# Learning Bayesian Networks under Local Differential Privacy

by

**Alireza Khodaie**

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ ÜNİVERSİTESİ

January 16, 2025

**Learning Bayesian Networks under Local Differential Privacy**

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

**Alireza Khodaie**

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assist. Prof. Dr. Mehmet Emre Gürsoy (Advisor)

_____

Prof. Dr. Alptekin Küpçü

_____

Prof. Dr. Yücel Saygın

Date: _____

*To my dear family*

*My mother, father, brother and sister*

# ABSTRACT

**Learning Bayesian Networks under Local Differential Privacy**
**Alireza Khodaie**
**Master of Science in Computer Science and Engineering**
**January 16, 2025**

Bayesian networks are widely used for causal discovery and probabilistic modeling across diverse domains including healthcare, multi-dimensional data analysis, environmental modeling, and industrial processes. Although previous work has studied the learning of Bayesian networks under centralized differential privacy, to the best of our knowledge, the problem of learning Bayesian networks under local differential privacy (LDP) remains open. In this thesis, we address this problem by proposing two solution methods for learning Bayesian networks under LDP: LDP-BN and LDP-BN+. Our first solution called LDP-BN utilizes a novel algorithm for computing mutual information values necessary for building a Bayesian network under LDP, but it suffers from high utility loss since the privacy budget needs to be divided into many pairs of attributes and candidate parent sets. To reduce the amount of noise, we propose LDP-BN+ which utilizes a novel density-aware covering design algorithm that ensures all necessary mutual information values will be computed while the privacy budget is used more effectively. We experimentally evaluate LDP-BN and LDP-BN+ using multiple utility metrics and datasets. Results show that LDP-BN+ outperforms LDP-BN and enables the generation of high-utility Bayesian networks that can be used in practice.

# ÖZETÇE

**Bayes Ağlarının Lokal Diferansiyel Mahremiyet Altında Öğrenilmesi**
**Alireza Khodaie**
**Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans**
**16 Ocak 2025**

Bayes ağları, sağlık, çok-boyutlu veri analizi, çevresel modelleme ve endüstriyel süreçler de dahil olmak üzere birçok alanda nedensel keşif ve olasılıksal modelleme amacıyla yaygın olarak kullanılmaktadır. Önceki çalışmalar her ne kadar merkezi diferansiyel mahremiyet altında Bayes ağlarının öğrenilmesini çalışmış olsalar da, bildiğimiz kadarıyla lokal diferansiyel mahremiyet (LDP) altında Bayes ağlarının öğrenilmesi problemi halen açık bir problemdir. Bu tezde, Bayes ağlarının LDP altında öğrenilmesi problemini çözmek için iki çözüm yöntemi öneriyoruz: LDP-BN ve LDP-BN+. İlk çözümümüz olan LDP-BN, LDP altında Bayes ağı kurmak için gerekli olan ortak enformasyon miktarı değerlerini hesaplamak için özgün bir algoritma kullanır; fakat mahremiyet bütçesinin çok sayıda öznitelik ve aday ebeveyn kümesine bölünmesi gerektiği için yüksek fayda kaybına uğramaktadır. Gürültü miktarını azaltmak için önerdiğimiz LDP-BN+ çözümü, gerekli olan tüm ortak enformasyon miktarı değerlerinin hesaplanacağını garanti ederken, mahremiyet bütçesinin daha verimli kullanılabilmesi adına özgün bir yoğunluk-farkındalıklı kaplama tasarımı algoritması kullanır. Çalışmamızda, LDP-BN ve LDP-BN+ çözümlerini birden fazla fayda ölçütü ve veri kümesi kullanarak deneysel olarak değerlendiriyoruz. Sonuçlar, LDP-BN+'nın LDP-BN'e kıyasla daha iyi performans gösterdiğini ve LDP-BN+ sayesinde pratikte kullanılabilecek, yüksek faydaya sahip Bayes ağlarının öğrenilebildiğini göstermektedir.

# ACKNOWLEDGMENTS

I sincerely thank my advisor and mentor, *Assist. Prof. Dr. Mehmet Emre Gürsoy*, for his guidance, support, and encouragement throughout my Master's journey. His knowledge and dedication have inspired me and helped me overcome challenges. I am truly grateful for his mentorship, which made this work possible.

I would like to thank *Prof. Dr. Alptekin Küpçü* from Koç University and *Prof. Dr. Yücel Saygın* from Sabancı University for serving on my thesis jury committee and for their valuable comments and suggestions that helped improve this work.

I would like to sincerely thank my family and friends for their support and encouragement throughout my Master's journey. My family's love and unwavering belief in me have been my greatest source of strength, motivating me to overcome challenges and strive for success.

To my friends, Aydin and Amir, I am deeply grateful for your understanding, kindness, and the moments of joy that brightened even the most difficult days. I also want to convey my gratitude to the members of the SPADE Lab, in particular Kemal and Emirhan, for their support and friendship. Your presence has been invaluable, and this achievement is as much yours as it is mine.

Lastly, this research was supported by The Scientific and Technological Research Council of Turkiye (TUBITAK) under project number 121E303. I sincerely thank TUBITAK for their support.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ABBREVIATIONS

| | |
|---|---|
| DP | Differential Privacy |
| LDP | Local Differential Privacy |
| OUE | Optimized Unary Encoding |
| DAG | Directed Acyclic Graph |
| JDE | Joint Distribution Error |
| MDE | Marginal Distribution Error |
| SHD | Structural Hamming Distance |

# Chapter 1

# INTRODUCTION

Bayesian networks have been established as powerful tools for causal discovery and probabilistic modeling, and they found widespread applications in fields ranging from healthcare to environmental systems and industrial processes [Zhang et al., 2017, Chen and Pollino, 2012, Kaur et al., 2021, Hittmeir et al., 2022]. In particular, Bayesian networks have been used actively in multi-dimensional data modeling and synthetic data generation under centralized differential privacy (DP) [Zhang et al., 2017, Bao et al., 2021, Qi et al., 2020, Ma et al., 2023]. Yet, centralized DP requires trusted databases to store and manage sensitive data records, which poses privacy risks.

With the advent of local differential privacy (LDP), users can now perturb their records before sharing the perturbed versions with the server (i.e., the data collector). This enables data collection and analysis in environments where users do not trust the data collector. Thanks to its desirable privacy properties, LDP has received significant attention from the academia and the industry, and it has also been deployed in various products of tech companies such as Apple, Google, and Microsoft [Cormode et al., 2018, Gursoy et al., 2022, Yang et al., 2023, Erlingsson et al., 2014, Ding et al., 2017]. On the other hand, while there exist a few works on learning Naive Bayes classifiers and other causal models under LDP [Xue et al., 2019, Yilmaz et al., 2020, Ohnishi and Awan, 2023, Binkyte et al., 2024], to the best of our knowledge, the problem of learning Bayesian networks under LDP is an open problem.

In this thesis, we address this problem by proposing two solution methods for learning Bayesian networks under LDP: LDP-BN and LDP-BN+. We first introduce

a common Bayesian network construction algorithm which is used as the basis of our two solutions. Then, we propose LDP-BN which utilizes a novel algorithm for computing the mutual information between an attribute $A_i$ and a set of attributes $S$ while satisfying LDP. This algorithm is used by LDP-BN so that the mutual information values necessary for Bayesian network construction are computed with LDP. Yet, we observe that LDP-BN suffers from substantial utility loss since the total LDP budget $\varepsilon$ needs to be divided into each pair of $A_i$ and $S$, causing high noise in each mutual information computation. To reduce the amount of noise, we propose and develop LDP-BN+ which employs a novel density-aware covering design algorithm, enabling more effective use of the $\varepsilon$ budget.

We experimentally evaluate LDP-BN and LDP-BN+ using three popular datasets (Adult, ACS Income, Titanic) and three utility metrics (SHD, JDE, MDE). Results demonstrate that LDP-BN+ outperforms LDP-BN. Furthermore, the utility loss remains low under popularly used values of the privacy budget $\varepsilon$, especially in the case of larger datasets. This shows that LDP-BN+ can indeed be used in practice to build high-utility Bayesian networks. Furthermore, motivated by the applications of Bayesian networks to synthetic data generation in centralized DP, we perform case studies by generating synthetic datasets using our Bayesian networks. To measure the utility of synthetic datasets, we train machine learning models on synthetic data and compare the resulting models with models trained on real data. Results show that models trained on synthetic data have similar accuracy to models trained on real data, validating the usefulness of our solution in synthetic data generation.

## *1.1 Contributions*

The main contributions of this thesis can be summarized as follows:

- We propose the problem of learning Bayesian networks under LDP, and we formulate this problem in a way that privatizing the mutual information computations will ensure LDP.

- We propose LDP-BN for learning Bayesian networks under LDP, which utilizes

a novel algorithm for computing the mutual information between an attribute and a set of attributes while satisfying LDP.

- Motivated by the utility loss of LDP-BN, we propose LDP-BN+ which utilizes a novel density-aware covering design algorithm to reduce the amount of LDP noise in mutual information computations.

- We experimentally evaluate LDP-BN and LDP-BN+ using multiple datasets and utility metrics. Results show that LDP-BN+ outperforms LDP-BN and yields high-utility Bayesian networks.

## *1.2 Thesis Organization*

The rest of this thesis is organized as follows. In Chapter 2, we present related work. In Chapter 3, we describe the background and preliminaries regarding the data model, notations, LDP, and Bayesian networks. In Chapter 4, we present our LDP-BN and LDP-BN+ solutions for learning Bayesian networks under LDP. We conduct experimental evaluations and discuss their results in Chapter 5. Finally, Chapter 6 concludes the thesis.

# Chapter 2

# RELATED WORK

## *2.1 Learning Bayesian Networks under Centralized DP*

The problem of learning Bayesian networks under centralized DP was proposed and studied in the seminal work of Zhang et al. [Zhang et al., 2017] titled PrivBayes. The PrivBayes method was also used in the NIST Differential Privacy Synthetic Data Challenge [Bao et al., 2021], but the version described in [Bao et al., 2021] incorporates some modifications to improve utility and efficiency. In [Cheng et al., 2019], Cheng et al. proposed DP-SUBN which enables differentially private high-dimensional data publishing using a sequentially updated Bayesian network in a multi-party setting. In [Qi et al., 2020], Qi et al. proposed APrivBayes which aims to address three aspects of PrivBayes: (i) choosing the network's first node, (ii) reducing the number of subnetworks to increase the privacy budget of each subnetwork, and (iii) reducing the set of candidates when selecting the parent sets of attributes. In [Ma et al., 2023], Ma et al. introduced FAPrivBayes to expedite the generation of Bayesian networks, as well as to refine the networks using the junction tree algorithm.

Application of Bayesian networks to generate synthetic healthcare data was studied in [Kaur et al., 2021]. Practical lessons from generating synthetic healthcare data with Bayesian networks were discussed in [de Benedetti et al., 2020]. In [Hittmeir et al., 2022], Hittmeir et al. extended the PrivBayes approach by substituting its greedy algorithm with a faster genetic algorithm. In addition, they aimed to protect particularly sensitive attributes and decrease disclosure risks by decreasing correlations between specific attributes. In [Ma et al., 2022], Ma et al. proposed NoLeaks, a differentially private causal discovery algorithm using functional causal models. In [Wang et al., 2024], a temporal data publishing method based on dynamic Bayesian

networks was proposed and developed.

While the above works study the problem of learning and/or utilizing Bayesian networks, they operate under *centralized DP*, not *local DP (LDP)*. In contrast, the solutions we propose in this thesis operate under LDP. Hence, our work differs from these works fundamentally in terms of the privacy guarantee.

## *2.2 Bayesian Learning and Causal Discovery under LDP*

There are also a few works that study Bayesian learning and causal discovery under LDP. In [Xue et al., 2019], Xue et al. studied joint distribution estimation and Naive Bayes classification under LDP. Naive Bayes classification under LDP was also studied by Yilmaz et al. [Yilmaz et al., 2020]. In [Ju et al., 2020], Ju et al. proposed a high-dimensional perceptual data publication mechanism under LDP. They utilized Bayesian networks to retain correlations between attributes in high-dimensional data. In [Ohnishi and Awan, 2023], Ohnishi and Awan developed methods for inferring causal effects from locally privatized data under the Rubin Causal Model framework. Finally, Binkyte et al. [Binkyte et al., 2024] considered well-known LDP mechanisms (such as randomized response) and applied these mechanisms in various causal structure learning algorithms to analyze their privacy-utility tradeoffs.

While these works operate under LDP, they study other aspects related to Bayesian learning and causal discovery, such as Naive Bayes classification, Rubin Causal Model, and causal graphs. Our work differs from these works in terms of its goal, i.e., learning a Bayesian network.

# Chapter 3

# BACKGROUND AND PRELIMINARIES

In this chapter, we provide the necessary background, notations, and building blocks for LDP-BN and LDP-BN+. Section 3.1 introduces the multi-dimensional data model and notation. Section 3.2 formally presents LDP and the OUE protocol. Section 3.3 describes Bayesian networks.

## *3.1 Data Model*

We assume a setting with multi-dimensional data where $\mathcal{A} = \{A_1, ..., A_d\}$ denotes the set of attributes. For each attribute $A_i$, we denote its domain by $dom(A_i)$ and the cardinality of its domain by $|dom(A_i)|$. We denote by $a_j \in A_i$ the $j$'th possible value in $dom(A_i)$. For a set of attributes $\mathcal{S} \subseteq \mathcal{A}$, the domain of $\mathcal{S}$ is the Cartesian product of the domains of all attributes in $\mathcal{S}$, i.e.: $dom(\mathcal{S}) = \prod_{A_i \in \mathcal{S}} dom(A_i)$.

Let $\mathcal{P}$ denote a population of users: $\mathcal{P} = \{u_1, u_2, ...\}$. Each user $u \in \mathcal{P}$ has a multi-dimensional record $r_u$ such that:

$$r_u = < r_u^1, r_u^2, \ldots, r_u^d > \tag{3.1}$$

where each $r_u^i \in dom(A_i)$. We exemplify this data model and notations using Figure 3.1. Let Alice, Bob, Dave, and Eve be four users in the population $\mathcal{P}$. The set of attributes consists of $\mathcal{A} = \{$Workclass, Education, Age, Income, Gender$\}$. The records of the four users are shown on the left hand side of Figure 3.1. For example, for Bob's record, it holds that: Private $\in dom($Workclass$)$ and Doctorate $\in dom($Education$)$.

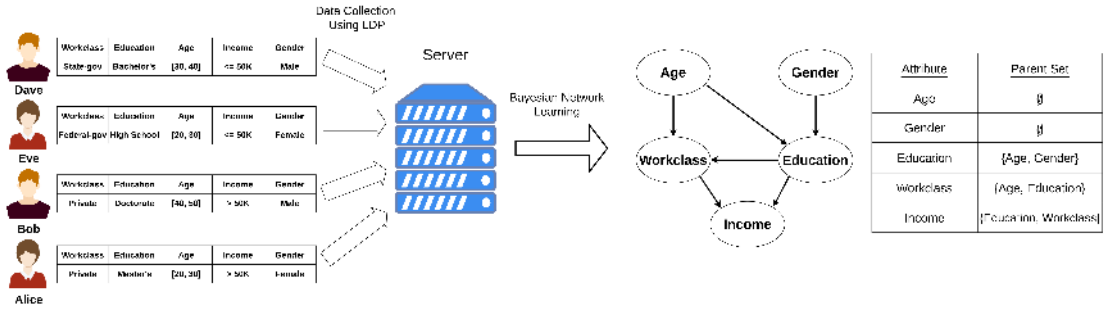


Figure 3.1: Overview of Bayesian network learning under LDP. Four sample users with multi-dimensional records are shown on the left. Users' records are collected by the server using LDP, and then a Bayesian network is learned. The visual representation of the Bayesian network and the corresponding attribute - parent set pairs are shown on the right.

## *3.2 Local Differential Privacy (LDP)*

Local Differential Privacy (LDP) has recently emerged as an accepted privacy standard and received significant research attention from the academia and industry [Cormode et al., 2018, Gursoy et al., 2022, Yang et al., 2023]. Unlike centralized differential privacy (DP) in which users' records are stored in a trusted database, in LDP, each user perturbs their record on their own device and sends the perturbed version to the server (i.e., the data collector). Since perturbation occurs on the users' devices, LDP enables data collection in environments where users do not trust the server. Due to its desirable privacy properties, LDP has also been deployed in various products of tech companies such as Apple, Google, and Microsoft [Erlingsson et al., 2014, app, 2020, Ding et al., 2017].

**Definition 1** ($\varepsilon$-LDP)**.** A randomized algorithm $\Psi$ satisfies $\varepsilon$-local differential privacy ($\varepsilon$-LDP), where $\varepsilon > 0$, if and only if for any two possible records $r_u, \tilde{r}_u$ in $dom(\mathcal{A})$, the following equation holds:

$$\forall y \in Range(\Psi) : \quad \frac{\Pr[\Psi(r_u) = y]}{\Pr[\Psi(\tilde{r}_u) = y]} \leq e^{\varepsilon} \tag{3.2}$$

where $Range(\Psi)$ denotes the set of all possible outputs of $\Psi$.

To ensure LDP, each user $u$ perturbs their true record $r_u$ using a randomized perturbation algorithm $\Psi$. Intuitively, $\Psi$ ensures that given the perturbed output

$y$, it is not possible for the server or any other observer of $y$ to distinguish whether the user's true record was $r_u$ or $\tilde{r}_u$ beyond the ratio $e^{\varepsilon}$. Here, $\varepsilon$ controls the privacy level. Lower $\varepsilon$ yields stronger privacy because $\Pr[\Psi(r_u) = y]$ and $\Pr[\Psi(\tilde{r}_u) = y]$ become closer.

An important property of LDP is *sequential composition*, which enables the total $\varepsilon$ budget to be spent in multiple steps [Dwork et al., 2014, Wang et al., 2018, Ye et al., 2019]. Each step consumes a portion of the budget, and as long as the summation of the portions does not exceed $\varepsilon$, the sequential composition of the steps satisfies $\varepsilon$-LDP.

**Definition 2** (Sequential Composition)**.** Consider $m$ algorithms $\Psi_1, \Psi_2, ..., \Psi_m$ such that each $\Psi_j$ satisfies $\varepsilon_j$-LDP. Then, the sequential execution of $\Psi_1(r_u), \Psi_2(r_u), ...,$ $\Psi_m(r_u)$ satisfies $(\sum_{j=1}^{m} \varepsilon_j)$-LDP.

Several LDP protocols were proposed in the literature to reduce utility loss, user-side computation cost, or the communication cost between the user and the server under varying conditions [Gursoy et al., 2022, Wang et al., 2017, Cormode et al., 2021]. Among them, Optimized Unary Encoding (OUE) was proposed in [Wang et al., 2017] and became one of the widely used building blocks in recent works [Du et al., 2023, Guner and Gursoy, 2023, Liu et al., 2023, Tire and Gursoy, 2024] due to its high utility. In this thesis, we also use OUE as a building block in our solutions. Similar to other LDP protocols, OUE consists of two main components: (i) user-side encoding and perturbation to satisfy LDP on users' devices, and (ii) server-side estimation after collecting perturbed data from the user population.

**User-Side Perturbation in OUE:** OUE uses unary encoding in which user $u$'s true response is encoded into a unary bitvector $B_u$ such that only one position in $B_u$ contains a 1 bit and the remaining positions contain 0s. The perturbation algorithm $\Psi$ of OUE takes $B_u$ as input and outputs a perturbed bitvector $B'_u$ as follows:

$$\Pr\Big[B'_u[i] = \Psi(B_u[i]) = 1\Big] = \begin{cases} \frac{1}{2} & \text{if } B_u[i] = 1 \\ \frac{1}{e^{\varepsilon}+1} & \text{if } B_u[i] = 0 \end{cases} \tag{3.3}$$

In other words, each position $i \in [1, |B_u|]$ is individually processed, and $B_u[i]$ is either retained or flipped in $B'_u[i]$ based on the probabilities in Equation 3.3. These probabilities are selected to minimize the variance of server-side estimation, ensuring that the protocol maintains high accuracy even when $|B_u|$ is large. The resulting bitvector $B'_u$ is then sent from the user to the server.

**Server-Side Estimation in OUE:** The server collects perturbed bitvectors $B'_u$ from users $u \in \mathcal{P}$. Then, the server computes the reported count of position $i$, denoted $\widehat{C}(i)$, as:

$$\widehat{C}(i) = \sum_{u \in \mathcal{P}} B'_u[i] \tag{3.4}$$

The server estimates how many users have $B_u[i] = 1$ in their original bitvectors as:

$$\bar{C}(i) = \frac{2 \cdot \left((e^{\varepsilon} + 1) \cdot \widehat{C}(i) - |\mathcal{P}|\right)}{e^{\varepsilon} - 1} \tag{3.5}$$

where $\bar{C}(i)$ denotes the server's estimate. The server repeats this estimation independently for each $i \in [1, |B_u|]$ to estimate the frequency of each position.

## *3.3 Bayesian Networks*

Consider a centralized setting in which a dataset $\mathcal{D}$ with set of attributes $\mathcal{A}$ consisting of $|\mathcal{D}| = |\mathcal{P}|$ records is available at the server. In this setting, $\mathcal{D}$ can be regarded as an instantiation of the joint probability distribution over all attributes $\mathcal{A}$. A Bayesian network over $\mathcal{A}$ offers a compact representation of this distribution by specifying which attributes in $\mathcal{A}$ are conditionally independent. More formally, a Bayesian network is a directed acyclic graph (DAG) in which each attribute $A_i \in \mathcal{A}$ is represented using a node, and the graph's directed edges indicate dependence or independence relationships among the attributes. For example, consider the Bayesian network in Figure 3.1, constructed using a set of five attributes: $\mathcal{A} = \{\text{Workclass, Education, Age, Income, Gender}\}$. For any two attributes $A_i$, $A_j \in \mathcal{A}$, there are three types of relationships between $A_i$ and $A_j$:

- **Direct dependence.** There is an edge between $A_i$ and $A_j$, e.g., from $A_i$ to $A_j$. This implies that $A_j$ directly depends on $A_i$, i.e., $A_i$ is a direct cause or

influential factor for $A_j$. For example, in Figure 3.1, Income directly depends on Workclass and Education. In this case, $A_i$ is called a parent of $A_j$, and the set of all parents of $A_j$ is called the parent set of $A_j$.

- **Conditional independence.** If there is a path from $A_i$ to $A_j$ but no direct edge between them, then $A_i$ and $A_j$ are said to be conditionally independent given the parent set of $A_j$. For example, in Figure 3.1, Gender and Income are conditionally independent given Education.

- **Strong conditional independence.** If there is no path from $A_i$ to $A_j$, they fall under this category. For example, Age and Gender in Figure 3.1 have strong conditional independence.

**Definition 3** (Bayesian network)**.** A Bayesian network $\mathcal{N}$ is a collection of $d$ attribute-parent set pairs $(A_1, \Pi_1), (A_2, \Pi_2), ..., (A_d, \Pi_d)$ such that each $A_i \in \mathcal{A}$ and $\Pi_i$ is the parent set of $A_i$. By definition, $\Pi_i$ is a subset of attributes: $\Pi_i \subseteq \mathcal{A} \setminus \{A_i\}$. To ensure that $\mathcal{N}$ is acyclic, it holds that for any $1 \leq i < j \leq d$, we have: $A_j \notin \Pi_i$. The *degree* $k$ of a Bayesian network $\mathcal{N}$ is defined as the maximum size of any parent set in $\mathcal{N}$.

For the Bayesian network illustrated in Figure 3.1, the pairs of attributes and parent sets are shown in the rightmost table. For attribute Age, its parent set is $\emptyset$. For attribute Education, its parent set is {Age, Gender}. The degree of the Bayesian network in Figure 3.1 is $k = 2$.

Next, we briefly describe how the collection of attribute - parent set pairs in a Bayesian network $\mathcal{N}$ approximates the joint probability distribution over all attributes $\Pr[\mathcal{A}]$:

$$\begin{aligned}\Pr_{\mathcal{N}}[\mathcal{A}] &= \Pr[A_1, A_2, \ldots, A_d] \\ &= \Pr[A_1] \cdot \Pr[A_2 \mid A_1] \ldots \Pr[A_d \mid A_1, \ldots, A_{d-1}] \\ &= \prod_{i=1}^{d} \Pr[A_i \mid \Pi_i]\end{aligned}$$

The first step follows from the chain rule, and the second step follows from the conditional independence assumption of Bayesian networks. In other words, if an attribute $A_j$ is indeed conditionally independent of attributes that are not within its parent set, then $\Pr[A_j \mid A_1, \ldots, A_{j-1}]$ can be simplified and written as $\Pr[A_j \mid \Pi_j]$. This enables $\Pr_{\mathcal{N}}[\mathcal{A}]$ to be a good approximation of $\Pr[\mathcal{A}]$ and brings an important benefit: when $k$ is small, $\Pr[A_j \mid \Pi_j]$ is a low-dimensional distribution. Learning $d$ low-dimensional distributions $\Pr[A_1 \mid \Pi_1]$, $\Pr[A_2 \mid \Pi_2]$, ..., $\Pr[A_d \mid \Pi_d]$ is easier and more accurate than learning high-dimensional distributions such as $\Pr[A_d \mid A_1, \ldots, A_{d-1}]$. Consequently, $\Pr_{\mathcal{N}}[\mathcal{A}]$ of a Bayesian network can provide an efficient approximation of $\Pr[\mathcal{A}]$ especially when $d$ is large and/or $k$ is small.

# Chapter 4

# LEARNING BAYESIAN NETWORKS UNDER LDP

In this chapter, we describe our two solution methods for learning Bayesian networks under LDP: LDP-BN and LDP-BN+. First, we explain the fundamental Bayesian network construction algorithm used in both LDP-BN and LDP-BN+. This algorithm is formulated in a way such that privatizing the mutual information computation step in it enables us to achieve LDP. Second, we describe our LDP-BN solution. Third, motivated by the utility loss of LDP-BN, we develop and explain our LDP-BN+ solution, which yields utility improvements while still satisfying LDP.

## *4.1 Bayesian Network Construction*

Recall that for a Bayesian network $\mathcal{N}$, it is desirable for $\Pr_{\mathcal{N}}[\mathcal{A}]$ to be a good approximation of $\Pr[\mathcal{A}]$. Kullback-Leibler divergence (KL-divergence) $D_{KL}$ can be used to quantify the distance between $\Pr_{\mathcal{N}}[\mathcal{A}]$ and $\Pr[\mathcal{A}]$, which is defined as:

$$D_{KL}(\Pr[\mathcal{A}]\|\Pr_{\mathcal{N}}[\mathcal{A}]) = -\sum_{i=1}^{d} I(A_i, \Pi_i) + \sum_{i=1}^{d} H(A_i) - H(\mathcal{A}) \tag{4.1}$$

Here, $I(\cdot,\cdot)$ represents mutual information and $H(\cdot)$ represents entropy. For an attribute or set of attributes $S \subseteq \mathcal{A}$ with domain $dom(S)$, the entropy of $S$ is calculated as:

$$H(S) = -\sum_{s \in dom(S)} \Pr[S = s] \cdot \log(\Pr[S = s]) \tag{4.2}$$

For an attribute $A$ and its parent set $\Pi$, let $a \in dom(A)$ and $\pi \in dom(\Pi)$. Let $\Pr[a, \pi]$ denote the joint probability $\Pr[a, \pi] = \Pr[A = a, \Pi = \pi]$, and let $\Pr[a] = \Pr[A = a]$ and $\Pr[\pi] = \Pr[\Pi = \pi]$ denote the marginal probabilities. Then, the mutual information $I(A, \Pi)$ is calculated as:

$$I(A, \Pi) = \sum_{a \in dom(A)} \sum_{\pi \in dom(\Pi)} \Pr[a, \pi] \cdot \log\left(\frac{\Pr[a, \pi]}{\Pr[a] \cdot \Pr[\pi]}\right) \tag{4.3}$$

**Algorithm 1:** Bayesian Network Construction

**Input :** All joint and marginal probabilities, degree $k$, set of attributes $\mathcal{A}$

**Output:** Bayesian network $\mathcal{N}$

```
Initialize 𝒩 and 𝒰 as empty
Choose a random attribute A_s ∈ 𝒜 as the start node
Add (A_s, ∅) to 𝒩
Remove A_s from 𝒜 and add it to 𝒰
while 𝒜 is not empty do
    Initialize Ω as empty list
    for A_i ∈ 𝒜 do
        candidates ← Enumerate all possible subsets of 𝒰 with size ≤ k
        for S ∈ candidates do
            m ← Calculate I(A_i, S) using Equation 4.3
            Insert triplet (A_i, S, m) to Ω
    (A*, S*, m*) ← Find the triplet in Ω that has the highest value of m
    Add (A*, S*) to 𝒩
    Remove A* from 𝒜 and add it to 𝒰
return 𝒩
```

According to Equation 4.1, in order to reduce the distance $D_{KL}(\Pr[\mathcal{A}]\|\Pr_{\mathcal{N}}[\mathcal{A}])$, one should maximize $\sum_{i=1}^{d} I(A_i, \Pi_i)$ and minimize $\sum_{i=1}^{d} H(A_i) - H(\mathcal{A})$. Note that $\sum_{i=1}^{d} H(A_i) - H(\mathcal{A})$ is entirely dependent on the original data distribution since it measures the entropy of individual attributes and the entropy over all attributes. It cannot be changed by altering the Bayesian network $\mathcal{N}$. Yet, when learning the structure of $\mathcal{N}$, we have direct control over which attributes will be included in the parent set $\Pi_i$ of an attribute $A_i$. This enables us to control the mutual information $I(A_i, \Pi_i)$. Hence, in order to achieve the goal of reducing the distance $D_{KL}(\Pr[\mathcal{A}]\|\Pr_{\mathcal{N}}[\mathcal{A}])$, we will aim to construct parent sets that maximize $I(A_i, \Pi_i)$.

Algorithm 1 describes our algorithm which aims to minimize $D_{KL}(\Pr[\mathcal{A}]\|\Pr_{\mathcal{N}}[\mathcal{A}])$. The inputs of Algorithm 1 are the joint and marginal probabilities, the degree $k$ of

the Bayesian network, and the set of attributes $\mathcal{A}$. For now, we assume that all joint probabilities (e.g., $\Pr[a, \pi]$) and marginal probabilities (e.g., $\Pr[a]$) are known by the server. Therefore, Equation 4.3 can be used to compute mutual information. The output of the algorithm is the Bayesian network $\mathcal{N}$, represented as a collection of $d$ attribute-parent set pairs as defined in Def. 3.

Algorithm 1 starts by initializing the Bayesian network $\mathcal{N}$ and the set $\mathcal{U}$ as empty. Here, $\mathcal{U}$ is a set to store the attributes that have already been used (processed) by the algorithm. On line 2, a random attribute is selected as the starting node, and it is added to $\mathcal{N}$ with an empty parent set (line 3). Since the starting node is processed, it is removed from $\mathcal{A}$ and added to $\mathcal{U}$ (line 4). Then, the main loop between lines 5-14 iterates as long as there remains an attribute in $\mathcal{A}$ that has not yet been processed. In each iteration, the remaining attributes $A_i \in \mathcal{A}$ and their candidate parent sets are considered (lines 7-8). Since the Bayesian network has a degree of $k$, the parent sets must be of size at most $k$ (fewer than $k$ attributes in a parent set are allowed, but more than $k$ are not allowed). For each attribute $A_i$ and candidate parent set $S$, the mutual information between them $I(A_i, S)$ is calculated using Equation 4.3 (line 10). A triplet with $(A_i, S, m)$ is constructed and inserted into $\Omega$, where $m$ is the mutual information between $A_i$ and $S$. After all attributes and candidate parent sets are processed, the triplet in $\Omega$ that has the highest mutual information is found (line 12). This triplet, denoted by $(A^*, S^*, m^*)$, is selected to be added to the Bayesian network so that $D_{KL}(\Pr[\mathcal{A}] \| \Pr_{\mathcal{N}}[\mathcal{A}])$ will be minimized. After the addition to the Bayesian network, the added attribute is removed from $\mathcal{A}$ and added to $\mathcal{U}$ (line 14) since it has now been processed. When all attributes are processed, then the Bayesian network will contain a node for each attribute and its corresponding parent set. At this point, the main loop between lines 5-14 will terminate, and the resulting $\mathcal{N}$ will be returned (line 15).

An example demonstrating the iteration by iteration execution of Algorithm 1 is given in Figure 4.1. Before the main loop iterates (lines 5-14), one attribute is selected and added to $\mathcal{N}$, say it is the Age attribute. Then, in the first iteration, as shown in Figure 4.1, the Gender attribute is added to $\mathcal{N}$. Both Gender and Age have empty parent sets. By the end of the first iteration, $\mathcal{U} = \{\text{Age, Gender}\}$ and

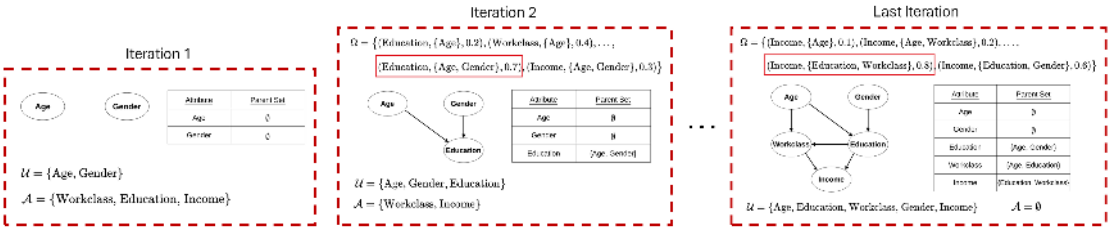


Figure 4.1: Sample iteration by iteration execution of Algorithm 1.

$\mathcal{A}$ = {Workclass, Education, Income}. Then, in the second iteration of the loop, $\Omega$ is constructed as follows. The possible subsets of $\mathcal{U}$ are {Age}, {Gender}, {Age, Gender}, and $\emptyset$. Each attribute Workclass, Education, Income is matched with each of {Age}, {Gender}, {Age, Gender} and $\emptyset$; thereby resulting in 12 triplets that are added to $\Omega$. Say that the mutual information of the triplet (Education, {Age, Gender}, 0.7) is the highest among all triplets in $\Omega$. Then, Education is added to the Bayesian network as a node, and its parent set consists of {Age, Gender}. By the end of this iteration, Education is removed from $\mathcal{A}$ and added to $\mathcal{U}$. The algorithm continues its execution in this manner. Say that when it arrives at the last iteration, the only attribute remaining in $\mathcal{A}$ is $\mathcal{A}$ = {Income}. Then, $\Omega$ is constructed as shown on the rightmost end of Figure 4.1 by matching Income with all subsets of {Age, Gender, Education, Workclass}. Say that the mutual information of the triplet (Income, {Education, Workclass}, 0.8) is the highest. Then, Income is added to the Bayesian network as a node, and its parent set consists of {Education, Workclass}. Since Income was the last attribute in $\mathcal{A}$, removing it on line 14 causes $\mathcal{A}$ to become empty, and therefore Algorithm 1 terminates.

### *4.2 Challenges and Initial Solution Attempt*

In Algorithm 1, it is assumed that all joint and marginal probabilities are known by the server, i.e., the party executing Algorithm 1. However, under LDP, the server does not possess users' records; therefore, these probabilities are unavailable. Instead, users perturb and send their records, and the server must estimate the probabilities from the perturbed records. Here, an initial solution attempt could be

the following:

- Each user $u \in \mathcal{P}$ initializes a bitvector $B_u$ with length equal to $dom(\mathcal{A})$. There is a bijection between the positions of this bitvector and all possible records in $dom(\mathcal{A})$, such that each position $i$ corresponds to a different possible record $r_i \in dom(\mathcal{A})$.

- User $u$ sets $B_u[i] = 1$ if and only if their record satisfies $r_u = r_i$, and $B_u[i] = 0$ otherwise.

- User $u$ perturbs their $B_u$ using the OUE protocol and sends the perturbed bitvector $B'_u$ to the server.

- Upon collecting perturbed bitvectors from all users, the server performs estimation to recover $\bar{C}(i)$ for all $i \in [1, dom(\mathcal{A})]$. The same bijection is used to retrieve the estimated count of every possible record $r_i \in dom(\mathcal{A})$. Using these counts, joint and marginal probabilities are calculated.

However, this approach is infeasible in practice since it requires each user's bitvector to have length equal to $dom(\mathcal{A})$. Consider 10 attributes with $dom(A_i) = 10$ for each attribute. Then, we have $dom(\mathcal{A}) = 10^{10}$ which means the length of each bitvector $B_u$ will be $10^{10}$ and its size will be approximately 1.25 gigabytes. Perturbing a bitvector of length $10^{10}$ by individually processing each bit is time-consuming on the user's device, and the user-side storage space cost and the user-server communication cost of 1.25 gigabytes are prohibitively large. Furthermore, considering that the server will collect perturbed bitvectors from potentially hundreds of thousands of users, the server-side storage cost will be in the order of terabytes or petabytes, exacerbating the problem.

### *4.3 LDP-BN Solution*

To circumvent the above problems and to enable Bayesian network learning under LDP, our LDP-BN solution makes the following key observation: Algorithm 1 uses

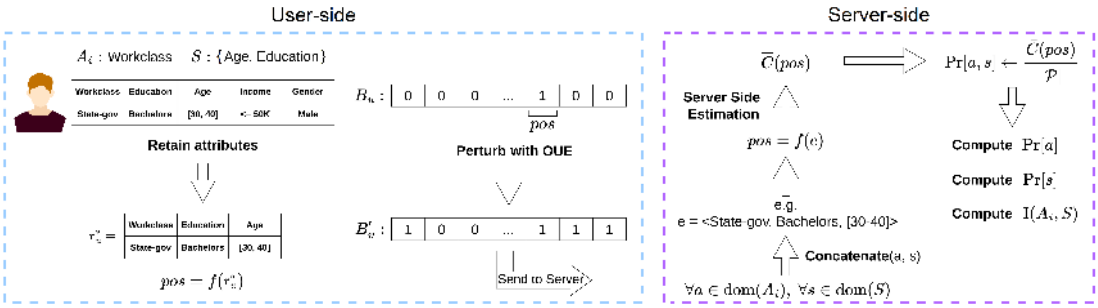


Figure 4.2: Sample step-by-step execution of Algorithm 2.

the joint and marginal probabilities when calculating $I(A_i, S)$ on line 10, but in doing so, it does not need the $d$-dimensional joint probability defined over all attributes $A_1, A_2, ..., A_d$. Instead, the $(k+1)$-dimensional joint probability defined over $A_i \cup S$ is sufficient. Since $k$ is typically much smaller than $d$ (e.g., $d \geq 10$ but $k \leq 3$), collecting $(k+1)$-dimensional joint probabilities using OUE is much easier. For example, assuming attributes with domain sizes equal to 10, for a typical value of $k = 2$, the size of a user's bitvector will be $10^3$, implying less than 1 kilobyte in size. After the joint probability defined over $A_i \cup S$ is collected using OUE, the required probabilities for calculating $I(A_i, S)$ based on Equation 4.3 are $\Pr[a, s]$, $\Pr[a]$ and $\Pr[s]$ for all $a \in dom(A_i)$ and $s \in dom(S)$. Among them, $\Pr[a, s] = \Pr[A_i = a, S = s]$ is directly retrieved from the joint probability defined over $A_i \cup S$. The marginals $\Pr[a]$ and $\Pr[s]$ can be found as:

$$\Pr[a] = \Pr[A_i = a] = \sum_{s \in dom(S)} \Pr[a, s] \tag{4.4}$$

$$\Pr[s] = \Pr[S = s] = \sum_{a \in dom(A_i)} \Pr[a, s] \tag{4.5}$$

Following this intuition, the building block of LDP-BN is shown in Algorithm 2. This algorithm takes as input $A_i$, $S$, and the LDP privacy budget $\hat{\varepsilon}$ that will be consumed when calculating the mutual information $I(A_i, S)$. First, the server computes $dom(A_i \cup S)$ and defines a bijection between each element $e \in dom(A_i \cup S)$ and integers $[1, n]$ so that each element in the domain will be mapped (one-to-one and onto) to integers in $[1, n]$. The server sends this bijection $f$ to all users so that users will use this same $f$ when encoding their records as bitvectors. Then, each user

**Algorithm 2:** Mutual Information under LDP

**Input :** Attribute $A_i$, set of attributes $S$, allocated privacy budget $\hat{\varepsilon}$

**Output:** Mutual information $I(A_i, S)$

`// Server-side initialization`

1 Server computes $dom(A_i \cup S)$ and defines a bijection: $f : e \to [1, n]$ for all $e \in dom(A_i \cup S)$

2 Server sends $f$ to all users

`// User-side encoding and perturbation`

3 **for** *each user* $u \in \mathcal{P}$ **do**

4 Initialize bitvector $B_u = [0, 0, ..., 0]$ with length equal to $|dom(A_i \cup S)|$

5 $r_u^* \leftarrow$ RETAIN_ATTRIBUTES$(r_u, A_i \cup S)$

6 $pos \leftarrow f(r_u^*)$

7 $B_u[pos] \leftarrow 1$

8 $B'_u \leftarrow$ Perturb $B_u$ using the OUE protocol with budget $\hat{\varepsilon}$ (Equation 3.3)

9 User $u$ sends $B'_u$ to the server

`// Server-side estimation and mutual information computation`

10 **for** *each pair* $(a \in dom(A_i), s \in dom(S))$ **do**

11 $e \leftarrow$ CONCATENATE$(a, s)$

12 $pos \leftarrow f(e)$

13 $\bar{C}(pos) \leftarrow$ Use server-side estimation of OUE with budget $\hat{\varepsilon}$ (Equation 3.5)

14 Assign $\Pr[a, s] \leftarrow \dfrac{\bar{C}(pos)}{|\mathcal{P}|}$

15 Use Equation 4.4 to compute $\Pr[a]$ for all $a \in dom(A_i)$

16 Use Equation 4.5 to compute $\Pr[s]$ for all $s \in dom(S)$

17 $I(A_i, S) \leftarrow \sum_{a \in dom(A)} \sum_{s \in dom(S)} \Pr[a, s] \cdot \log\left(\frac{\Pr[a,s]}{\Pr[a] \cdot \Pr[s]}\right)$

18 **return** $I(A_i, S)$

initializes a bitvector $B_u$ (line 4). When encoding their record $r_u$ into the bitvector, the user retains the relevant attributes $A_i \cup S$ and discards the rest (line 5). This

sub-record is fed into the bijection $f$ and the result is the position in $B_u$ that should be set to 1 (lines 6-7). Then, the user perturbs their bitvector using OUE and sends the perturbed bitvector to the server (lines 8-9). After receiving perturbed bitvectors from all users, the server-side estimation and mutual information computation phase begins. In this phase, the server first computes the joint probability $\Pr[a, s]$ for all $a \in dom(A_i)$ and $s \in dom(S)$. To do so, the server needs to find which position *pos* the $(a, s)$ pair corresponds to in the reported bitvectors. To find this, the server concatenates $a$ and $s$ and feeds the result to the bijection $f$ (lines 11-12). After the corresponding position is obtained, the server-side estimation process of the OUE protocol is executed to retrieve the estimated count $\bar{C}(pos)$, and it is divided by the population size $|\mathcal{P}|$ so that the probability $\Pr[a, s]$ is obtained (lines 13-14). After $\Pr[a, s]$ for all $a \in dom(A_i)$ and $s \in dom(S)$ is found, Equations 4.4 and 4.5 are used to compute the marginals $\Pr[a]$ and $\Pr[s]$. Finally, the mutual information $I(A_i, S)$ is computed similarly to Equation 4.3 and the result is returned.

A sample execution of Algorithm 2 is shown in Figure 4.2. Say that $A_i$ = Workclass, $S$ = {Age, Education}, and $f$ has been sent to all users. On the user-side, RETAIN_ATTRIBUTES is executed on the user's record so that in the resulting $r_u^*$, only the Workclass, Age, and Education attributes remain. By applying $f(r_u^*)$, the corresponding position *pos* is obtained. In the user's bitvector $B_u$, the position *pos* is set to 1 whereas the remaining bits are 0. $B_u$ is perturbed with OUE and the resulting bitvector $B'_u$ is sent to the server. The server-side process is repeated for different pairs of $a \in dom(A_i), s \in dom(S)$ (lines 10-14). Say that in the current iteration, $a$ = State-gov and $s$ = <Bachelors, [30,40]>. These $a$ and $s$ are concatenated and fed into $f$, resulting in *pos*. The server-side estimation of OUE is used to obtain $\bar{C}(pos)$, which is then used to obtain $\Pr[a, s]$. After repeating this process for all $a \in dom(A_i), s \in dom(S)$, the probabilities $\Pr[a]$, $\Pr[s]$, as well as the mutual information $I(A_i, S)$ are found.

Overall, in LDP-BN the server uses the procedure given in Algorithm 1, but in order to satisfy $\varepsilon$-LDP, it calculates the necessary $I(A_i, S)$ values using Algorithm 2. This approach satisfies LDP since the only step of Algorithm 1 which involves collecting data from the user population is the calculation of $I(A_i, S)$. However, the

main weakness of LDP-BN is having to execute Algorithm 2 for each different $A_i$ and $S$ pair. In the worst case, this requires $\binom{d}{k+1}$ executions of Algorithm 2. For $\binom{d}{k+1}$ executions of Algorithm 2 to satisfy $\varepsilon$-LDP overall, each execution of Algorithm 2 uses the budget:

$$\hat{\varepsilon} = \frac{\varepsilon}{\binom{d}{k+1}} \tag{4.6}$$

so that $\varepsilon$-LDP is achieved by sequential composition. When $d$ is high, $\binom{d}{k+1}$ yields a large number which means $\hat{\varepsilon}$ becomes very small. Small $\hat{\varepsilon}$ causes the mutual information learned by Algorithm 1 to contain too much noise caused by LDP, which decreases the accuracy of the Bayesian network. LDP-BN+ aims to solve this problem by increasing the budgets used in the calculation of each $I(A_i, S)$.

### *4.4 Density-Aware Covering Designs*

Recall that in order to calculate $I(A_i, S)$, the joint probabilities $\Pr[a, s]$ for all $a \in dom(A_i)$ and $s \in dom(S)$ are needed. To increase the LDP budget used in the estimation of each joint probability, we formulate the following problem. We divide the given set of all attributes $\mathcal{A}$ into potentially overlapping groups $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m)$ such that:

- Every possible $A_i$ and $S$ pair must co-exist in at least one $\mathcal{G}$ so that the joint probability $\Pr[a, s]$ can be estimated for every possible pair of $A_i$ and $S$.
- The total number of groups should be kept as low as possible so that the $\varepsilon$ budget will be divided into the fewest number of pieces.
- For each group $\mathcal{G}$, the Cartesian product of the attributes in that group $dom(\mathcal{G})$ must have sufficiently small size so that the use of LDP to collect users' records remains computationally feasible.

We use the concept of *covering designs* to tackle this problem. Covering designs are combinatorial structures used to ensure that all possible combinations of certain elements are included in a collection of groups [Bluskov and Hämäläinen, 1998,

Gordon et al., 1995]. The primary goal of a covering design is to "cover" all required combinations with the minimal number of groups.

**Definition 4** (Covering design)**.** Let $\alpha$ denote a set of elements, $\beta$ denote the number of groups, and $\gamma$ denote the sizes of combinations that need to be covered. The covering design $Cov(\alpha, \beta, \gamma)$ consists of a collection of $\beta$-sized groups of the elements in $\alpha$, such that every $\gamma$-sized subset of $\alpha$ is contained in at least one group.

Given $\alpha$, $\beta$, and $\gamma$, our algorithm for finding an appropriate covering design $Cov(\alpha, \beta, \gamma)$ is given in Algorithm 3. Since every $\gamma$-sized subset of $\alpha$ must be contained in at least one group, all possible combinations of $\binom{\alpha}{\gamma}$ are generated and all of them are initially marked as uncovered (i.e., not contained in at least one group). The candidate groups are generated as combinations of $\binom{\alpha}{\beta}$. Then, the main loop of Algorithm 3 (lines 4-16) iterates as long as there is still at least one uncovered subset. In each iteration, the group with maximum coverage (i.e., maximum number of subsets it contains among remaining uncovered subsets) is found (lines 7-14). This best candidate is added to the list of groups that will be returned by the algorithm (line 15) and the subsets covered by this candidate are removed from the list of uncovered subsets (line 16). After the main loop (lines 4-16) terminates, all subsets are contained in at least one group, and the constructed list of groups is returned.

In LDP-BN+, we use covering designs to construct the groups $\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m$ as:

$$(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m) \leftarrow Cov(\mathcal{A}, \beta, k+1) \tag{4.7}$$

In other words, we enforce that every $(k+1)$-sized subset of $\mathcal{A}$ is contained in at least one group $\mathcal{G}_j$, where $\beta$ denotes the group size. This guarantees that the joint probability $\Pr[a, s]$ will be available for every possible pair of $A_i$ and $S$ in the Bayesian network, since $|A_i \cup S| = k+1$. If a pair of $A_i$ and $S$ appears in more than one group, then $\Pr[a, s]$ can be computed for each group and then averaged.

We exemplify this use of covering designs in Figure 4.3. Consider that we have 5 attributes: $\mathcal{A} = \{$Workclass, Education, Age, Income, Gender$\}$, $k = 1$, and $\beta = 3$. A covering design consisting of 4 groups $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4)$ is shown on the right side of Figure 4.3. It can be observed that all subsets of $\mathcal{A}$ with size = 2 are contained

**Algorithm 3:** Create Covering Design

**Input :** $\alpha$, $\beta$, $\gamma$

**Output:** Resulting groups $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m)$

```
groups ← ∅
uncovered ← all possible combinations of (α choose γ)
candidates ← all possible combinations of (α choose β)
while uncovered ≠ ∅ do
    best_cand ← NULL
    max_covered ← ∅
    for candidate ∈ candidates do
        coverage ← ∅
        for subset ∈ uncovered do
            if subset ⊆ candidate then
                coverage ← coverage ∪ subset
        if |coverage| > |max_covered| then
            max_covered ← coverage
            best_cand ← candidate
    groups ← groups ∪ best_cand
    uncovered ← uncovered \ max_covered
return groups
```

in at least one group $\mathcal{G}_j$. This guarantees that $\Pr[a, s]$ is available for every possible pair of $A_i$ and $S$ in at least one $\mathcal{G}_j$.

In the above formulation, $\beta$ remains as an important parameter which impacts the number of groups and the size of $dom(\mathcal{G}_j)$ for each group. When $\beta$ is large, there are fewer groups but $dom(\mathcal{G}_j)$ is large. This causes both computational feasibility concerns and utility concerns since there will be few users whose records $r_u$ match with each element in $dom(\mathcal{G}_j)$, and LDP noise will dominate the true counts. On the other hand, when $\beta$ is small, $dom(\mathcal{G}_j)$ per group is small, but there are many groups. As a result, the server needs to select a good value for $\beta$ to attain high

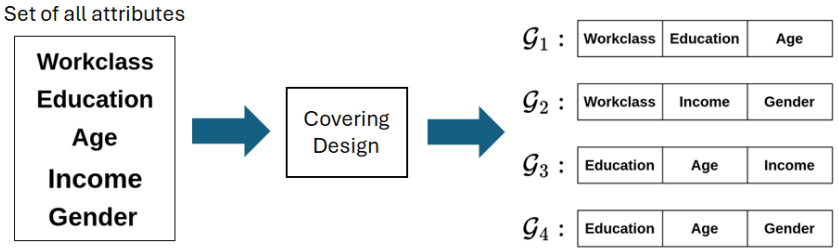


Figure 4.3: Example of creating a covering design for a set of attributes $\mathcal{A}$ and the resulting groups $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4)$.

**Algorithm 4:** Density-Aware Covering Design

**Input :** Set of attributes $\mathcal{A}$, degree $k$, population $\mathcal{P}$, density threshold $\tau$

**Output:** Resulting groups $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m)$

Initialize $\beta \leftarrow |\mathcal{A}|$
**while** $\beta \geq k + 1$ **do**
  $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m) \leftarrow$ call Alg. 3 with $\mathcal{A}$, $\beta$, $k + 1$
  $satisfied \leftarrow$ TRUE
  **for** *each* $\mathcal{G}_j \in (\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m)$ **do**
    $density \leftarrow \frac{|\mathcal{P}|}{|dom(\mathcal{G}_j)|}$
    **if** $density < \tau$ **then**
      $satisfied \leftarrow$ FALSE
      **break**
  **if** $satisfied =$ TRUE **then**
    **return** $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m)$
  **else**
    $\beta \leftarrow \beta - 1$
**return** $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m)$

utility, i.e., $\beta$ should be large enough to ensure there are few groups, but also small enough to ensure $dom(\mathcal{G}_j)$ will be sufficiently small.

To address this challenge, we propose the density-aware covering design approach

given in Algorithm 4. It takes as input the set of attributes $\mathcal{A}$, the degree $k$ of the Bayesian network, user population $\mathcal{P}$, and a density threshold parameter $\tau$. It utilizes a *density-aware* approach to determine $\beta$ internally as follows. To ensure $\beta$ is large, it is initialized as $\beta \leftarrow |\mathcal{A}|$. Then, $\beta$ is decreased by 1 in each iteration. Within each iteration, Algorithm 3 is used to calculate the covering design $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m)$ with the current $\beta$. If $\frac{|\mathcal{P}|}{|dom(\mathcal{G}_j)|} \geq \tau$ for all $\mathcal{G}_j \in (\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m)$, then the domains are considered to be sufficiently small. However, if there is a group $\mathcal{G}_j$ such that the density $\frac{|\mathcal{P}|}{|dom(\mathcal{G}_j)|}$ is less than $\tau$, then the density constraint is not satisfied (lines 7-9). If the density constraint is satisfied, then the current $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m)$ can be returned (line 11); otherwise, $\beta$ will keep decreasing until it hits $k+1$, which is the smallest possible value. Overall, this algorithm ensures that the resulting groups $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m)$ are sufficiently few by starting with high $\beta$, but also the domains of the groups are sufficiently small by iteratively decreasing $\beta$ until the density threshold with parameter $\tau$ is satisfied.

### *4.5 LDP-BN+ Solution*

Finally, our LDP-BN+ solution for learning Bayesian networks under LDP, which combines the previously explained methods and algorithms, is provided in Algorithm 5. The algorithm starts with the server executing the density-aware covering design algorithm (Algorithm 4) to obtain the groups $\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m$. Then, the server divides the user population $\mathcal{P}$ randomly into $m$ non-overlapping subpopulations, such that each subpopulation $\mathcal{P}_j$ will contribute to $\mathcal{G}_j$. We do this so that each user contributes exactly once with the full budget $\varepsilon$, rather than dividing $\varepsilon$ into $m$ pieces. This strategy yields fewer contributing users per $\mathcal{G}_j$ but each contribution has higher utility since the full $\varepsilon$ budget is used. The strategy of dividing users rather than budgets was shown to yield a favorable tradeoff in prior works [Wang et al., 2017, Tire and Gursoy, 2024, Yang et al., 2020]. It is possible to utilize this strategy effectively in LDP-BN+ but not LDP-BN since the number of subpopulations in LDP-BN+ is $m$ whereas it would be $\binom{d}{k+1}$ in LDP-BN. The latter is oftentimes too large, yielding extremely few users per subpopulation.

For each $\mathcal{G}_j$, the server executes a process similar to Algorithm 2 for collecting data under LDP (lines 4-12). A bijection $f$ is defined for elements in $dom(\mathcal{G}_j)$ and sent to users in $\mathcal{P}_j$; the users create their bitvectors using their records $r_u$ and $f$; users perturb their bitvectors using OUE; and the perturbed bitvectors are sent to the server. Then, for each $A_i$ and $S$ pair that are included in $\mathcal{G}_j$, the server estimates the marginal probabilities $\Pr[a]$, $\Pr[s]$, the joint probability $\Pr[a, s]$, and the mutual information $I(A_i, S)$. The estimation here re-uses the estimation procedure from Algorithm 2. Results of the estimation are stored in the server-side memory (line 16) since they will be necessary for learning the Bayesian network $\mathcal{N}$. Once the main loop of Algorithm 5 terminates (lines 3-16), since $\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m$ is a covering design, all necessary probabilities and mutual information values for building the Bayesian network will have been calculated and stored in the server's memory. On line 17, the Bayesian network learning process from Algorithm 1 is executed using the stored probabilities and mutual information values, and the resulting Bayesian network $\mathcal{N}$ is returned.

**Algorithm 5:** LDP-BN+ Solution

**Input :** Set of attributes $\mathcal{A}$, degree $k$, population $\mathcal{P}$, privacy budget $\varepsilon$, density threshold $\tau$

**Output:** Bayesian network $\mathcal{N}$

`// Server-side initialization`

1 $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m) \leftarrow$ call Alg. 4 with $\mathcal{A}$, $k$, $\mathcal{P}$, $\tau$

2 $(\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_m) \leftarrow$ divide $\mathcal{P}$ uniformly at random into $m$ non-overlapping subpopulations

`// Collect distributions from users`

3 **for** *each* $\mathcal{G}_j \in (\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m)$ **do**

4 Server computes $dom(\mathcal{G}_j)$ and defines a bijection: $f : e \rightarrow [1, n]$ for all $e \in dom(\mathcal{G}_j)$

5 Server sends $f$ to all users in $\mathcal{P}_j$

`// User-side encoding and perturbation`

6 **for** *each user* $u \in \mathcal{P}_j$ **do**

7 Initialize bitvector $B_u = [0, 0, ..., 0]$ with length equal to $|dom(\mathcal{G}_j)|$

8 $r_u^* \leftarrow$ RETAIN_ATTRIBUTES$(r_u, \mathcal{G}_j)$

9 $pos \leftarrow f(r_u^*)$

10 $B_u[pos] \leftarrow 1$

11 $B'_u \leftarrow$ Perturb $B_u$ using the OUE protocol with budget $\varepsilon$ (Equation 3.3)

12 User $u$ sends $B'_u$ to the server

`// Server-side estimation`

13 Let $A_i$ denote an attribute in $\mathcal{G}_j$ and $S$ be a set of attributes $S \subseteq \mathcal{G}_j \setminus \{A_i\}$ with $|S| \leq k$

14 **for** *each possible* $A_i$ *and* $S$ **do**

15 Execute lines 10-17 of Algorithm 2

16 Store resulting $\Pr[a, s]$, $\Pr[a]$, $\Pr[s]$, and $I(A_i, S)$ in memory

17 Execute lines 1-14 of Algorithm 1 using the probabilities and mutual information stored above

18 **return** $\mathcal{N}$

# Chapter 5

# EXPERIMENTAL EVALUATION

We implemented all algorithms and performed experimental evaluations in Python. Our experiments aim to measure the effectiveness of our solutions under various conditions such as different datasets, $\varepsilon$ budgets, and $k$. We first describe our experiment setup, and then present and discuss our results.

## *5.1 Experiment Setup*

**Datasets.** We used three popular benchmark datasets in our experiments: Adult, ACS Income, and Titanic.

*Adult* is a widely used dataset in the data privacy literature. It can be downloaded from the UCI ML Repository[1] and contains demographic and employment-related information about individuals. We removed records that had missing values and used the remaining 45.222 records. Among the 14 attributes that originally exist in Adult, we removed fnlwgt, education-num, capital-gain, and capital-loss attributes, resulting in a total of 10 remaining attributes.

*ACS Income* is similar to the Adult dataset, but it is more recent [Ding et al., 2021] and contains more than 1.6 million records. It was compiled from the American Community Survey (ACS) Public Use Microdata Sample. Both the Adult and ACS Income datasets are typically used in classification tasks to predict whether an individual earns more than $50K salary in a year.

*Titanic* is a synthetic dataset containing records of 1 million passengers. It contains attributes such as the passenger's age, gender, fare, port of embarkation, and ticket class. The dataset is typically used in classification tasks to predict whether a passenger survived the Titanic disaster. From all records, we extracted

[1] `https://archive.ics.uci.edu/dataset/2/adult`

those passengers with ages between 18 and 99, and removed records with missing values.

**Utility Metrics.** Let $\mathcal{N}_g$ denote the gold standard, non-private Bayesian network that would be built without privacy protection. Let $\mathcal{N}_p$ denote a Bayesian network built under $\varepsilon$-LDP using LDP-BN or LDP-BN+. It is desired that $\mathcal{N}_p$ closely resembles $\mathcal{N}_g$ from a utility perspective. However, in reality, $\mathcal{N}_p$ and $\mathcal{N}_g$ may contain differences caused by LDP noise. To measure these differences, we propose and utilize three utility metrics: Structural Hamming Distance, Joint Distribution Error, and Marginal Distribution Error. For all three metrics, higher errors indicate higher utility loss.

*Structural Hamming Distance (SHD)*: Recall that Bayesian networks can be represented as directed acyclic graphs. SHD quantifies the structural differences between two directed acyclic graphs by measuring the minimum number of edit operations required to transform one graph into the other. More formally, SHD is equal to the minimum number of edge additions, edge removals, and/or edge direction reversals necessary to turn $\mathcal{N}_p$ into $\mathcal{N}_g$.

*Joint Distribution Error (JDE):* JDE enables us to quantify how well the joint probability distributions among attributes are preserved. For all pairs of attributes in $\mathcal{A}$, we generate their joint probability distributions from $\mathcal{N}_g$ and store them in a list denoted by $\mathbf{J}_g$. Then, we generate their joint probability distributions from $\mathcal{N}_p$ and store them in a list denoted by $\mathbf{J}_p$. To measure JDE, we calculate the average $\ell_2$ distances between the distributions in $\mathbf{J}_g$ and $\mathbf{J}_p$:

$$\text{JDE} = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{J}_g[i] - \mathbf{J}_p[i]\|_2 \tag{5.1}$$

*Marginal Distribution Error (MDE):* MDE quantifies how well the marginal probabilities of attributes are preserved. For each attribute $A_i \in \mathcal{A}$, we generate its marginal probability distribution $\Pr[A_i]$ from $\mathcal{N}_g$ and store it in a list denoted by $\mathbf{M}_g$. Then, we generate its marginal probability distribution from $\mathcal{N}_p$ and store it in a list denoted by $\mathbf{M}_p$. To measure MDE, we calculate the average $\ell_2$ distances

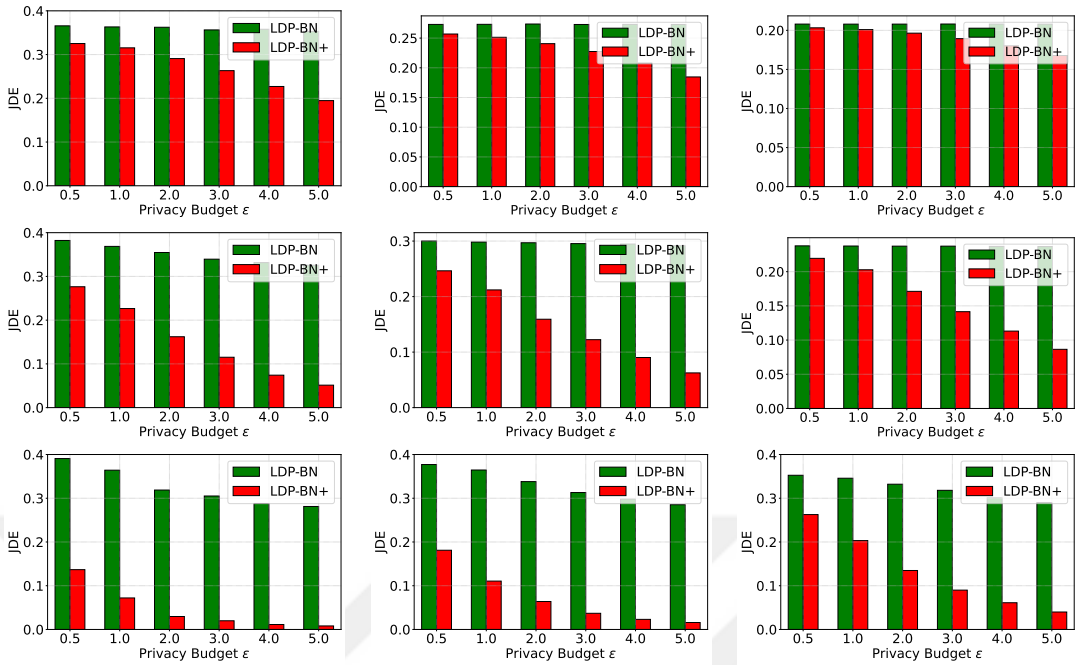


Figure 5.1: JDE results for the Adult (first row), ACS Income (second row), and Titanic (third row) datasets. The leftmost column corresponds to $k = 1$, middle column is $k = 2$, and rightmost column is $k = 3$.

between the distributions in $\mathbf{M}_g$ and $\mathbf{M}_p$:

$$\text{MDE} = \frac{1}{d} \sum_{A_i \in \mathcal{A}} \|\mathbf{M}_g[i] - \mathbf{M}_p[i]\|_2 \tag{5.2}$$

### *5.2 Results and Discussion*

We report results with the JDE metric in Figure 5.1 and the MDE metric in Figure 5.2. In both figures, we experiment with all three datasets, $k$ ranging from 1 to 3, and $\varepsilon$ ranging from 0.5 to 5.0. We observe from Figure 5.1 that LDP-BN+ significantly outperforms LDP-BN across varying datasets, $k$ values, and $\varepsilon$ values. We also observe that as $\varepsilon$ increases, JDE values of LDP-BN+ decrease, which is intuitive because the amount of LDP noise is lower. However, JDE values of LDP-BN do not decrease as much. The reason behind this trend is that LDP-BN divides the total $\varepsilon$ budget into $\binom{d}{k+1}$ pieces; therefore, despite an increase in $\varepsilon$ the individual pieces are not that strongly affected. Hence, in the case of LDP-BN, the amount of LDP noise remains high despite increasing $\varepsilon$.

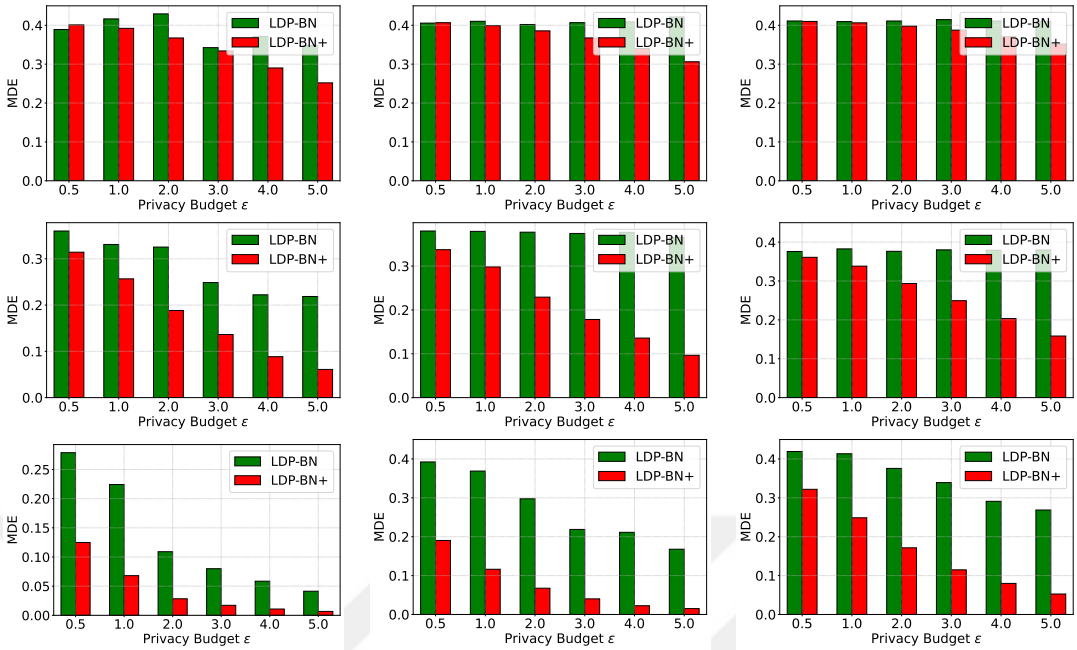


Figure 5.2: MDE results for the Adult (first row), ACS Income (second row), and Titanic (third row) datasets. The leftmost column corresponds to $k = 1$, middle column is $k = 2$, and rightmost column is $k = 3$.

We observe from Figure 5.2 that LDP-BN+ outperforms LDP-BN in terms of the MDE metric as well. On the other hand, there are a few cases with the Adult dataset and low $\varepsilon$ values (such as $\varepsilon = 0.5$, 1.0) in which LDP-BN is similar to LDP-BN+ or slightly better. The reason behind this trend is because of the size of the Adult dataset. In general, LDP performs better when there are large volumes of data since larger datasets enable LDP noise to better cancel out during estimation. Yet, the Adult dataset is considerably smaller than the ACS Income and Titanic datasets. Hence, the noise better cancels out on the ACS Income and Titanic datasets, causing both JDE and MDE values to be lower on these two datasets.

Overall, JDE and MDE results are reasonably low under popularly used $\varepsilon$ values such as $\varepsilon = 1$, 2, and 4 especially in the case of larger datasets. This shows that LDP-BN+ can indeed be used in practice to build Bayesian networks with high utility (low error). On the other hand, we observe that errors are lower when $k = 1$ or 2 compared to $k = 3$. This shows that there is still room for improvement in reducing errors in high-degree Bayesian networks.

Table 5.1: SHD results for different values of $k$ and $\varepsilon$ across all three datasets (Adult, ACS Income, Titanic). The better-performing solution under each setting is highlighted using bold text and gray cell background.

| **Dataset** | $k$ | **Solution** | $\varepsilon = 0.5$ | $\varepsilon = 1.0$ | $\varepsilon = 2.0$ | $\varepsilon = 3.0$ | $\varepsilon = 4.0$ | $\varepsilon = 5.0$ |
|---|---|---|---|---|---|---|---|---|
| Adult | $k = 1$ | LDP-BN | 14 | 16 | **14** | 14 | 14 | 14 |
| | | LDP-BN+ | **10** | **8** | **14** | **12** | **12** | **4** |
| | $k = 2$ | LDP-BN | **18** | **16** | **18** | 18 | 20 | 20 |
| | | LDP-BN+ | 20 | 18 | 20 | **16** | **18** | **18** |
| | $k = 3$ | LDP-BN | 26 | **18** | **16** | 16 | 16 | 28 |
| | | LDP-BN+ | **20** | **18** | **16** | **10** | **14** | **20** |
| ACS Income | $k = 1$ | LDP-BN | **8** | **8** | 10 | **8** | 8 | **10** |
| | | LDP-BN+ | **8** | **8** | **8** | **8** | **6** | **10** |
| | $k = 2$ | LDP-BN | **10** | **8** | **12** | 12 | 16 | 14 |
| | | LDP-BN+ | 12 | 12 | **12** | **10** | **10** | **12** |
| | $k = 3$ | LDP-BN | **6** | 10 | **10** | **10** | **8** | 12 |
| | | LDP-BN+ | 8 | **6** | **10** | **10** | **8** | **10** |
| Titanic | $k = 1$ | LDP-BN | **6** | **4** | 8 | 8 | 8 | 6 |
| | | LDP-BN+ | **6** | **4** | **2** | **4** | **4** | **0** |
| | $k = 2$ | LDP-BN | **6** | **8** | **4** | **6** | 9 | **8** |
| | | LDP-BN+ | 10 | **8** | 6 | 12 | **8** | **8** |
| | $k = 3$ | LDP-BN | 10 | **7** | 12 | 10 | 10 | **8** |
| | | LDP-BN+ | **9** | 8 | **7** | **8** | **8** | **8** |

We report results with the SHD metric in Table 5.1. SHD values tend to decrease as $\varepsilon$ increases, which is intuitive. Yet, in several circumstances, SHD values may remain similar despite increasing $\varepsilon$. Thus, unlike JDE and MDE metrics, there is not a strong correlation between $\varepsilon$ and the amount of error in terms of the SHD metric. Table 5.1 also shows that LDP-BN+ generally provides equal or better SHD compared to LDP-BN across a range of conditions. This observation is similar to the JDE and MDE metrics. In many cases, especially in the Adult and ACS Income datasets, LDP-BN+ clearly outperforms LDP-BN. However, the degree of improvement differs by dataset. The Adult dataset sees consistent improvement from LDP-BN+, whereas ACS Income shows moderate to strong improvements at certain $\varepsilon$ values, whereas Titanic results are more mixed. Across multiple datasets, it should be noted that the benefit of LDP-BN+ is more pronounced when $\varepsilon$ is higher.

### *5.3 Case Study: Synthetic Data Generation with LDP-BN+*

While Bayesian networks can be used for several purposes, one of their applications – especially in the centralized DP domain – has been synthetic data generation [Zhang et al., 2017, Bao et al., 2021, Ma et al., 2023, Qi et al., 2020]. Therefore, we also evaluate LDP-BN+ in a case study involving synthetic data generation. We follow the same steps as in [Zhang et al., 2017] and [Bao et al., 2021] for generating synthetic data. We order the attributes in the Bayesian network in topological order and iterate over them to ensure that the values of the parent attributes are sampled before their children. For the initial attribute (without any parents), the value for that attribute is sampled according to the attribute's marginal distribution. For subsequent attributes, we use the conditional probability distributions calculated from the Bayesian network together with the values of the parents.

After generating synthetic datasets with cardinality equal to $|\mathcal{P}|$, we train two machine learning models on these synthetic datasets: XGBoost and LightGBM. We choose XGBoost and LightGBM models since they generally perform accurately on tabular datasets, and they perform accurately on our three datasets as well. We compare the accuracy of the XGBoost and LightGBM models trained on synthetic data versus models trained on real data. We say that the synthetic datasets have high utility if the models trained on synthetic data have similar accuracy compared to models trained on real data.

Table 5.2 contains the results of this experiment. For each dataset, we observe that the models trained on the real data consistently achieve the highest accuracy. This sets the benchmark for evaluating the synthetic datasets. The results indicate a clear trend where lower values of $\varepsilon$ (corresponding to higher privacy) result in lower model accuracy, which is expected as more noise is introduced to achieve LDP. As the value of $\varepsilon$ increases, the accuracy of models trained on synthetic data generally improves. For example, on the Adult dataset with $k = 1$, XGBoost's accuracy increases from 60.58% at $\varepsilon = 0.5$ to 80.57% at $\varepsilon = 5$, approaching the clean data accuracy of 84.25%. We observe that the parameter $k$, which controls the complexity of the generated Bayesian network, also has an impact on model accuracy. In many

Table 5.2: Accuracy results of XGBoost and LightGBM models built on real data (rightmost column) versus synthetic data generated from Bayesian networks under varying $k$ and $\varepsilon$.

| **Dataset** | $k$ | **Model** | $\varepsilon = 0.5$ | $\varepsilon = 1.0$ | $\varepsilon = 2.0$ | $\varepsilon = 3.0$ | $\varepsilon = 4.0$ | $\varepsilon = 5.0$ | **Real Data** |
|---|---|---|---|---|---|---|---|---|---|
| Adult | $k = 1$ | XGBoost | 60.58 | 67.90 | 70.88 | 67.89 | 71.75 | 80.57 | 84.25 |
| | | LightGBM | 60.42 | 68.01 | 71.11 | 68.77 | 71.65 | 80.69 | 84.09 |
| | $k = 2$ | XGBoost | 72.45 | 72.21 | 72.91 | 70.71 | 72.07 | 69.18 | 84.25 |
| | | LightGBM | 72.25 | 72.38 | 72.76 | 70.84 | 72.17 | 70.02 | 84.09 |
| | $k = 3$ | XGBoost | 78.99 | 82.64 | 70.67 | 74.52 | 72.21 | 70.07 | 84.25 |
| | | LightGBM | 79.82 | 83.15 | 71.47 | 74.68 | 73.58 | 70.90 | 84.09 |
| ACS Income | $k = 1$ | XGBoost | 64.66 | 67.59 | 73.26 | 68.84 | 67.51 | 66.47 | 81.71 |
| | | LightGBM | 64.73 | 67.65 | 73.28 | 68.83 | 67.53 | 66.48 | 81.64 |
| | $k = 2$ | XGBoost | 76.76 | 74.62 | 73.06 | 72.15 | 72.63 | 74.32 | 81.71 |
| | | LightGBM | 76.61 | 74.71 | 73.14 | 72.24 | 72.70 | 74.29 | 81.64 |
| | $k = 3$ | XGBoost | 74.62 | 75.13 | 75.13 | 78.94 | 79.70 | 76.10 | 81.71 |
| | | LightGBM | 74.97 | 75.34 | 75.23 | 79.27 | 79.68 | 76.30 | 81.64 |
| Titanic | $k = 1$ | XGBoost | 80.86 | 80.76 | 81.17 | 80.98 | 80.86 | 81.45 | 85.16 |
| | | LightGBM | 80.89 | 80.76 | 81.17 | 80.98 | 80.86 | 81.45 | 85.14 |
| | $k = 2$ | XGBoost | 79.15 | 76.53 | 79.50 | 79.47 | 80.81 | 79.87 | 85.16 |
| | | LightGBM | 79.13 | 76.45 | 79.55 | 79.49 | 80.77 | 79.89 | 85.14 |
| | $k = 3$ | XGBoost | 79.14 | 74.42 | 84.59 | 71.19 | 71.51 | 80.69 | 85.16 |
| | | LightGBM | 71.78 | 69.36 | 80.09 | 68.33 | 68.79 | 80.62 | 85.14 |

cases, increasing the complexity by using larger $k$ seems to be beneficial for increased model accuracy. Yet, this trend is not always consistent, as there are also several instances in which $k = 3$ yields models with lower accuracy compared to $k = 1$ or 2. This is likely because high-dimensional joint and conditional probabilities become too dominated by LDP noise, which increases utility loss.

Finally, we observe that the XGBoost and LightGBM models exhibit similar performance trends across all datasets and parameter settings. LightGBM only slightly outperforms XGBoost in several scenarios. This consistency between LightGBM and XGBoost suggests that the observed trends are likely to be robust across various models and they are not specific to XGBoost or LightGBM.

# Chapter 6

# CONCLUSION

In this thesis, we addressed the problem of learning Bayesian networks under LDP by proposing two solution methods: LDP-BN and LDP-BN+. We first introduced a common Bayesian network construction algorithm and then adapted its mutual information computations to the LDP setting, yielding the LDP-BN solution. Motivated by the utility limitations of LDP-BN due to the need to divide the privacy budget across multiple mutual information computations, we proposed LDP-BN+ which employs a novel density-aware covering design algorithm, enabling more effective use of the privacy budget. Our experimental evaluations on three datasets and multiple utility metrics demonstrated that LDP-BN+ substantially outperforms LDP-BN, and yields high-utility Bayesian networks.

There are multiple avenues for future work. First, we plan to further optimize the budget allocation by using uneven budgets or uneven subpopulation sizes for different groups $(\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_m)$. Second, we plan to improve the utility and efficiency of Bayesian networks for higher $k$ values. Third, we plan to extend our solutions to temporal and dynamic Bayesian networks to address evolving and time-dependent datasets.

# BIBLIOGRAPHY

[app, 2020] (2020). Apple – learning with privacy at scale. `https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf`.

[Bao et al., 2021] Bao, E., Xiao, X., Zhao, J., Zhang, D., and Ding, B. (2021). Synthetic data generation with differential privacy via bayesian networks. *Journal of Privacy and Confidentiality*.

[Binkyte et al., 2024] Binkyte, R., Pinzón, C. A., Lestyán, S., Jung, K., Arcolezi, H. H., and Palamidessi, C. (2024). Causal discovery under local privacy. In *Causal Learning and Reasoning*, pages 325–383. PMLR.

[Bluskov and Hämäläinen, 1998] Bluskov, I. and Hämäläinen, H. (1998). New upper bounds on the minimum size of covering designs. *Journal of Combinatorial Designs*, 6(1):21–41.

[Chen and Pollino, 2012] Chen, S. H. and Pollino, C. A. (2012). Good practice in bayesian network modelling. *Environmental Modelling & Software*, 37:134–145.

[Cheng et al., 2019] Cheng, X., Tang, P., Su, S., Chen, R., Wu, Z., and Zhu, B. (2019). Multi-party high-dimensional data publishing under differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1557–1571.

[Cormode et al., 2018] Cormode, G., Jha, S., Kulkarni, T., Li, N., Srivastava, D., and Wang, T. (2018). Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1655–1658. ACM.

[Cormode et al., 2021] Cormode, G., Maddock, S., and Maple, C. (2021). Frequency estimation under local differential privacy. *Proceedings of the VLDB Endowment*, 14(11):2046–2058.

[de Benedetti et al., 2020] de Benedetti, J., Oues, N., Wang, Z., Myles, P., and Tucker, A. (2020). Practical lessons from generating synthetic healthcare data with bayesian networks. In *Workshops of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2020)*, pages 38–47. Springer.

[Ding et al., 2017] Ding, B., Kulkarni, J., and Yekhanin, S. (2017). Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3571–3580.

[Ding et al., 2021] Ding, F., Hardt, M., Miller, J., and Schmidt, L. (2021). Retiring adult: New datasets for fair machine learning. *Advances in Neural Information Processing Systems*, 34:6478–6490.

[Du et al., 2023] Du, Y., Hu, Y., Zhang, Z., Fang, Z., Chen, L., Zheng, B., and Gao, Y. (2023). Ldptrace: Locally differentially private trajectory synthesis. *Proceedings of the VLDB Endowment*, 16(8):1897–1909.

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

[Erlingsson et al., 2014] Erlingsson, Ú., Pihur, V., and Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1054–1067. ACM.

[Gordon et al., 1995] Gordon, D. M., Patashnik, O., and Kuperberg, G. (1995). New constructions for covering designs. *Journal of Combinatorial Designs*, 3(4):269–284.

[Guner and Gursoy, 2023] Guner, E. and Gursoy, M. E. (2023). Learning markov chain models from sequential data under local differential privacy. In *European Symposium on Research in Computer Security*, pages 359–379. Springer.

[Gursoy et al., 2022] Gursoy, M. E., Liu, L., Chow, K.-H., Truex, S., and Wei, W. (2022). An adversarial approach to protocol analysis and selection in local differential privacy. *IEEE Transactions on Information Forensics and Security*, 17:1785–1799.

[Hittmeir et al., 2022] Hittmeir, M., Mayer, R., and Ekelhart, A. (2022). Efficient bayesian network construction for increased privacy on synthetic data. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 5721–5730. IEEE.

[Ju et al., 2020] Ju, C., Gu, Q., Wu, G., and Zhang, S. (2020). Local differential privacy protection of high-dimensional perceptual data by the refined bayes network. *Sensors*, 20(9):2516.

[Kaur et al., 2021] Kaur, D., Sobiesk, M., Patil, S., Liu, J., Bhagat, P., Gupta, A., and Markuzon, N. (2021). Application of bayesian networks to generate synthetic health data. *Journal of the American Medical Informatics Association*, 28(4):801–811.

[Liu et al., 2023] Liu, G., Tang, P., Hu, C., Jin, C., Guo, S., Stoyanovich, J., Teubner, J., Mamoulis, N., Pitoura, E., and Mühlig, J. (2023). Multi-dimensional data publishing with local differential privacy. In *EDBT*, pages 183–194.

[Ma et al., 2022] Ma, P., Ji, Z., Pang, Q., and Wang, S. (2022). Noleaks: Differentially private causal discovery under functional causal model. *IEEE Transactions on Information Forensics and Security*, 17:2324–2338.

[Ma et al., 2023] Ma, X., Qi, X., Meng, Y., and Yang, T. (2023). Improved bayesian network differential privacy data-releasing method based on junction tree. In *47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 759–764. IEEE.

[Ohnishi and Awan, 2023] Ohnishi, Y. and Awan, J. (2023). Locally private causal inference. *arXiv preprint arXiv:2301.01616*.

[Qi et al., 2020] Qi, X., Ma, X., Bai, X., and Li, W. (2020). Differential privacy preserving data publishing based on bayesian network. In *19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1718–1726. IEEE.

[Tire and Gursoy, 2024] Tire, E. and Gursoy, M. E. (2024). Answering spatial density queries under local differential privacy. *IEEE Internet of Things Journal*.

[Wang et al., 2017] Wang, T., Blocki, J., Li, N., and Jha, S. (2017). Locally differentially private protocols for frequency estimation. In *Proc. of the 26th USENIX Security Symposium*, pages 729–745.

[Wang et al., 2018] Wang, T., Li, N., and Jha, S. (2018). Locally differentially private frequent itemset mining. In *IEEE Symposium on Security and Privacy (SP)*. IEEE.

[Wang et al., 2024] Wang, Y., Zhang, Z., Qian, H., Gao, Y., and Wang, Q. (2024). A high-dimensional temporal data publishing method based on dynamic bayesian networks and differential privacy. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[Xue et al., 2019] Xue, Q., Zhu, Y., and Wang, J. (2019). Joint distribution estimation and naïve bayes classification under local differential privacy. *IEEE Transactions on Emerging Topics in Computing*, 9(4):2053–2063.

[Yang et al., 2020] Yang, J., Wang, T., Li, N., Cheng, X., and Su, S. (2020). Answering multi-dimensional range queries under local differential privacy. *Proceedings of the VLDB Endowment*, 14(3):378–390.

[Yang et al., 2023] Yang, M., Guo, T., Zhu, T., Tjuawinata, I., Zhao, J., and Lam,

K.-Y. (2023). Local differential privacy and its applications: A comprehensive survey. *Computer Standards & Interfaces*, page 103827.

[Ye et al., 2019] Ye, Q., Hu, H., Meng, X., and Zheng, H. (2019). Privkv: Key-value data collection with local differential privacy. In *IEEE Symposium on Security and Privacy (SP)*, pages 317–331. IEEE.

[Yilmaz et al., 2020] Yilmaz, E., Al-Rubaie, M., and Chang, J. M. (2020). Naive bayes classification under local differential privacy. In *7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 709–718. IEEE.

[Zhang et al., 2017] Zhang, J., Cormode, G., Procopiuc, C. M., Srivastava, D., and Xiao, X. (2017). Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42(4):1–41.