

Given the above NP-hardness result, we know that there is no efficient way to calculate F. Fortunately, we have shown how a pseudo-polynomial dynamic programming solution can be designed to compute F (in Section 4.4), by utilizing a special property of PrivBayes. That is, all numbers in the joint distribution are multiples of $1/n$, which provides a possibility to represent the states of dynamic programming at a cost polynomial in n. However, the extension of this solution to general domains has time complexity $O(|\operatorname{dom}(\Pi)| \cdot n^{|\operatorname{dom}(x)|-1})$; therefore, it is not efficient for any non-binary X.

Recall that F measures the L_1 distance from the input $\Pr[X, \Pi]$ to a joint distribution that *maximizes* $I(X, \Pi)$. The rationale behind this design is twofold: (i) the L_1 distance measurement guarantees a relatively small sensitivity, that is, $S(F) = O(1/n)$, and (ii) if $\Pr[X, \Pi]$ is close to a joint distribution of maximum $I(X, \Pi)$, then intuitively, $\Pr[X, \Pi]$ is likely to give a large mutual information between X and Π. Now, we borrow the ideas from F, and propose an alternative score function R, which can be computed efficiently on non-binary domains. The main difference of R

Table 4. Properties of Score Functions

Function	Range	Sensitivity	Time complexity
$I(X, \Pi)$	$O(1)$	$O(\log n/n)$	$O(\lvert \mathrm{dom}(X)\rvert \cdot \lvert \mathrm{dom}(\Pi)\rvert)$
$F(X, \Pi)$	$O(1)$	$O(1/n)$	$O(\lvert \mathrm{dom}(\Pi)\rvert \cdot n^{\lvert \mathrm{dom}(x)\rvert - 1})$
$R(X, \Pi)$	$O(1)$	$O(1/n)$	$O(\lvert \mathrm{dom}(X)\rvert \cdot \lvert \mathrm{dom}(\Pi)\rvert)$

is that it relies on the L_1 distance from $\Pr[X, \Pi]$ to a joint distribution that *minimizes* $I(X, \Pi)$. In the following, we specify the construction of R.

Lemma 5.2. *If* $I(X, \Pi) = 0$, *then* $\Pr[X = x, \Pi = \pi] = \Pr[X = x] \cdot \Pr[\Pi = \pi]$ *for any pair of* $x \in \mathrm{dom}(X)$ *and* $\pi \in \mathrm{dom}(\Pi)$.

The proof is rather straightforward given that $I(X, \Pi) = 0$ implies mutual independence between X and Π, so is omitted here. With Lemma 5.2, one can easily generate a joint distribution, denoted as $\overline{\Pr}[X, \Pi]$, that satisfies $I(X, \Pi) = 0$. In particular, for any $x \in \mathrm{dom}(X)$, $\pi \in \mathrm{dom}(\Pi)$, we have

$$\overline{\Pr}[X = x, \Pi = \pi] = \Pr[X = x] \cdot \Pr[\Pi = \pi],$$

where $\Pr[X]$ and $\Pr[\Pi]$ are marginal distributions derived from $\Pr[X, \Pi]$. Then, our new score function to evaluating AP pair (X, Π) is defined as

$$R(X, \Pi) = \frac{1}{2} \left\| \Pr[X, \Pi] - \overline{\Pr}[X, \Pi] \right\|_1. \tag{11}$$

Intuitively, if $R(X, \Pi)$ is small, then $\Pr[X, \Pi]$ must be close to $\overline{\Pr}[X, \Pi]$; therefore, the mutual information between X and Π is likely to be small. On the other hand, a large $R(X, \Pi)$ indicates a large scale of distortion between $\Pr[X, \Pi]$ and $\overline{\Pr}[X, \Pi]$, which implies a strong correlation (and a large I) between X and Π. Note that R can be interpreted as the total variation distance to a distribution whose mutual information is zero. This can be formalized by the following inequality between R and I:

$$\begin{aligned} R(X, \Pi) &= \frac{1}{2} \left\| \Pr[X, \Pi] - \overline{\Pr}[X, \Pi] \right\|_1 \\ &\le \sqrt{\frac{\ln 2}{2} D_{KL}(\Pr[X, \Pi] \parallel \overline{\Pr}[X, \Pi])} \qquad \text{(by Pinsker's inequality)} \\ &= \sqrt{\frac{\ln 2}{2} D_{KL}(\Pr[X, \Pi] \parallel \Pr[X] \Pr[\Pi])} = \sqrt{\frac{\ln 2}{2} I(X, \Pi)}, \end{aligned}$$

where $D_{KL}(\cdot \parallel \cdot)$ is the KL-divergence defined in Equation (6). In summary, the value of $R(X, \Pi)$ tends to grow with $I(X, \Pi)$, albeit at most with a square-root relationship. Last, we prove that R has a small sensitivity.

Theorem 5.3. $S(R) \le 3/n + 2/n^2$.

Therefore, we conclude that R is a feasible score function for PrivBayes.

Last, we compare R with its predecessors I and F. Table 4 summarizes the properties of the three score functions. Among all choices, F is the most competitive one in terms of sensitivity, that is, $S(F)$ is less than 1/3 of $S(R)$ (comparing the constants in Theorems 4.5 and 5.3) and both of them are much smaller than $S(I)$. This makes F preferable to the other two functions, but only in the case of binary domains. As for non-binary attributes, R is the score function of first choice. In the experimental section, we will empirically evaluate the performance of different score functions in constructing Bayesian networks.

Table 5. Dataset Characteristics

Dataset	Cardinality	Dimensionality	Domain size
NLTCS	21,574	16	$\approx 2^{16}$
ACS	47,461	23	$\approx 2^{23}$
Adult	45,222	15	$\approx 2^{52}$
BR2000	38,000	14	$\approx 2^{32}$

6 EXPERIMENTS

6.1 Experimental Settings

Datasets. We make use of four real datasets[5] in our experiments: (i) NLTCS [35], which contains records of 21,574 individuals participated in the National Long Term Care Survey, (ii) ACS [44], 47,461 rows of personal information obtained from 2013 and 2014 ACS sample sets in IPUMS-USA, (iii) Adult [1], which includes the information of 45,222 individuals extracted from the 1994 U.S. Census, and (iv) BR2000 [44], which consists of 38,000 census records collected from Brazil in year 2000. The first two datasets contain only binary attributes, while the last two have both continuous and categorical attributes, for which we also provide taxonomy trees derived from common knowledge on country locations, work classes, school grades, marriage status and so on. Table 5 illustrates main properties of the datasets.

Tasks. We evaluate the performance of PrivBayes on two different tasks. The first task is to build all α-way marginals of a dataset [2]. For convenience, we use Q_α to denote the set of all α-way marginals. We evaluate Q_3 and Q_4 on binary datasets NLTCS and ACS, but examine Q_2 and Q_3 instead on the remaining two datasets, since each of those datasets leads to a prohibitively large number of queries in Q_4. We measure the accuracy of each noisy marginal by the *total variation distance* [16] between itself and the noise-free marginal (i.e., half of the L_1 distance between the two marginals, when both of them are treated as probability distributions). We use the average accuracy over all marginals as the final error metric for Q_α.

The second task that we consider is to *simultaneously* train multiple SVM classifiers on a dataset, where each classifier predicts one attribute in the data based on all other attributes. Specifically, on NLTCS, we construct four classifiers to predict whether a person (i) is unable to get outside, (ii) is unable to manage money, (iii) is unable to bathe, and (iv) is unable to travel, respectively. Meanwhile, on ACS, we train four classifiers to predict whether an individual (i) owns a private dwelling, (ii) has a mortgage loan, (iii) lives in a multi-generation family, and (iv) attends school, respectively. On Adult, four classifiers are trained to predict whether an individual (i) is a female, (ii) makes over 50K a year, (iii) holds a post-secondary degree, and (iv) has never married, respectively. Last, on BR2000, we train four classifiers to predict whether an individual (i) is a Catholic, (ii) owns at least one car, (iii) has at least one child, and (iv) is older than 20, respectively. For each classification task, we use 80% of the tuples in the data as the training set, and the other 20% as the testing set. We apply PrivBayes on the training data to generate a synthetic dataset, and then use the synthetic data to construct SVM classifiers. The quality of each classifier is measured by its *misclassification rate* on the testing set, that is, the fraction of tuples in the testing data that are incorrectly classified.

For each of the aforementioned tasks, we repeat each experiment on each method 100 times, and we report the average measurements in our experimental results.

[5]The PrivBayes code and datasets (with taxonomy trees) are available at https://sourceforge.net/projects/privbayes.

Baselines. For the task of answering count queries in Q_α, we compare PrivBayes with five approaches: (i) *Laplace* [19], which generates all α-way marginals of a dataset and then injects Laplace noise directly into each cell of the marginals, (ii) *Fourier* [2], which transforms the input data D into the Fourier domain, adds Laplace noise to a subset of Fourier coefficients and uses the noisy coefficients to construct α-way marginals, (iii) *MWEM* [26], which maintains an approximation to the input data that is repeatedly improved to answer the query set better, and (iv) *Contingency*, which first builds the noisy contingency table, and then projects it onto attribute subsets to compute marginals. However, *MWEM* and *Contingency* are only applicable to NLTCS and ACS, since their computational cost is proportional to the total domain size of input data (which grows exponentially in the number of attributes). We also considered several other existing approaches [17, 31, 32, 48] for answering count queries under differential privacy, but find them inapplicable due to our datasets' large total domain size, which is orders of magnitude larger than the dataset size. Last, we compare to a trivial baseline (v) *Uniform*, that simply returns a uniform distribution to any marginal query. For fair comparison, we adopt two consistency techniques sequentially to boost the accuracies of baselines: non-negativity, which rounds all negative counts in a noisy marginal to 0, then normalization, which applies non-negativity and then linearly rescales the non-negative counts in a noisy marginal to make them sum to n.

For the task of training multiple SVM classifiers, we compare PrivBayes against four methods: *PrivateERM* [8], *PrivGene* [50], *NoPrivacy*, and *Majority*. In particular, *PrivateERM* and *PrivGene* are two state-of-the-art methods for SVM classification under differential privacy. *NoPrivacy* constructs classifiers directly on the input data without any privacy protection. *Majority* is a naïve classification method under differential privacy that works as follows. Let $Y = \{0, 1\}$ be the attribute to be classified, and n be the number of tuples in the training data. *Majority* first counts the number of tuples in the training data with $Y = 1$, and then adds Laplace noise (with scale $1/\varepsilon$) into the count to ensure ε-differential privacy. If the noisy count is larger than $n/2$, then *Majority* predicts that all tuples in the testing data should have $Y = 1$; otherwise, *Majority* predicts $Y = 0$ for all tuples. For PrivBayes, *PrivGene*, and *NoPrivacy*, we adopt the standard hinge-loss C-SVM model [6] with $C = 1$; for *PrivateERM*, we adopt a slightly different SVM model with Huber loss [8], as it does not support the hinge-loss model.

6.2 Effect of Score Functions

In the first set of experiments, we evaluate the effectiveness of score functions F (in Section 4.3) and R (in Section 5.3), against the mutual information function I. We introduce an additional baseline that we dub "NoPrivacy." This function shows the sum of mutual information of the optimal k-degree Bayesian network, without any perturbation affecting the choice of model. The value of k is chosen by the same θ-usefulness criterion; hence, the behavior does vary as a function of ε: all results for a given ε are working with the same parameter k for ease of comparison. Figure 4 illustrates the performance of PrivBayes when combined with F, R, and I, respectively, as well as the best k-degree network. The performance of each method is evaluated by the sum of the mutual information of every AP pair in the Bayesian network $\mathcal{N}$, that is, $\sum_{i=1}^{d} I(X_i, \Pi_i)$. The parameter of θ-usefulness is set to 4, which is our default baseline.

The general trend that we observe in Figure 4 is that as the privacy budget ε increases, the quality (captured by the sum of mutual information of the model) increases, and this trend is followed by all the choices of target function (I, R, or F). The exact behavior varies across data sets, where in some case it asymptotes, while in others it has more of an s-shape. It appears that this behavior is mostly due to the innate ability of Bayesian networks of differing arity to capture the data distribution, as demonstrated by the NoPrivacy line, which the other lines tend to mimic. Recall

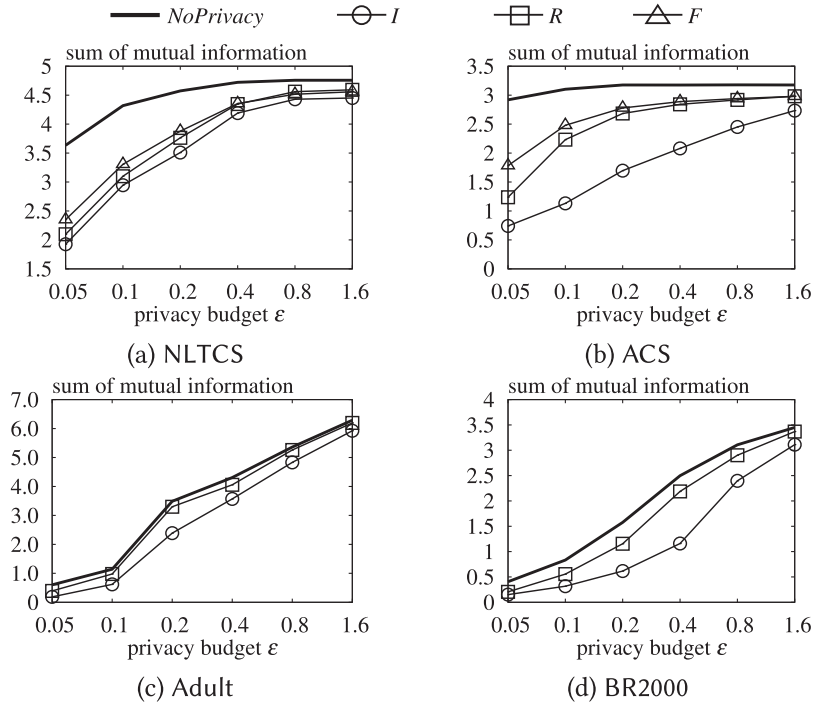
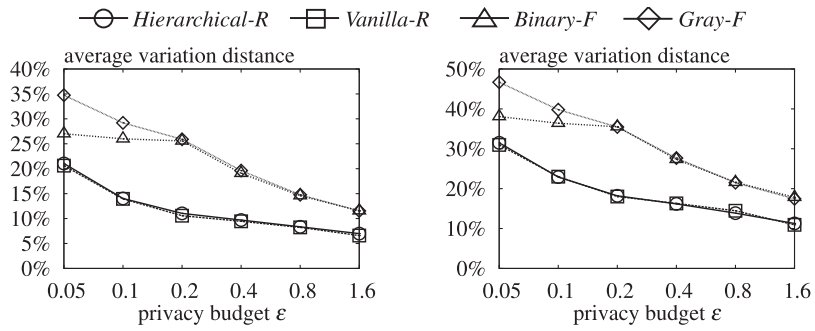


Fig. 4. Comparison among different score functions.

that this method is capturing the consequence of varying k, determined indirectly as a function of ε via θ-usefulness.

For binary datasets NLTCS and ACS, Figure 4 shows that F and R consistently outperform I in all cases. This is consistent with our analysis in Sections 4.3 and 5.3 that these advanced score functions help improve the quality of the Bayesian network constructed by PrivBayes. Compare score functions F and R. They achieve almost identical results on large ε (e.g., $\varepsilon \geq 0.4$), while F becomes preferable on small ε. The reason is that F and R share the same range (e.g., 0.5 for binary attributes), but the sensitivity of F is only 1/3 of that of R. This leads to better utility of F especially when the scale of perturbation in selecting AP pairs is large. In short, F is the best score function for PrivBayes when all attributes are binary. As for the non-private method, it performs better with larger privacy budget ε, as the θ-usefulness allows it to build higher degree Bayesian networks. However, the quality of networks converges when $\varepsilon \geq 0.4$, which provides the key insight in designing PrivBayes that a network of low degree is sufficient to approximate the data. As a consequence, we set up a strict θ-usefulness criterion to limit the network degree (see Section 6.4), as a large degree may not help much in network learning, but make marginals more vulnerable to noise. On the other hand, the performance gap between *NoPrivacy* and PrivBayes methods shrinks rapidly as ε increases. Therefore, we consider the noisy networks learnt by PrivBayes to be of high quality.

On datasets Adult and BR2000, we adopt a different setting to test the performance of score functions in non-binary cases. In particular, the vanilla encoding is applied on both continuous and categorical attributes of these datasets. As a consequence, we have to omit F, because it is hard to compute in general domains (see proof in Section 5.3). The experimental results again



(a) Adult, Q_2 (b) Adult, Q_3

Fig. 5. Comparison among different encodings (α-way marginals on Adult).

support the superiority of R to I. Therefore, we will adopt R as the score function of PrivBayes, when non-binary attributes are included. It is also worth noting that all methods behave quite differently compared to the binary cases. To explain, recall that non-binary attributes convey more information than binary ones, but are more likely to be rejected by θ-usefulness due to their large domain sizes. As ε increases, some informative relations among attributes of large domains will be included in the network, resulting in a large quality improvement. This explains why the curves are not as smooth as in binary cases.

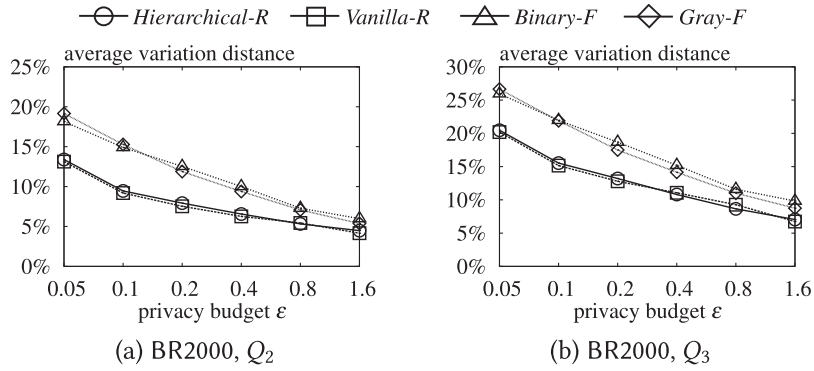
Last, regarding the efficiency of computing score functions, F is significantly more time-consuming than the other two, since it takes $O(n2^k)$ for one AP pair. Functions R and I, in contrast, only take $O(2^k)$. For small k values (i.e., $k = 0, 1, 2$), the time taken to construct a Bayesian network with F is less than 1min and is negligible. For larger values of k, the time taken is typically higher, for example, a few hours in the case of $k = 6$ on NLTCS and about 40min for $k = 4$ on ACS. Note that this does not represent a major concern for the applicability of these methods, since we do not consider data release to be a real-time problem. The computation of F for different combinations of attributes can be easily parallelized if higher levels of performance are required.

6.3 Encodings on Non-Binary Attributes

As discussed in Section 5.1, we implement four encodings for datasets with non-binary attributes. Figures 5–8 show all experimental results over count and classification tasks, on non-binary datasets Adult and BR2000. The name of each reported method indicates the type of encoding and the score function used in the exponential mechanism, for example, Hierarchical-R stands for the hierarchical encoding and the score function R. We set parameters of PrivBayes to their default values: $\beta = 0.3$ and $\theta = 4$, which will be justified in the next set of experiments.

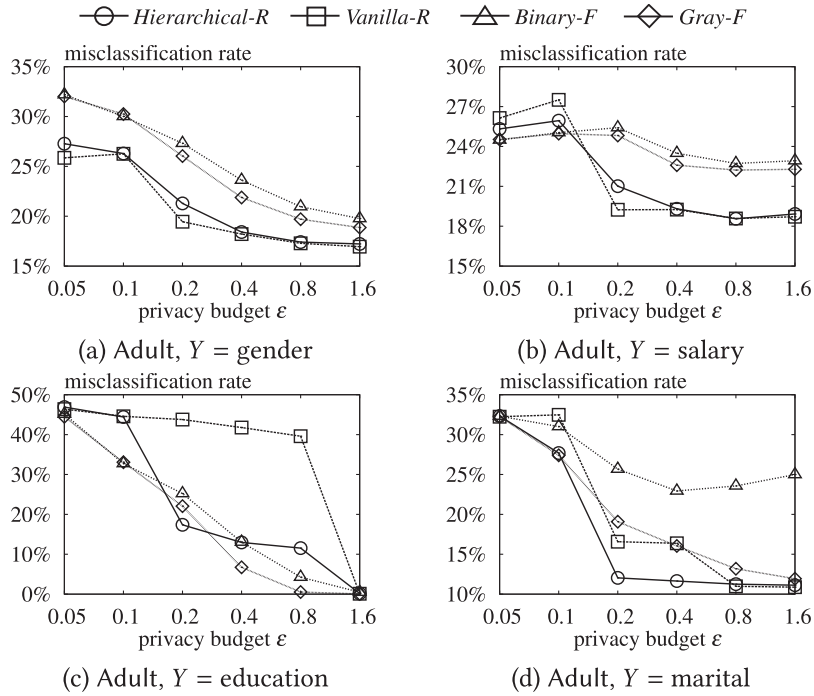
On the results of count queries in Figures 5 and 6, non-binary encodings are clearly superior to binary ones when ε is small, but the gap shrinks as ε increases. It implies that, in the binary encodings, the information loss caused by the redundant attributes overwhelms the gain of flexible encoding, especially when the perturbation in networking learning is large. This matches our analysis on the drawbacks of binary encoding in Section 5.1. Between two non-binary encodings, Hierarchical-R and Vanilla-R achieve almost identical results, which again proves that the count queries in low-dimensional marginals are not sensitive to the flexibility of encoding.

The second set of tasks that we evaluate is training SVM classifiers (see Figures 7 and 8). In summary, Hierarchical-R achieves the best overall performance among all methods, and its performance is quite stable over different tasks. The reason is that Hierarchical-R is the only method



(a) BR2000, Q_2 (b) BR2000, Q_3

Fig. 6. Comparison among different encodings (α-way marginals on BR2000).



(a) Adult, Y = gender (b) Adult, Y = salary

(c) Adult, Y = education (d) Adult, Y = marital

Fig. 7. Comparison among different encodings (multiple SVM classifiers on Adult).

that provides flexible encoding while preserving the semantics of attributes. Take the tasks in Figures 7(a) and 7(c) as examples. The former task is to predict the gender of an individual. As the predicted attribute is the most flexible in nature (a binary attribute), the advantage of flexible encoding becomes insignificant. As a consequence, the encodings aware of attribute semantics are more preferable. In contrast, the latter task requires to predict the value of a categorical attribute of large domain size (i.e., 16), for which Vanilla-R fails to preserve any correlation until $\varepsilon > 0.8$.

To conclude, we strongly recommend to use the hierarchical encoding on datasets with general domains. In the rest of experiments, we use Hierarchical-R as the routine of PrivBayes on datasets Adult and BR2000.

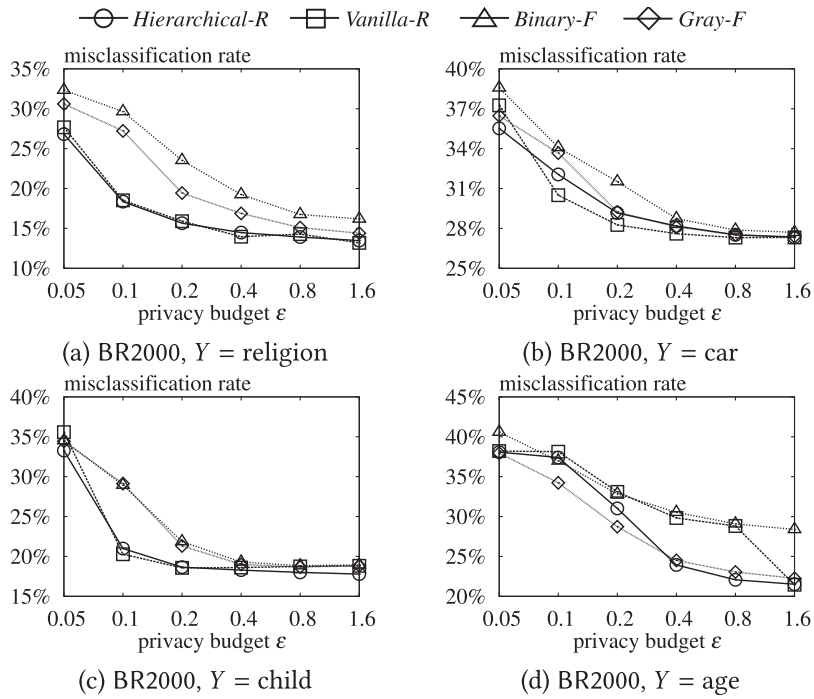


Fig. 8. Comparison among different encodings (multiple SVM classifiers on BR2000).

6.4 Choice of Parameters

Recall that PrivBayes has two internal parameters: the budget allocation parameter, β, and the usefulness of noisy marginals, θ. In this set of experiments, we quantify their impact to the performance of PrivBayes.

The parameter β controls the portion of privacy budget assigned to the network learning and distribution learning phases. To evaluate the effect of β on PrivBayes, we pick one counting and one classification task for each dataset. For instance, in the case of NLTCS, these are counting query Q_4 and the classification query to predict disabilities on getting outside. We examine the performance of PrivBayes (on $\theta = 4$) in answering these queries, with varying β and ε. Figure 9 presents the error metrics for these two tasks (the average variation distance on counting and the misclassification rate on classification, respectively) as β varies. Observe that the error tends to be higher when β is very small or very large.[6] This is because (i) small β leads to very noisy Bayesian network in the first phase of PrivBayes, which makes the synthetic dataset drastically different from the input data, and (ii) large β makes it difficult for PrivBayes to construct high quality marginals in its second phase, which also leads to inferior synthetic data. However, we observe that in general there is a good range of β values where the error is comparable to the minimum. This tends to occur below the mid-point, that is, we should allocate a greater proportion of the budget to parameter setting than to the choice of model. We believe that this is the case, because for these datasets, the tasks are quite robust to a slightly suboptimal choice of model, and we are

[6]One exception is with ACS, Q_4 at $\beta = 0.9$, where the θ-usefulness sets the degree of Bayesian networks to 0, that is, we just materialize all one-way marginals. In this case, β is automatically reset to 0 as there is only one possible network to be learnt.

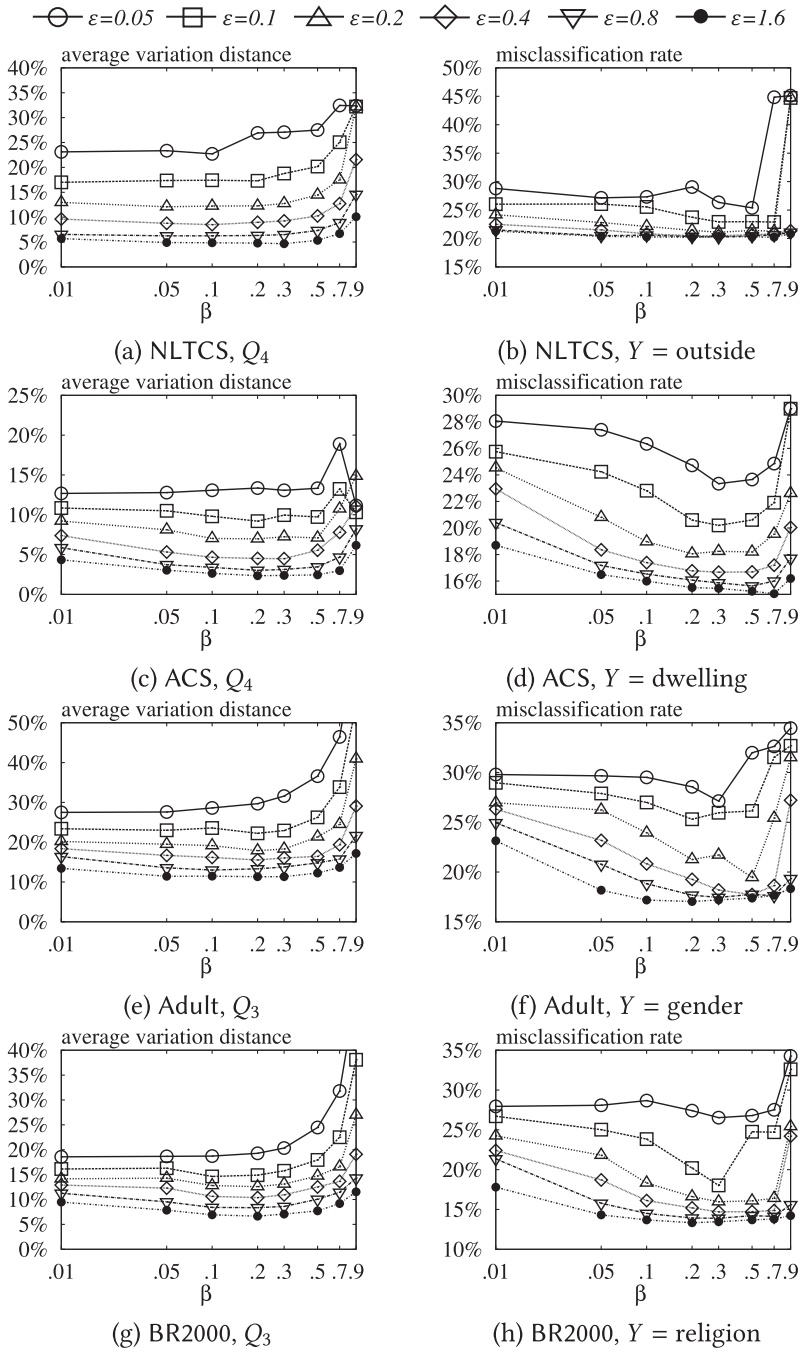


Fig. 9. Choice of β.

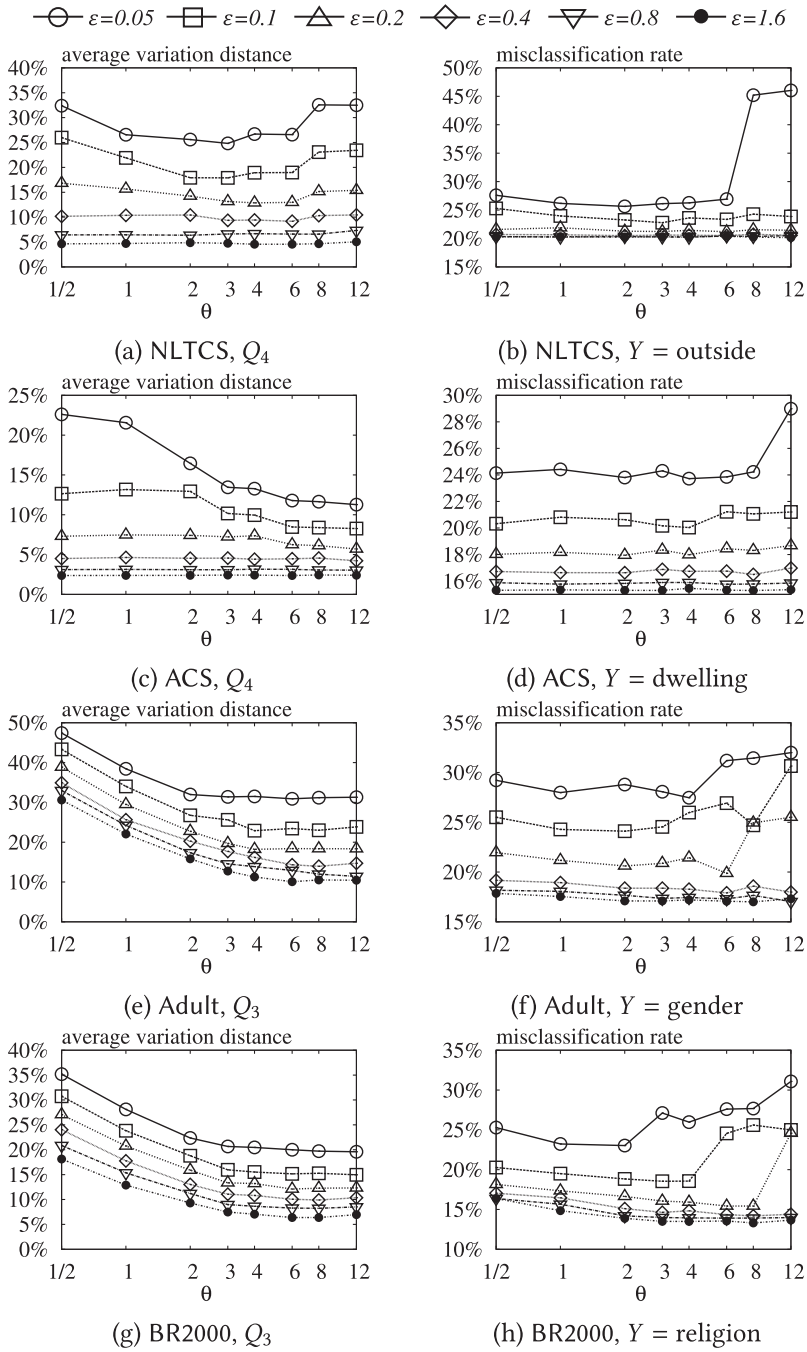
better off to devote our budget to learning a good-enough model with greater fidelity. Based on Figure 9, we surmise that an appropriate value for β should be in the range of $[0.2, 0.5]$. For all subsequent experiments, we set $\beta = 0.3$.

The second set of experiments is on the tuning of θ. As discussed in Section 4.5, we adopt the θ-usefulness criterion to automatically select the degree of the Bayesian network, based on the number of tuples in the input data D as well as the domains of the attributes in D. To evaluate, we adopt the same set of tasks as in Figure 9, with β set to our new default value 0.3. Figure 10 illustrates the results as a function of θ and ε. Similar to its behaviors on tuning β, PrivBayes achieves the near-best performance on a quite wide range of values of θ, which is consistent with our analysis in Section 4.5. In particular, the results suggest an appropriate value should be within $[3, 6]$. In the following experiments, we will use $\theta = 4$ as the default value.

After finalizing the default values of β and θ, we conduct the last set of experiments aiming to distinguish the approximation error from network learning and distribution learning phases. Toward this end, we introduce two alternatives of PrivBayes, namely, *BestNetwork* and *BestMarginal*. *BestNetwork* is an instance of PrivBayes with unlimited budget for network learning phase, while *BestMarginal* is an instance with unlimited budget for distribution learning. Therefore, the performance gap between PrivBayes and *BestNetwork* (resp. *BestMarginal*) indicates the error introduced by network learning (respectively, distribution learning) phase. Figure 11 illustrates the performance of three methods on our battery of eight tasks. In summary, the error in counting queries mainly comes from the noise introduced in materializing private marginals, as the *BestMarginal* significantly outperforms the other two methods especially on non-binary datasets Adult and BR2000. This is consistent with the results on tuning β and θ: counting queries lead to a choice of a small β (i.e., less noise in marginals) and large θ (i.e., low-degree marginals that are robust against noise). In contrast, classification tasks suffer more from noisy networks, resulting in a slight preference to relatively larger β and smaller θ. However, across all these experiments, we are satisified with the choice of $\beta = 0.3$, $\theta = 4$ as defaults for a range of tasks.

6.5 α-way Marginals

This section compares PrivBayes with the *Laplace*, *Fourier*, and *Uniform* approaches on eight sets of marginals over four datasets. We additionally compare *Contingency*, *MWEM* on four sets of marginals over NLTCS and ACS. As noted earlier, these methods do not scale to the full collection of datasets used. Figures 12–15 show the average variation distance of each method for each query set Q_α, varying the privacy budget ε. PrivBayes clearly outperforms the other five baselines in all cases. The relative superiority of PrivBayes is more pronounced when (i) ε decreases or (ii) the value of α increases. To explain, observe that when ε is small, PrivBayes chooses to construct a very low-degree Bayesian network (down to $k = 0$), due to the θ-usefulness criterion. As a consequence, the marginal distributions in the second phase of PrivBayes will be more robust against noise injection, which ensures the quality of the synthetic data will not degrade too significantly. In contrast, the performance of *Laplace* and *Fourier* is highly sensitive to ε, owing to which they incur considerable errors when ε decreases. The simple *Contingency* approach generates inaccurate marginals with low privacy budget, with performance improving slightly as ε increases. On ACS, it achieves almost identical results to the naive *Uniform* method, which implies the noise in the released data is overwhelming. *MWEM* also suffers when ε is small, as the number of iterations to improve the approximation is highly limited. Specifically, we change the budget consumed in each iteration of *MWEM* from 1.0 (the default value by authors) to 0.05, to ensure that at least one round of improvement occurs. But even with this modification, the performance of *MWEM* does not significantly surpass the naive *Uniform* when $\varepsilon < 0.2$.



(a) NLTCS, Q_4 (b) NLTCS, Y = outside

(c) ACS, Q_4 (d) ACS, Y = dwelling

(e) Adult, Q_3 (f) Adult, Y = gender

(g) BR2000, Q_3 (h) BR2000, Y = religion

Fig. 10. Choice of θ.

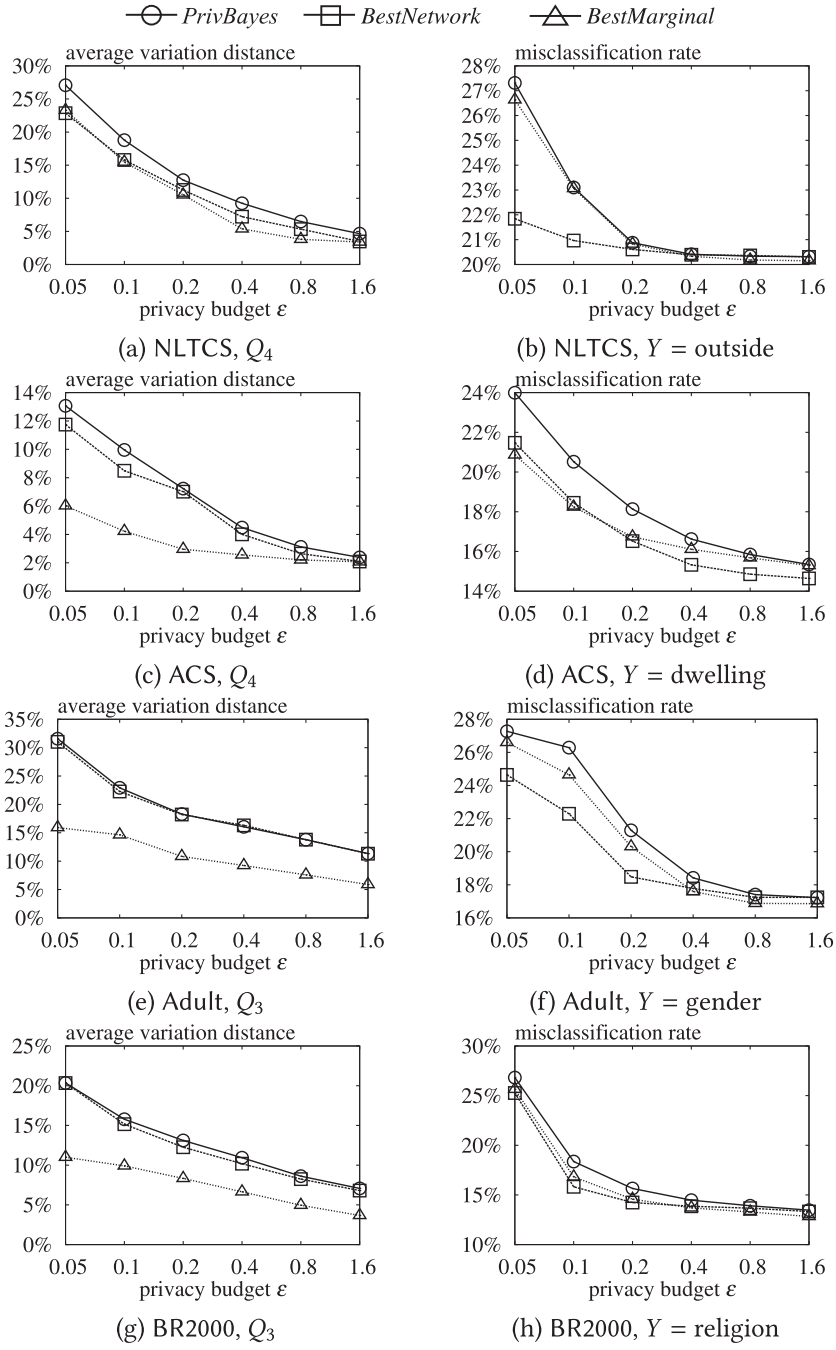
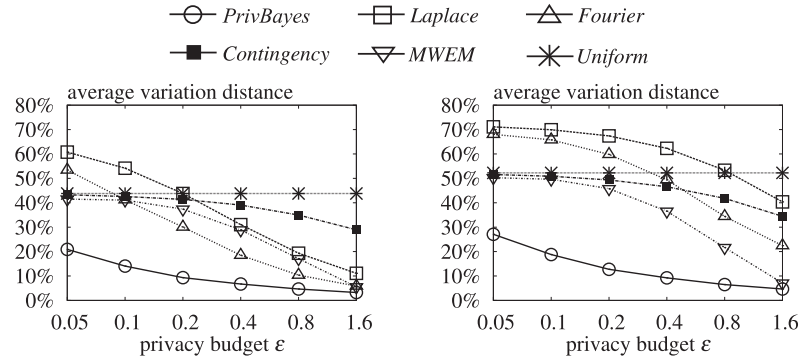
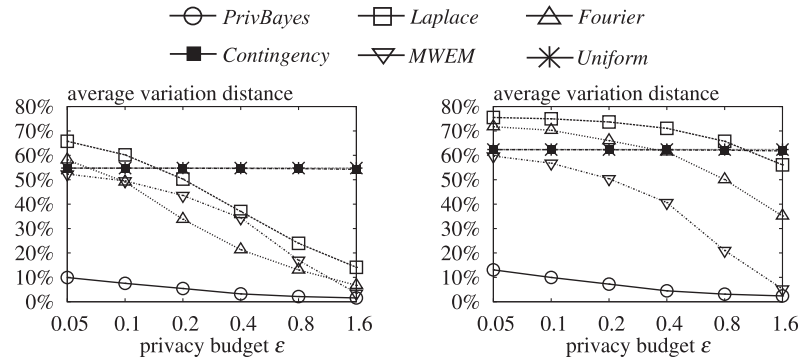


Fig. 11. Source of error.



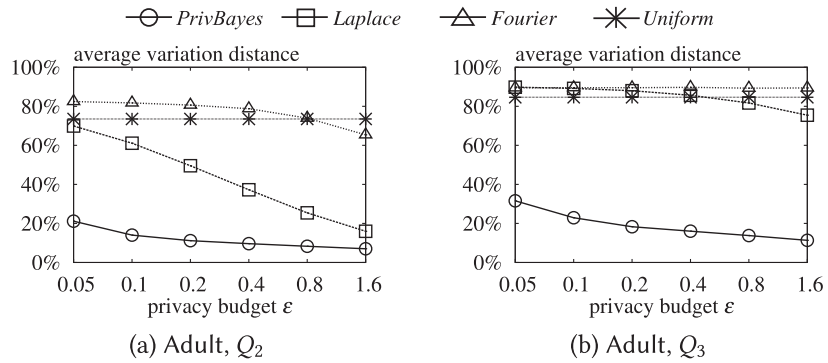
(a) NLTCS, Q_3 (b) NLTCS, Q_4

Fig. 12. Comparison to baselines (α-way marginals on NLTCS).



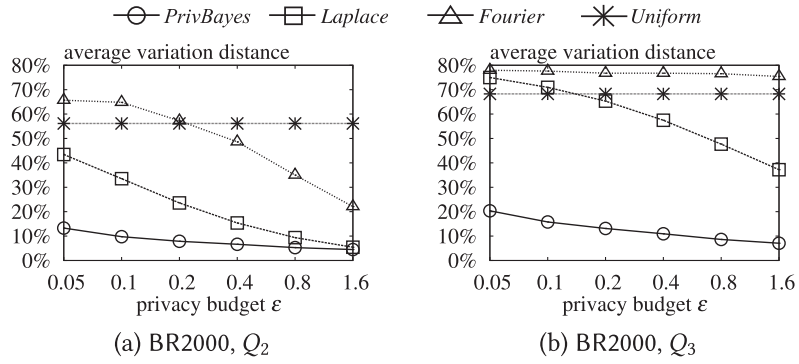
(a) ACS, Q_3 (b) ACS, Q_4

Fig. 13. Comparison to baselines (α-way marginals on ACS).



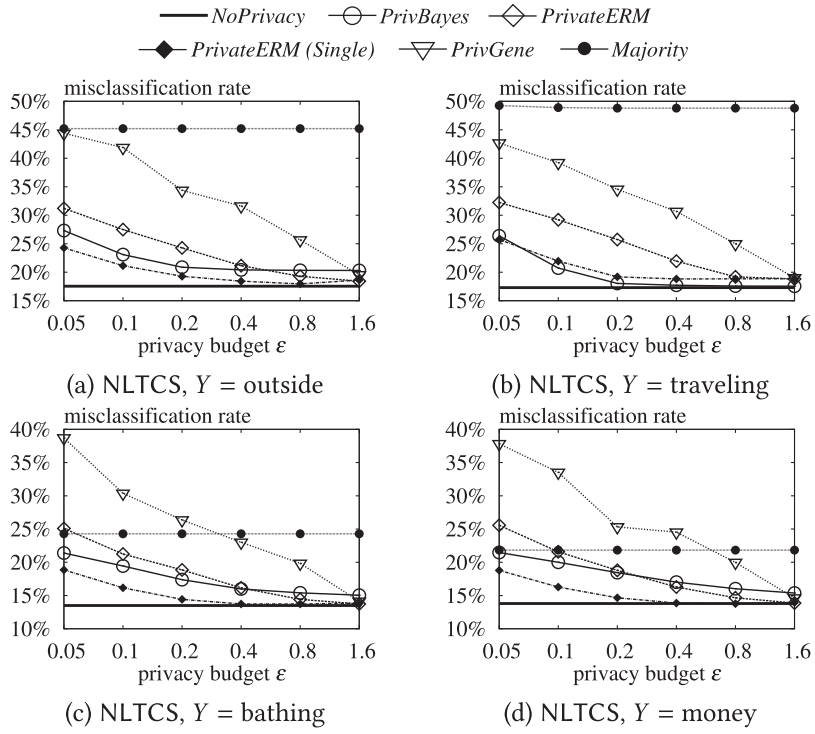
(a) Adult, Q_2 (b) Adult, Q_3

Fig. 14. Comparison to baselines (α-way marginals on Adult).



(a) BR2000, Q_2 (b) BR2000, Q_3

Fig. 15. Comparison to baselines (α-way marginals on BR2000).



(a) NLTCS, Y = outside (b) NLTCS, Y = traveling

(c) NLTCS, Y = bathing (d) NLTCS, Y = money

Fig. 16. Comparison to baselines (multiple SVM classifiers on NLTCS).

Meanwhile, when α increases, the query set Q_α corresponds to a larger set of marginals, in which case the queries Q_α have a higher sensitivity. Therefore, *Laplace* needs to inject a larger amount of noise into Q_α for privacy protection, leading to higher query errors. *Fourier* also suffers from a similar issue. On the other hand, the error of PrivBayes is not sensitive to α, as the Bayesian network constructed (once) by PrivBayes enables it to nicely capture the correlations among attributes. Consequently, it can closely approximate the marginals pertinent to Q_α even when α increases.

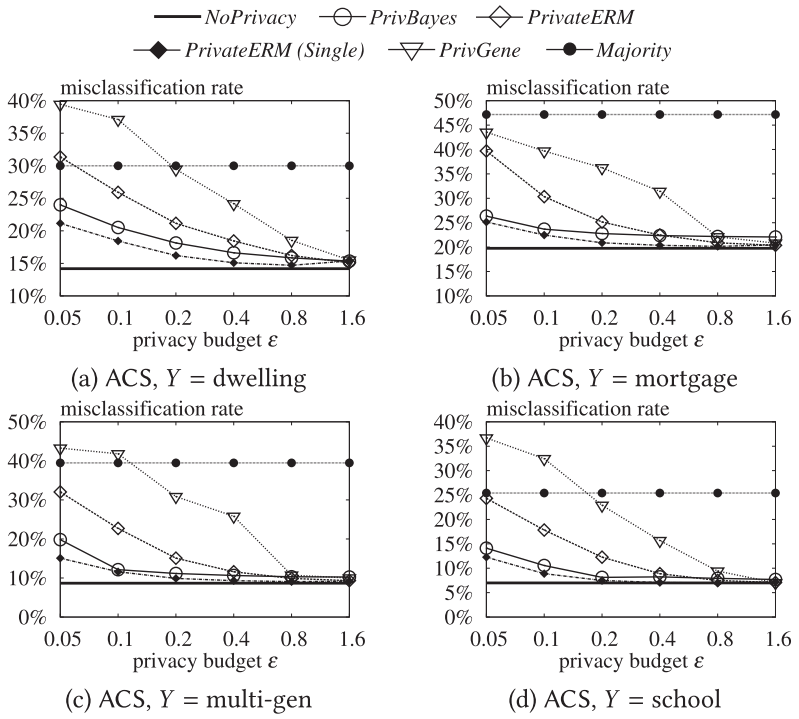


Fig. 17. Comparison to baselines (multiple SVM classifiers on ACS).

6.6 Multiple SVM Classifiers

In the last set of experiments, we evaluate different methods for SVM classification. As explained in Section 6.1, on each dataset, we train four SVM classifiers simultaneously. For PrivBayes, we apply it to generate only *one* synthetic dataset D^* from each training set, and then use D^* to train all four classifiers required. The other differentially private methods (i.e., *PrivateERM*, *PrivGene*, and *Majority*) can only produce one classifier at a time. Therefore, for each of those method, we evenly divide the privacy budget ε into four parts, and use $\varepsilon/4$ budget to train each classifier. To illustrate the performance of *PrivateERM* when building a single classifier, we include an additional baseline referred to as *"PrivateERM (Single)."* This baseline is identical to *PrivateERM*, expect that it uses a privacy budget of ε (instead of $\varepsilon/4$) in training each classifier.

Figures 16–19 show the misclassification rate of each method as a function of the overall ε. The error of *NoPrivacy* remains unchanged for all ε, since it does not enforce ε-differential privacy—it represents the best case to aim for. The accuracy of *Majority* is insensitive to ε, since (i) it performs classification only by checking whether there exists more than 50% tuples in the training set with a certain label, and (ii) this check is quite robust against noise injection when the number of tuples in the training set is large (as is the case in our experiments). As for the other methods, PrivBayes consistently outperforms *PrivateERM* and *PrivGene* in almost all datasets, except for a few settings in Figures 16. Interestingly, in Figure 18(b), the misclassification rate of PrivBayes increases when ε changes from 0.05 to 0.1. The reason is that we have tuned the parameters for PrivBayes based on the overall performance among all datasets, and hence, our choices of β and θ do not always guarantee the best performance for PrivBayes on every specific task. Overall, PrivBayes is superior to both *PrivateERM* and *PrivGene* on the classification task.

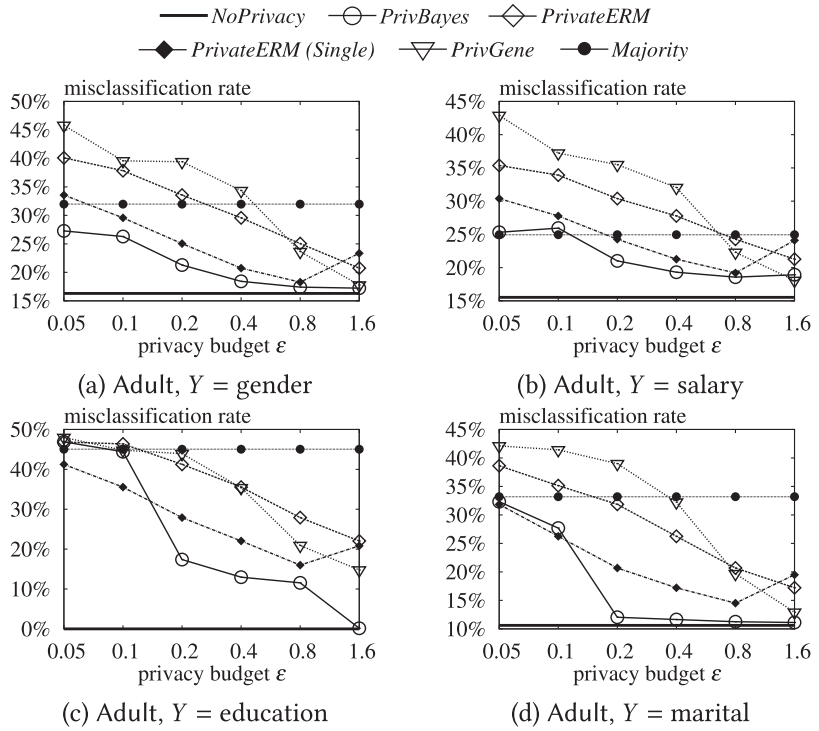


Fig. 18. Comparison to baselines (multiple SVM classifiers on Adult).

On the other hand, PrivBayes is outperformed by *PrivateERM (Single)*[7] in most cases (except on Adult). This is reasonable given that *PrivateERM* is designed solely for SVM classification, whereas PrivBayes does not specifically optimize for SVM classification when it generates the synthetic data. In general, the fact that PrivBayes can support multiple analytical tasks (without incurring extra privacy overhead) makes it highly favorable in the common case when the user does not have a specific task in mind and would like to conduct *exploratory* data analysis by experimenting with various tasks.

7 CONCLUDING REMARKS

The model of Bayesian networks has proven a powerful way to represent correlated data approximately. We have seen that it is also highly effective as a model to release data while respecting privacy. We see that data released this way is very accurate and indeed offers better accuracy than customized mechanisms for particular objectives, such as classification. A crucial part of our approach is the crafting of novel score functions as surrogates for mutual information, which dramatically improve the quality of the released data. Our results show that the sampled data is generally useful; one direction for exploration is whether certain questions could be answered directly from the materialized model and its parameters, rather than via random sampling.

[7]The behavior of *PrivateERM (Single)* on Adult with $\varepsilon = 1.6$ is an artifact of the algorithm itself: it computes an internal parameter ε'_p as a function of ε, which yields a sub-optimal choice when $\varepsilon = 1.6$.

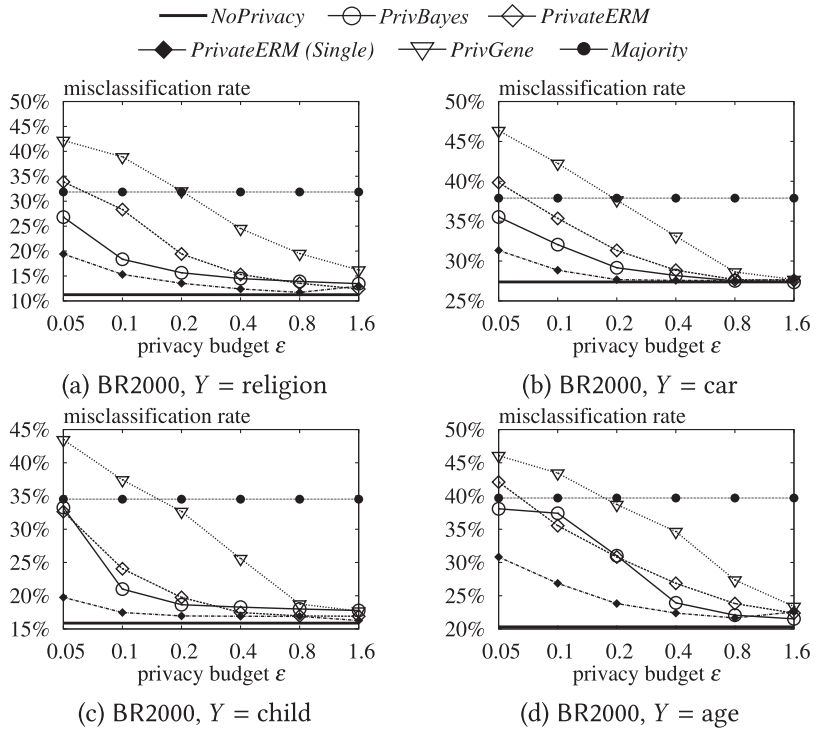


Fig. 19. Comparison to baselines (multiple SVM classifiers on BR2000).

The natural next step is to extend this work to databases over multiple tables. The approach of building a graphical model and releasing this privately applies here also. However, care is needed: in the examples we consider, each individual affects a single row of the initial database table. As we consider more complex schemas, the impact of an individual (and hence the scale of noise needed for privacy) may grow very large, and a more careful analysis is needed to ensure that noise does not outweigh the signal.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for many useful comments and suggestions. The interpretation of R in terms of I via Pinsker's inequality at the end of Section 5 is due to a reviewer.

ELECTRONIC APPENDIX

The electronic appendix to this article can be accessed in the ACM Digital Library.

REFERENCES

[1] Kevin Bache and Moshe Lichman. 2013. UCI Machine Learning Repository (2013). Retrieved from http://archive.ics.uci.edu/ml.

[2] 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 *Proceedings of PODS*. 273–282.

[3] Roberto J. Bayardo and Rakesh Agrawal. 2005. Data privacy through optimal k-anonymization. In *Proceedings of ICDE*. 217–228.

[4] Raghav Bhaskar, Srivatsan Laxman, Adam Smith, and Abhradeep Thakurta. 2010. Discovering frequent patterns in sensitive data. In *Proceedings of KDD*. 503–512.

[5] Hayes Brian. 2002. Computing science: The easiest hard problem. *American Scientist* 90, 2 (2002), 113–117.
[6] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM TIST* (2011), 27.
[7] Kamalika Chaudhuri and Claire Monteleoni. 2008. Privacy-preserving logistic regression. In *Proceedings of NIPS*. 289–296.
[8] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. 2011. Differentially private empirical risk minimization. *J. Mach. Learn. Res.* 12 (2011), 1069–1109.
[9] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. 2015. Differentially private high-dimensional data publication via sampling-based inference. In *Proceedings of SIGKDD*. 129–138.
[10] Yan Chen and Ashwin Machanavajjhala. 2015. On the privacy properties of variants on the sparse vector technique. *CoRR* abs/1508.07306 (2015).
[11] David Maxwell Chickering, David Heckerman, and Christopher Meek. 2004. Large-sample learning of bayesian networks is NP-hard. *J. Mach. Learn. Res.* 5 (2004), 1287–1330.
[12] C. K. Chow and C. N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Info. Theory* 14 (1968), 462–467.
[13] CIA. 2015. *The World Factbook 2014–15*. Government Printing Office.
[14] Graham Cormode, Cecilia Magdalena Procopiuc, Entong Shen, Divesh Srivastava, and Ting Yu. 2012. Differentially private spatial decompositions. In *Proceedings of ICDE*.
[15] Graham Cormode, Cecilia Magdalena Procopiuc, Divesh Srivastava, and Thanh T. L. Tran. 2012. Differentially private publication of sparse data. In *Proceedings of ICDT*.
[16] Aleksandr B. Cybakov. 2009. *Introduction to Nonparametric Estimation*. Springer.
[17] Bolin Ding, Marianne Winslett, Jiawei Han, and Zhenhui Li. 2011. Differentially private data cubes: Optimizing noise sources and consistency. In *Proceedings of SIGMOD*. 217–228.
[18] Cynthia Dwork. 2006. Differential privacy. In *Proceedings of ICALP*. 1–12.
[19] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Proceedings of TCC*. 265–284.
[20] Cynthia Dwork and Aaron Roth. 2013. The algorithmic foundations of differential privacy. *Theor. Comput. Sci.* 9, 3–4 (2013), 211–407.
[21] Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. 2009. Private coresets. In *Proceedings of STOC*. 361–370.
[22] Arik Friedman and Assaf Schuster. 2010. Data mining with differential privacy. In *Proceedings of KDD*. 493–502.
[23] 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 *Proceedings of ICML*. 1170–1178.
[24] F. Gray. 1953. Pulse code communication (March 17 1953). Retrieved from https://www.google.com/patents/US2632058. U.S. Patent 2,632,058.
[25] Moritz Hardt. 2011. *A Study of Privacy and Fairness in Sensitive Data Analysis*. Ph.D. Dissertation. Princeton University.
[26] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A simple and practical algorithm for differentially private data release. In *Proceedings of NIPS*. 2348–2356.
[27] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the accuracy of differentially private histograms through consistency. *PVLDB* 3, 1 (2010), 1021–1032.
[28] Vijay S. Iyengar. 2002. Transforming data to satisfy privacy constraints. In *Proceedings of IGKDD*. 279–288.
[29] Daniel Kifer, Adam D. Smith, and Abhradeep Thakurta. 2012. Private convex optimization for empirical risk minimization with applications to high-dimensional regression. *J. Mach. Learn. Res. Proc. Track* 23 (2012), 25.1–25.40.
[30] Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.
[31] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. 2010. Optimizing linear counting queries under differential privacy. In *Proceedings of PODS*. 123–134.
[32] Chao Li and Gerome Miklau. 2012. An adaptive mechanism for accurate query answering under differential privacy. *PVLDB* 5, 6 (2012), 514–525.
[33] Chao Li and Gerome Miklau. 2013. Optimal error of query sets under the differentially-private matrix mechanism. In *Proceedings of ICDT*. 272–283.
[34] Ninghui Li, Wahbeh Qardaji, Dong Su, and Jianneng Cao. 2012. PrivBasis: Frequent itemset mining with differential privacy. *PVLDB* 5, 11 (2012), 1340–1351.
[35] Kenneth G. Manton. 2010. National long-term care survey: 1982, 1984, 1989, 1994, 1999, and 2004. (2010).
[36] Dimitris Margaritis. 2003. *Learning Bayesian Network Model Structure from Data*. Ph.D. Dissertation. School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.
[37] Frank McSherry and Ratul Mahajan. 2010. Differentially-private network trace analysis. In *Proceedings of SIGCOMM*. 123–134.
[38] Frank McSherry and Ilya Mironov. 2009. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *Proceedings of KDD*. 627–636.

[39] Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *Proceedings of FOCS*. 94–103.
[40] Prashanth Mohan, Abhradeep Thakurta, Elaine Shi, Dawn Song, and David Culler. 2012. GUPT: Privacy preserving data analysis made easy. In *Proceedings of SIGMOD*. 349–360.
[41] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of STOC*. 75–84.
[42] Vibhor Rastogi and Suman Nath. 2010. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of SIGMOD*. 735–746.
[43] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. 2012. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *J. Priv. Confident.* 4, 1 (2012), 65–100.
[44] Steven Ruggles, Katie Genadek, Ronald Goeken, Josiah Grover, and Matthew Sobek. 2015. Integrated Public Use Microdata Series: Version 6.0. (2015). Retrieved from https://international.ipums.org.
[45] Adam Smith. 2011. Privacy-preserving statistical estimation with optimal convergence rate. In *Proceedings of STOC*.
[46] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2010. Differential privacy via wavelet transforms. In *Proceedings of ICDE*. 225–236.
[47] Grigory Yaroslavtsev, Graham Cormode, Cecilia M. Procopiuc, and Divesh Srivastava. 2013. Accurate and efficient private release of datacubes and contingency tables. In *Proceedings of ICDE*. 745–756.
[48] Ganzhao Yuan, Zhenjie Zhang, Marianne Winslett, Xiaokui Xiao, Yin Yang, and Zhifeng Hao. 2012. Low-rank mechanism: Optimizing batch queries under differential privacy. *PVLDB* 5, 11 (2012), 1352–1363.
[49] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2014. PrivBayes: Private data release via bayesian networks. In *Proceedings of SIGMOD*. 1423–1434.
[50] Jun Zhang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Marianne Winslett. 2013. PrivGene: Differentially private model fitting using genetic algorithms. In *Proceedings of SIGMOD*. 665–676.

Received July 2016; revised March 2017; accepted August 2017