Course Code: Doctor of Philosophy

PhD Thesis: Computer Science

May 2026

Leveraging Arithmetic Circuits for Differentially Private and Cost-Efficient Synthetic Data Generation

Sandeep Suresh

School of Computer Science

Faculty of Engineering and Information Technology

University of Technology Sydney

NSW - 2007, Australia

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Sandeep Suresh* declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the *School of Computer Science, Faculty of Engineering and Information Technology* at the University of Technology Sydney, Australia.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship doi.org/10.82133/C42F-K220.

SIGNATURE: Production Note: Signature removed prior to publication.

Sandeep Suresh

DATE: 6th May, 2026

PLACE: Sydney, Australia

ABSTRACT

Synthetic data generation involves creating artificial data that captures the important statistics of real-world data. Synthetic data can be used as substitutes for highly sensitive datasets. The Lumos database that represents the health system in New South Wales, Australia is an example of such a sensitive dataset. It is a statewide linked database that connects general practice (GP) electronic records with records from emergency, hospital and other health services. Another example of highly sensitive data would be individual credit card transaction histories spanning multiple years. One particular synthetic data generation approach involves constructing a generating process that emits data, where the generating process is designed to match the real-world reference dataset. A Probabilistic Graphical Model (PGM) can be used for such a data generating process. PGMs are a way of encoding probabilistic distributions in a structured graphical form. The synthetic data generation approach used in this thesis involves constructing complex probabilistic models to capture the important statistics of some reference dataset.

Differential Privacy (DP) is a mathematical framework that is used to introduce privacy guarantees to synthetic datasets. DP works by adding controlled noise to the data in order to reduce the probability of leaking private or sensitive information. DP can introduce significant computational challenges. To understand this, consider a schema called a cross-table. Cross-tables for a collection of random variables are tabular formats used to summarize the count of the number of occurrences of different combinations of values of the random variables. Adding noise to these cross-tables to mask individual data combinations can present significant computational challenges. The primary difficulty arises from zero-frequency elements: while adding random noise to observed data is straightforward, data that never appears (revealing nonexistent combinations) must be managed carefully to maintain privacy guarantees. This is not merely a theoretical concern but a practical challenge, as the computational cost of noise addition scales with the number of possible data combinations. In large datasets, there can be so many combinations that the time and resources needed to handle them become impractical. Motivated by the need to reduce this computational cost, one contribution of this thesis is a novel decomposition scheme for Laplacian DP that restructures the noise perturbation process. The novel decomposition scheme is evaluated in terms accuracy, privacy gain and noise generation time. The proposed method reduces the time required for noise injection by a factor of three, while maintaining the same accuracy as a standard method without the decomposition scheme.

After creating a generative model (PGM), synthetic data may be generated by sampling the model. Sampling methods where each sample is generated independently of others

(i.e. independent sampling) are preferred for synthetic data generation. This is because dependent sampling methods can introduce additional statistical relationships beyond that which is in the model. These additional statistical relationships make it difficult to determine whether a relationship apparent from the data is the effect of the model or the sampling algorithm. Independent samplers will not have this problem. But independent sampling is computationally costly because each sample is generated from scratch without using information from previous samples. Therefore I was motivated to develop efficient methods for drawing large numbers of independent samples from conditioned probability distributions. In general, probabilistic models can be made more tractable by compiling them into an arithmetic circuit, which represents a function as a directed acyclic graph of basic arithmetic operations. The next contribution in this thesis is the development and evaluation of two novel methods to draw independent samples from a PGM, where the PGM is compiled to an arithmetic circuit. The methods are a variation of inverse transform sampling. They reduce computational cost and improve efficiency by decomposing the cumulative probability distribution into a small number of terms where each term can be easily computed from the arithmetic circuit. The methods were evaluated in terms of sampling time and accuracy by comparing them against other prominent dependent and independent sampling methods. Within the context of independent sampling, the proposed samplers remained the most computationally efficient with no statistically significant difference in accuracy. The reduction in computational cost is because the proposed methods directly sample from an arithmetic circuit instead of the PGM.

The final contribution of this thesis is the construction of a novel cost-efficient and differentially private synthetic data generator that maintains referential integrity. Referential integrity guarantees that relationships between tables are valid by ensuring that a foreign key value in one table always points to an existing primary key in a table. The proposed synthetic data generator constructs a complex probabilistic model to ensure referential integrity. It does this by creating a hierarchy of simpler probabilistic models with each model representing entities with relationships of definable fixed or random cardinalities. An entity represents a real-world object or concept that needs to be modeled for generating synthetic data. Parent child entity relationships represent 1-to-many relationship between their records. Every record of an entity is conditioned on both the previous record and its parent context set. The proposed synthetic data generator uses the aforementioned novel sampling method and the novel decomposition scheme for DP. It was evaluated against two state-of-the-art generators using multi-table relational datasets as workloads. Two metrics were considered for this evaluation: (1) data generation time and (2) referential integrity. The generation time is measured for DP values ranging from 0.1 to 1. The proposed synthetic data generator had the best data generation time and maintains referential integrity.

DEDICATION

This thesis is dedicated to my parents, whose unwavering support has been the foundation for all my achievements...

ACKNOWLEDGMENTS

First and foremost, I would like to thank my principal supervisor Dr. Bin Liang whose guidance was indispensable for the completion of my research. I'm extremely grateful for his support and patience throughout my research journey. In addition, I would like to thank my co-supervisor Prof. Barry Drake for his counsel and informative advice, especially relating to probabilistic graphical models, arithmetic circuits, sampling, differential privacy and synthetic data generation. I would also like to express my gratitude to my other co-supervisor Prof. Massimo Piccardi for his advice and support on navigating my PhD journey.

I would like to acknowledge a debt of gratitude to my colleagues and co-authors, including Dr. Kun Yu, Associate Prof. Bo Liu and Dr. Guangsheng Zhang. I would also like to thank my colleagues at Digital Health CRC and NSW Health for their support. Finally I would like to express my heartfelt gratitude to my parents for their unwavering support and love.

LIST OF PUBLICATIONS

RELATED TO THE THESIS

1. S. SURESH AND B. DRAKE, *Sampling of large probabilistic graphical models using arithmetic circuits*, in AI 2024: Advances in Artificial Intelligence, Singapore, 2025, Springer Nature Singapore, pp. 174–187

2. S. SURESH, G. ZHANG, B. LIU, AND B. DRAKE, *Cost-efficient and privacy-preserving synthesis of complex sensitive data*, Proceedings of the 2025 Australasian Computer Science Week, (2025), pp. 30–39

NOT RELATED TO THE THESIS

1. M. POWN, S. SURESH, AND B. LAKSHMI, *Investigation of homo and hetero-junction double-gate tunnel-fet-based adiabatic inverter circuits*, IETE journal of research, 69 (2023), pp. 1124–1132

TABLE OF CONTENTS

List of Publications **xi**

RELATED TO THE THESIS . . . xi

NOT RELATED TO THE THESIS . . . xi

List of Figures **xvii**

List of Tables **xix**

1 Introduction **1**

1.1 Introduction . . . 1

1.2 Research Questions . . . 6

1.3 Key Contributions . . . 8

1.4 Overview of Thesis . . . 9

2 Background and Literature Review **11**

2.1 Background . . . 11

2.1.1 Probabilistic Graphical Models . . . 12

2.1.2 Inferencing and Learning . . . 18

2.1.3 Common types of probabilistic queries . . . 19

2.1.4 Parameter Learning and Structure Learning . . . 21

2.1.5 Tractable Inference in PGMs: Knowledge Compilation . . . 23

2.2 Sampling of Probabilistic Graphical Models . . . 33

2.2.1 Forward Sampling . . . 34

2.2.2 Rejection Sampling . . . 34

2.2.3 Markov Chain Monte Carlo (MCMC) Sampling . . . 35

2.2.4 Inverse Transform Sampling . . . 36

2.2.5 Sampling for synthetic data generation . . . 36

2.3 Privacy Preservation of Sensitive Data . . . 37

2.3.1 Differential Privacy . . . 37
2.3.2 Laplace algorithm . . . 38
2.3.3 Approximate Differential Privacy . . . 39
2.3.4 Gaussian algorithm . . . 40
2.3.5 Properties of Differential Privacy . . . 40
2.3.6 Exponential algorithm . . . 42
2.4 Synthetic Data Generation Techniques . . . 43
2.4.1 DataSynthesizer (DS) . . . 44
2.4.2 Synthetic Data Vault (SDV) . . . 45
2.4.3 Synthetic Data Generation Using Graphical Models . . . 48
2.4.4 Synthetic Data Generation Using GANs . . . 49
2.4.5 More Synthetic Data Generators . . . 54
2.4.6 Evaluation metrics . . . 55
2.5 Conclusion . . . 57

3 Sampling Of Large Probabilistic Graphical Models Using Arithmetic Circuits 59
3.1 Overview . . . 59
3.2 Proposal . . . 62
3.2.1 Theoretical Framing . . . 62
3.2.2 The Method . . . 63
3.2.3 Marginals Method Variation . . . 64
3.2.4 Theoretical support for circuit based sampling . . . 64
3.3 Evaluation . . . 66
3.3.1 Workloads . . . 67
3.3.2 Samplers . . . 67
3.3.3 Measures . . . 68
3.4 Results . . . 69
3.4.1 Sampling Time . . . 69
3.4.2 Accuracy . . . 71
3.5 Conclusion . . . 72

4 Novel Decomposition Scheme for Laplace Noise Perturbation to Enhance Differential Privacy 73
4.1 Overview . . . 73
4.2 Related Work . . . 75
4.3 Proposal . . . 77

4.3.1 Laplace Noise Addition . . . 79
4.3.2 Suppression. . . . 79
4.3.3 Reducing Computational Cost via Decomposition . . . 80
4.4 Evaluation . . . 83
4.4.1 Experimental Settings . . . 83
4.5 Results . . . 84
4.5.1 Utility . . . 85
4.5.2 Privacy . . . 86
4.5.3 Computational Cost . . . 88
4.6 Conclusions . . . 89

5 Cost-efficient and Differentially Private Synthetic Data Generation with Referential Integrity using Probabilistic Graphical Models 91
5.1 Overview . . . 91
5.2 Related Work . . . 95
5.3 Proposal . . . 97
5.4 Implementation Details . . . 100
5.5 Privacy composition across linked tables . . . 102
5.6 Evaluation . . . 103
5.6.1 Workloads . . . 103
5.6.2 Baselines . . . 104
5.6.3 Measures . . . 104
5.6.4 Relational Schema for 'Financial' workload . . . 106
5.7 Results . . . 108
5.7.1 Synthetic Data Generation Time . . . 108
5.7.2 Referential Integrity . . . 110
5.8 Conclusions . . . 111

6 Discussion & Conclusion 113

A Appendix A - Results Raw Data 121

B Appendix B - Additional Figures 127

Bibliography 131

LIST OF FIGURES

FIGURE **Page**

2.1 Markov process with 4 states 14

2.2 Single coin 2-state Markov model with two states 'heads' and 'tails' 16

2.3 2-coin 2-state Markov model with each state corresponding to a coin 16

2.4 3-coin 3-state Markov model with each state corresponding to a coin 17

3.1 Accuracy for all 8 samplers 72

4.1 A demonstration of the noise injection processes for privacy preservation. 78

4.2 The probabilistic graphical model used for the Texas dataset. Random variables are shown as circles. Crosstabs are shown as squares. 84

4.3 Graph showing the accuracy of the proposed and standard method. The highest, lowest and median histogram-intersection (HI) values are shown. 85

4.4 The comparison of the privacy gain (PG) results of linkability game between several generation models and the proposed method. 86

4.5 Total time taken for all crosstabs to generate noise for a range of epsilon values. 88

5.1 Diagram of the synthetic data generation process 93

5.2 Example entity relationships consisting of a single tree 94

5.3 **Financial Dataset**: This structure represents the original schema from the CTU relational repository. Only the Primary Key (PK) and Foreign Key (FK) are shown. 106

5.4 **Financial Dataset**: This structure was used for proposed generator. Not all columns in the tables of the dataset are shown. Only the Primary Key (PK) and Foreign Key (FK) are shown. 107

5.5 The synthetic data generation time results of the proposed generator vs DataSynthesizer for 'Restbase', 'Walmart' and 'Countries' datasets. 109

B.1 **Restbase Dataset**: This structure was used for *Synthetic Data Vault* generator. Not all columns in the tables of the dataset are shown. Only the Primary Key (PK) and Foreign Key (FK) are shown. 128
B.2 **Restbase Dataset**: This structure was used for proposed generator. Not all columns in the tables of the dataset are shown. Only the Primary Key (PK) and Foreign Key (FK) are shown. 128
B.3 **Walmart Dataset**: This structure was used for both proposed and *Synthetic Data Vault* generator. Not all columns in the tables of the dataset are shown. Only the Primary Key (PK) and Foreign Key (FK) are shown. 129
B.4 **Countries Dataset**: This structure was used for *Synthetic Data Vault* generator. Not all columns in the tables of the dataset are shown. Only the Primary Key (PK) and Foreign Key (FK) are shown. 129
B.5 **Countries Dataset**: This structure was used for proposed generator. Not all columns in the tables of the dataset are shown. Only the Primary Key (PK) and Foreign Key (FK) are shown. 130

LIST OF TABLES

TABLE **Page**

3.1 Workload Bayesian Networks and their sizes. 66

3.2 Sampler startup time (in milliseconds) for small Bayesian networks. 70

3.3 Sampler startup time (in milliseconds) for medium, large and very large networks. 70

3.4 Time per sample (in milliseconds) for small Bayesian networks. The best two sampling times are underlined. 71

3.5 Time per sample (in milliseconds) for medium, large and very large Bayesian networks. The best two sampling times for each workload are underlined. A dash (-) indicates that time per sample was greater than 1 second. 71

4.1 Example of cross-tab. 77

4.2 Four different cases of the combination of initial and subsequent weight. 80

5.1 Relational datasets and their sizes. 103

5.2 Proposed generator and DataSynthesizer synthetic data generation time for 'Financial' dataset. 108

5.3 Proposed generator and Synthetic Data Vault (SDV) synthetic data generation time for all datasets. 110

A.1 Raw accuracy values for all samplers (runs 1–5) used for evaluation in chapter 3. Data from figure 3.1 121

A.2 Raw accuracy values for all samplers (runs 6–9) used for evaluation in chapter 3. Data from figure 3.1 121

A.3 Accuracy of the proposed method (introduced in chapter 4) for laplace noise decomposition in differential privacy. The highest, lowest, mean and median histogram-intersection (HI) values are shown. Data used in figure 4.3 122

A.4 Accuracy of the standard method for differential privacy (used for comparing against proposed method in chapter 4). The highest, lowest, mean and median histogram-intersection (HI) values are shown. Data used in figure 4.3 123
A.5 Total time taken for all crosstabs to generate noise for a selected range of epsilon values (both the standard method and proposed method from chapter 4). Data used in figure 4.5 124
A.6 'Restbase' dataset: Synthetic data generation time for this workload for both proposed generator and DataSynthesizer (from chapter 5). Data used in figure 5.5 . 125
A.7 'Walmart' dataset: Synthetic data generation time for this workload for both proposed generator and DataSynthesizer (from chapter 5). Data used in figure 5.5 . 125
A.8 'Countries' dataset: Synthetic data generation time for this workload for both proposed generator and DataSynthesizer (from chapter 5). Data used in figure 5.5126

CHAPTER 1

INTRODUCTION

1.1 Introduction

Synthetic datasets can be used as substitutes for highly sensitive datasets in critical domains like healthcare, finance and law enforcement. They could also be used in areas like software testing, product development, teaching, product evaluation, demonstration and marketing. These datasets would need to represent complex relationships across time, objects or concepts. Synthetic data plays a crucial role in a wide variety of industries for several reasons including privacy protection, data scarcity mitigation and development of machine learning algorithms. It can come in a variety of forms including tabular data, image data, text data, video data and audio data.

In the finance sector, synthetic data generators have potential applications in fraud detection, credit risk modeling, market simulation and regulatory compliance testing. For instance, sensitive credit card transaction records can be used for fraud detection. The 'Credit Card Fraud Detection' dataset from Kaggle is a well-known dataset in this domain that contains anonymised credit card transactions made by European cardholders [68]. It serves as a suitable example to understand the challenges involved in generating synthetic versions of such data. The goal here is to capture statistical relationships in this data including feature dependencies, temporal patterns, and the links between features and the fraud label.

The 'Credit Card Fraud Detection' dataset [68] is fully structured tabular data that consists of individual financial transaction records. Each row consists of various numerical attributes that represent a single credit card transaction and a fraud label. The order of trans-

actions is recorded by including a timestamp for every transaction. In order to generate synthetic data from this dataset, the ability to model sequential behaviors per user is important (e.g. burst of fraud attempts on the same card in a short interval - a common fraud scenario). The extreme rarity of fraud cases means that minority class distributions must be carefully modeled. The model would also need to reproduce the realistic correlations present in the various attributes of this high-dimensional data. The extensive use of this dataset in academic research and practical machine learning applications demonstrates the importance of synthetic data generation in the finance sector. [116] [32] [31] [15] [16] [75] [17] [159].

Synthetic data generation has important use-cases in healthcare as well. Patient journey modeling is a technique used to map and analyze the full experience a patient has within a healthcare system [95] [5] [124]. The components of the Australian healthcare system include primary care, general practice, acute care, the emergency department and other allied health professionals [103]. A patient journey model is a way of studying a patient's traversal through the aforementioned services or components of the system. An individual patient's actual patient journey pathway may differ from recommended best-practice pathway due to various geographic, socioeconomic and demographic factors [162]. Modeling actual pathways often involve capturing long-range temporal dependencies, high dimensionality, noisy events and rare events. Synthetic data generation is necessary in handling sensitive datasets used for patient journey modeling. The Lumos dataset is one example where synthetic data generation would be useful [102].

Lumos is a statewide linked database representing the health system in New South Wales, Australia. It connects general practice (GP) electronic records with records from emergency, hospital and other health services [27]. Since this database contains patients' actual pathways, it can be used for comprehensive patient journey analysis across various healthcare settings in order to identify gaps and inefficiencies in care. Lumos can capture important relationships among variables and events in a patient's journey because it includes linked data from all the aforementioned healthcare settings. All the health records for an individual patient are linked with a central key. Typically, there is a natural one-to-many relationship between a patient and their healthcare events. For instance, a single patient can have multiple consultations at different healthcare settings. Capturing this one-to-many relationship during the synthetic data generation process is important for many applications. The synthetic data generation process may also need to capture the interdependencies between different variables (e.g. patient's age and comorbidities are closely tied to their healthcare utilization).

At a top level, there are two approaches to generate synthetic data [152]. One is taking records from a reference database and perturbing them, e.g., changing values and randomizing record elements. The other approach is to construct a generating process that emits data, where the generating process is designed to somehow match the reference database. This thesis focuses on the second approach. In particular, this thesis focuses on generating synthetic data from a probabilistic graphical model (PGM). PGMs are a way of encoding probabilistic distributions in a structured graphical form. A Bayesian network (BN) is a kind of such PGM. It defines conditional independency between random variables by its structure, which is a directed acyclic graph (DAG) with random variables as graph nodes. The nodes are connected to each other by directed arcs [40]. Each random variable will have a set of zero or more parent variables. A random variable and its parents are collectively known as a family. Each family of a BN is associated with a potential function which gives the conditional probability over the values of the child variable, conditioned on values of the parent variables [40] [108].

There are difficulties in representing data as a PGM, especially when the original data is a relational database and not just a single table. A big relational database can have complex entity relationships. An entity represents a real-world object or concept that needs to be modelled for generating synthetic data. This could be a person, a product, a transaction, a location, or anything else relevant to the given use case. The entity defines the fields which become the columns of a table. A record refers to an actual instance of an entity stored in a table (i.e. a single row of the table). A PGM model may need to represent complex entity relationships. Multiple independent tables in the relational database may refer to the same entity and the model would need to take that into account.

Referential integrity is an important consideration during synthetic data generation. It refers to the relationships between tables in a relational dataset. It can be understood by examining the relationship between primary keys and foreign keys in a relational dataset. A primary key is an unique identifier field in a table that individually identifies every single record of that table. When a field in one table references a primary key in another table, it is called a foreign key. Referential integrity is a property about relationships between tables which asserts that every foreign key value refers to an existing primary key. Value integrity is another important consideration for synthetic data generation. It means ensuring every field contains only consistent, accurate and valid data according to some defined constraints of the dataset. Preferably a model used for synthetic data generation would be designed in such a way as to maintain the referential integrity and value integrity of the reference dataset.

Privacy is an important consideration while generating synthetic datasets. An important first step toward preserving privacy is anonymisation, but on its own, it does not provide protection against certain kinds of risks. For instance, a bad actor could use an inference attack to deduce sensitive medical information about individuals from an anonymised healthcare dataset. Inference attacks are techniques that exploit indirect hints or correlations in the data to extract sensitive information even when all explicit identifiers have been removed by proper anonymisation [85] [28]. Homogeneity is a data property that arises when data records are too uniform in their sensitive attributes [88] [85]. This property can make it easier to infer sensitive information. Large number of fields (i.e. columns) combined with small number of records (i.e. rows) in a dataset makes each combination of attributes more unique and therefore makes it easier to infer sensitive information. Similarly, highly imbalanced distributions, where some attributes are extremely rare, can be a privacy risk. Uneven distribution of sensitive attributes (e.g., large number of records share a single attribute) is yet another data property that can reduce privacy. These data properties and privacy risks demonstrate the need for a framework that provides strict privacy and accuracy guarantees.

Differential privacy is a mathematical framework that adds controlled noise to the data in order to reduce the probability of leaking private or sensitive information. It is a well-defined framework with strong mathematical guarantees. It limits the effect that the addition or removal of a single database item has on the overall outcome of any analysis done on the dataset [48] [46]. This privacy protection process faces the same dilemma as most other privacy protection methods which is the privacy-utility tradeoff. The basic process is that a sensitive dataset is taken and a sanitised version of that data with privacy guarantees is generated. The sanitised version needs to retain the utility of the data for the given application while providing the aforementioned privacy guarantee[55] [78]. In differential privacy, this privacy-utility tradeoff is due to the noise being added to the probability distribution of the original data.

Differential privacy poses significant computational challenges. To understand these challenges, a schema called cross-table can be introduced. A cross table or crosstab is a way of collecting statistics for PGMs and other models. Cross-tables for a collection of random variables are tabular formats used to summarize the count of the number of occurrences of different combinations of values of the random variables. Adding random noise to cross-tables to mask individual data contributions can present significant computational challenges. The primary difficulty arises from zero-frequency elements: while adding random noise to observed data is straightforward, data that never appears (revealing nonexis-

tent combinations) must be managed carefully to maintain privacy guarantees. This is not merely a theoretical concern but a practical challenge, as the computational cost of noise addition scales with the number of possible data combinations. In large datasets, there can be so many combinations that the time and resources needed to handle them become impractical.

A synthetic data generation process involves designing a generative process that produces data in a way that captures the important statistics of the reference database. No generating process can provide information beyond what is inherent in the model. Therefore privacy guarantees, via noise addition through differential privacy, can focus on the modeling process itself and not the output data. This focus is useful when the generating process is a PGM because privacy preservation techniques can directly focus on the information embedded in the PGM. A generation process based on a PGM can produce potentially unlimited synthetic data without sacrificing privacy. The generating PGMs presented in this thesis are Bayesian networks (BN).

After creating a generative model, synthetic data may be created by sampling the model. Sampling a probabilistic model involves drawing samples from the model, possibly conditioned on assumptions about states of some random variables. Importantly, as the number of samples approaches infinity, the empirical distribution of the samples approaches the distribution defined by the model. This property is guaranteed by design of the sampling method even when the model's joint probability distribution is not directly computable. There are two main groups of sampling methods for probabilistic models, independent and dependent. Independent sampling refers to the process of drawing individual samples from a model in such a way that each sample is generated independently of the others. Methods include forward sampling, rejection sampling and inverse transform sampling [72]. Dependent sampling methods involve drawing a sequence of samples, where adjacent pairs of samples are statistically related. Methods include Metropolis-Hastings [61], Gibbs [18], slice sampling and variations.

Independent sampling methods are computationally costly because each sample is generated from scratch without using information from previous samples. Every sample requires a full evaluation of the target probability density or likelihood, which can be expensive, especially when dealing with high-dimensional data. For instance, consider rejection sampling, a sampling method that generates independent samples from a complex distribution by sampling from an easier proposal distribution and accepting these samples based on a probabilistic criterion. Here if the proposal distribution is not well-matched to the target, most samples are rejected, meaning you perform many costly computations

for few accepted samples. For synthetic data generation this computational cost may be impractically inefficient. Typically, faster dependent sampling methods are not suitable for synthetic data generation because the sampler can introduce additional statistical relationships, beyond that in the model. This makes it difficult to determine whether a relationship apparent from the data is the effect of the model or the sampling algorithm. Here the need for a cost-efficient independent sampling method is apparent.

Probabilistic models can be made more tractable by using knowledge compilation. The idea behind knowledge compilation is to perform a potentially computationally expensive offline pre-processing step to convert the original representation into a more tractable form. The probabilistic model may be compiled into an arithmetic or logic circuit. An arithmetic circuit is a function represented by a directed acyclic graph (DAG) where a directed arc points from nodes to parent nodes. Leaf nodes are function inputs or constants. All other nodes represent arithmetic operations, typically summation and multiplication over the entire set of real numbers, or other semi-ring [40].

1.2 Research Questions

As mentioned previously, creating a cost-efficient synthetic data generator that provides privacy guarantees is an important field of research. The synthetic data generator proposed in this thesis uses probabilistic models (specifically BNs) because the math behind them is well understood. It provides verifiable accuracy guarantees. The proposed generator uses Differential Privacy (DP) to ensure privacy guarantees because the math behind DP privacy guarantees is well understood. Independent sampling (specifically inverse transform sampling) is used because this method guarantees that the relationships and dependencies apparent in the data are restricted to those that are modeled, and not a side-effect of the sampling algorithm.

The proposed synthetic data generator reduces computational cost in two major areas of the synthetic data generation process: the sampling process and the noise perturbation process. The sampling process is made computationally efficient by compiling the PGM into an arithmetic circuit based on the method proposed by Chavira and Darwiche [20]. Two novel sampling methods were introduced to sample from the model. The methods are a form of inverse transform sampling which relies on a cumulative probability distribution. The contribution here is the decomposition of this cumulative probability distribution into a sum of a small number of terms, where each term can be efficiently calculated using the arithmetic circuit. In the next contribution, the noise perturbation process of

differential privacy (DP) was made more computationally efficient. The proposed decomposition scheme for Laplacian DP restructures the perturbation process such that certain cases, where initial weight is zero and subsequent weight after noise addition is less than a suppression threshold, require no processing at all. The suppression threshold is defined by a parameter called "minimum cell size".

An important research area for this thesis is maintaining referential integrity in synthetic data generators that use PGMs. The proposed generator constructs a complex probabilistic model that ensures referential integrity. It does this by creating a hierarchy of simpler probabilistic models with each model representing entities with relationships of definable fixed or random cardinalities. Parent-child entity relationships represents 1-to-many relationship between their records. It also ensures every record of an entity is conditioned, when needed explicitly by the model, on both the previous record and the parent entity. The proposed generator also utilizes the new sampling and differential privacy methods to reduce computational cost. The overall research question for this thesis can be summarised as follows: *How can we generate synthetic data that models complex relationships (across time, objects or concepts) while the generation process ensures computational efficiency, privacy guarantees and accuracy guarantees?* The main research question can be expanded to form three research questions that cover all this and forms the core part of the thesis.

1. How does decomposing cumulative probability distributions via knowledge compilation and arithmetic circuits enhance the efficiency of inverse transform sampling for independent sample generation in probabilistic graphical models? How does its performance compare to dependent and independent sampling methods across varying model sizes?

2. How does decomposing the additive Laplacian noise perturbation process by bypassing computation for zero counts that remain below the minimum desired cell-size threshold affect computational efficiency and privacy gain in enforcing strict ϵ-Differential Privacy on linked tables?

3. How does creating a synthetic data generator that utilizes a hierarchical tree structure to model the data, the aforementioned decomposition scheme to ensure privacy and the aforementioned sampling algorithm to generate the data impact the data generation time and referential integrity? How does this generator compare in these metrics against other publicly available generators on multi-table relational datasets?

1.3 Key Contributions

The first contribution of this thesis is a novel sampling method for efficiently sampling PGMs using arithmetic circuits. The proposed method is a variation of inverse transform sampling which is an independent sampling technique. Inverse transform sampling needs an ordering over all states of the given model. Assume that a point on the cumulative probability distribution $c(x)$ is given by $\Sigma_{y \leq x} P(X = y)$. The inverse transform sampler selects a random variate u uniformly from range $[0, 1]$. It then computes the corresponding joint states x via the inverse cumulative distribution - that is, it selects the x that minimizes $c(x)$ while satisfying $c(x) \geq u$. The contribution here is the decomposition of this cumulative probability distribution into a small number of terms, where each term can be efficiently calculated using the arithmetic circuit. The proposed method is evaluated in terms of sampling time and accuracy against prominent independent and dependent samplers.

The next contribution is a novel decomposition scheme for reducing the computational cost of Laplacian noise perturbation process. In order to enforce strict ϵ-differential privacy, both the zero and non-zero weights of all possible combinations of values must be considered during noise addition. The proposed method restructures the perturbation process such that certain cases, where initial weight is zero and subsequent weight after noise addition is less than the suppression threshold, require no processing at all. The suppression of weights is controlled by a parameter called 'minimum cell size'. The decomposition process treats the zero-cases collectively as a repeated application of noise, followed by a suppression decision. The proposed noise generator is evaluated in terms of accuracy, privacy gain and noise generation time. Noise generation time and accuracy are measured by comparing the proposed method against the standard version without the novel decomposition scheme. Privacy gain is evaluated by comparing the synthetic data generated using this method against prominent state-of-the-art baseline synthetic data generators.

The final contribution of this thesis is the overall construction of a novel synthetic data generator. The proposed generator constructs a complex probabilistic model that ensures referential integrity. It does this by creating a hierarchy of simpler probabilistic models with each model representing entities with relationships of definable fixed or random cardinalities. Parent-child entity relationships represents 1-to-many relationship between their records. It also ensures every record of an entity is conditioned, when needed explicitly by the model, on both the previous record and the parent entity. Both the novel sampling method and the novel decomposition scheme are used in the generator. The proposed generator is evaluated against two state-of-the-art generators using multi-table relational

datasets as workloads. Two metrics were considered for this evaluation: (1) data generation time and (2) referential integrity. The generation time is measured for DP epsilon values ranging from 0.1 to 1.

1.4 Overview of Thesis

Chapter 2 of this thesis covers the background and literature review for my work. In the background section I provide a survey on different PGMs including Bayesian Networks (BN), Markov Models, Hidden Markov Models (HMM) and factor graphs. Bayesian networks and factor graphs are particularly important for my research. A survey of the various inferencing and learning techniques for PGMs are also provided. Finally, the basics of knowledge compilation and arithmetic circuits are also provided. Arithmetic circuits are also important for my research work. The literature review section of Chapter 2 includes a survey of different sampling techniques, different methods for differential privacy and various publicly available state-of-the-art synthetic data generators.

Chapter 3 of this describes two novel sampling algorithms based on inverse transform sampling. The theory behind them is explained in detail. Chapter 3 also evaluates the performance of these samplers in terms of sampling time and accuracy against prominent independent and dependent samplers. Chapter 4 provides the mathematical framework for the new decomposition scheme for Laplacian DP. The evaluation of this new scheme is also provided. Chapter 5 describes the overall construction of the synthetic data generator. A mathematical framework is provided along with stopping conditions that determine when the record generation process stops. Chapter 5 also includes an evaluation of the proposed generator. This is done by comparing it against two state-of-the-art generators in terms of data generation time and referential integrity.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

2.1 Background

The early days of artificial intelligence (AI) and machine learning (ML) mostly comprised of systems based on classical logic. These types of systems could be portrayed as a collection of logical clauses or rules [140]. Their applications include theorem proving, rule-based games (like chess) and other areas that do not deal with uncertainty. But such logic-based systems cannot be used in situations where the system has to manage uncertainty. This is because these systems are monotonic (meaning they cannot alter an already established conclusion even if new facts are known) and modular (meaning each piece of information can be used to reach conclusions independently) [140].

Sophisticated applications of AI like mining and medicine needed an approach that could manage uncertainty. Certain techniques like fuzzy logic and the Dempster-Shafer theory were developed to handle this uncertainty in expert systems. In the early 1980s some systems based on probability theory were first proposed [140]. However, the naive application of probability theory encountered too much computational complexity at the time. The advent of new technologies like Bayesian networks and Markov models which could better represent the structure of the system made it possible to build more sophisticated probability-based models. A probabilistic model defines a joint probability distribution over a set of random variables. Each random variable has a set of possible values. In this thesis, the focus is mainly on discrete random variables where the set of possible values is finite. Although in general continuous random variables are possible. Thus, a probabilis-

tic model defines a probability for every possible combination of values for the random variables, and the sum of all probabilities for each possible combination is one.

2.1.1 Probabilistic Graphical Models

A probabilistic graphical model (PGM) is a probabilistic model for which a graph expresses the conditional dependence structure between random variables. For any given model, the random variables and their possible values remain constant. If they change, then by definition it becomes a new model [108]. The following sub-sections will describe different types of PGM models and the various queries that can be done on them.

2.1.1.1 Bayesian Networks

A Bayesian network defines conditional independency between random variables by its structure, which is a directed acyclic graph (DAG) with random variables as graph nodes [108]. Each node represents a random variable, and the nodes are connected to each other by directed arcs.

An arc from random variable X_1 to X_2 may be thought of as a causal connection, that is, the state of X_1 is a cause for the state of X_2. However, interpretation as a causal relationship is not necessary. An arc from X_1 to X_2 merely indicates that, the given information on the states of X_1 tells us something about the probability distribution over the states of X_2.

The model structure implies that each random variable will have a set of zero or more parent variables. A random variable and its parents are collectively known as a family. Thus, a Bayesian network of n random variables has n families, each family has one child variable and zero or more parent variables.

Each family of a Bayesian network is associated with a potential function which gives the conditional probability over the values of the child variable, conditioned on values of the parent variables. If all the random variables of the Bayesian Network are discrete, then the potential function may be represented as a table, known as a conditional probability table or CPT. Note that the potential function need not be a CPT for other graphical models. Mathematically, it is convenient to think of each potential function as a factor from the combined states of the family to a real number representing the conditional probability.

The probabilistic semantics of a Bayesian network model are defined here. Let X be the random variables of the Bayesian network, and let F be the set of all factors of the Bayesian network. The joint probability represented by the model is defined as

$$P(X = x) = \prod_{f \in F} f(x) \tag{2.1}$$

Here x is a possible world and $f(x)$ is the value of the factor f given random variables in the states implied by x.

The real significance of Bayesian networks can be understood by examining the qualitative nature of dependence in probability. Generally, in mathematics numeric quantities are used to establish whether two variables were dependent. However human intuition is often able to detect certain dependencies, even when numerical estimates are not available. For instance, it is intuitively obvious that two events like 'being the victim of a mugging' and 'asteroid impact on a nearby city' are independent. But providing a numerical estimate for that would be difficult. Such human reasoning-based assumptions of dependence can be encoded into Bayesian networks. These insights can save a lot of computation time for inference engines, which would otherwise have to numerically derive probability values for all possible combinations

2.1.1.2 Markov Models

A Markov process is a process in which the state of the process at a point (in space or time) depends only on the previous state. Markov processes relate to systems that change over time, or space, or other topological set. Markov processes are typically introduced considering changes in time with discrete time steps, as they are the simplest to understand.

The Markov blanket and Markov property are two concepts needed to understand Markov models. The Markov blanket of a random variable, is the set of random variables, that completely describe the probability distribution of that random variable [108]. If the probability distribution of the present state of a process can be determined by the information contained in the previous state, then the process is said to satisfy the Markov property [108]. A model which satisfies the Markov property is called a Markov model. Markov models are used to represent Markov processes. The focus here is on discrete time Markov models with a finite state space.

A simple form of Markov models, called a Markov chain, characterizes flow between states. The process is seen as transitioning from one state to the next over time (or as mentioned, over any topological set). The probability distribution over states of a Markov process at a point in time depends only on the state at the previous point. When it comes to using data from a Markov process (e.g., for machine learning), the state of the process may

not be observable, or may be only partially observable. If a Markov process is fully observable, then the process can be accurately described by a model in which data is available for all random variables. Sometimes authors will call such a model as observable [118], but it should be noted that it is actually the process that is observable rather than the model. In the observable case, the Markov chain is completely defined by a state transition probability distribution matrix, A, and the initial state distribution matrix, π.

Consider the case where a discrete Markov process is represented by an observable Markov model. The theory and derivations for this was organised in a 1989 paper by Rabiner [118]. A discrete Markov process with 4 states as per Rabiner [118] is represented in Figure 2.1.

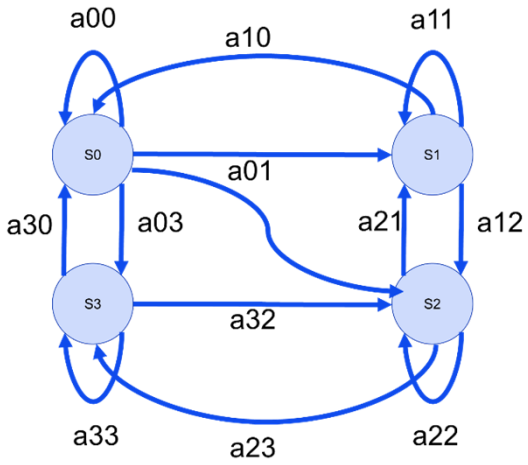


Figure 2.1: Markov process with 4 states

Figure 2.1 represents a Markov chain that undergoes a change of state in discrete steps. During a state change, the system can either stay in the same state or transition to a new one. The arrows represent all possible state changes. If there is no arrow from one particular state to another, then that transition has a zero probability. The set of state transition probabilities are given in the form of $a_{i,j}$.

$$a_{ij} = P\left[q_t = S_j \mid q_{t-1} = S_i\right], \quad 1 \leq i, j \leq N$$

N is the number of states. Here as mentioned $N = 4$.

$$\sum_{j=1}^{N} a_{ij} = 1 \quad \text{(Since they obey standard stochastic constraints)}$$

This describes a standard observable Markov model with completely observable states. As mentioned earlier, the state of the process may not be observable for some processes. Typi-

cally, such processes are represented using a Hidden Markov Model (HMM). HMMs will be covered in the following section.

2.1.1.3 Hidden Markov Models

Hidden Markov Models (HMM) are generally used to represent Markov processes taken to have some hidden state. For example, when a model is being built from data, the data may be seen as 'noisy' when the noise is explainable, perhaps partly, by some hidden state. An HMM may be represented as a double-layered Markov model where the observations (lower layer) are a probabilistic function of the hidden states (upper layer) and the stochastic process that drives the state transitions is hidden [118]. So, the HMM has one hidden variable and one observable variable in each time slice.

Observable Markov models (OMM) work well in situations where the data is clean. However, a lot of complicated use-cases involve extremely noisy data with a lot of errors. Consider the case of speech recognition. Different people pronounce the same word differently or, alternatively even the same person may pronounce the same word differently at different points in time. Similarly, different words can have the same pronunciation. An OMM would only be able to see the probability distribution of pronunciations of the words and not the underlying words (states). Therefore, the OMM would not work well with such noisy data since it cannot create distinct states for each word. On the other hand, an HMM can have the probability distribution of pronunciations which are observable (lower layer) linking to the hidden states (upper layer) representing the actual words. It should be noted that actual speech recognition works at the phoneme level and not at the word level.

To understand HMMs further, consider the case of a person tossing coins [118]. In this example the person has a choice of tossing one of two coins for each toss. Each coin is biased differently. In this scenario, only the result of each coin toss is known; the chosen coin is hidden. Therefore, if the person switches coin, the observer is unaware. This can be modelled as an OMM or HMM in many different ways.

For any given sequence of observations, the goal is to build a Markov model that accurately represents the underlying Markov process that created those observations. In the case of the coin toss experiment, the observations are a sequence of heads and tails. One possible model would a single coin model with two states: heads and tails. This model is an OMM and it can be accurately realised by just knowing the bias of the coin. Figure 2.2 represents this model. Another possible model is the 2-coin model, with each coin corresponding to a state, shown in Figure 2.3. It's obvious that there two separate probability distributions here: the probability distribution of transitioning from one state to another

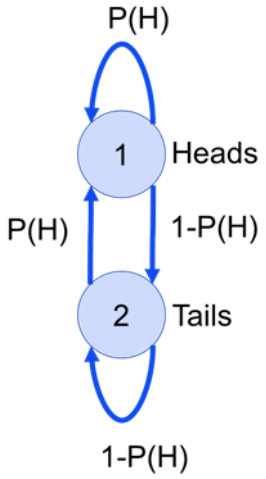


Figure 2.2: Single coin 2-state Markov model with two states 'heads' and 'tails'

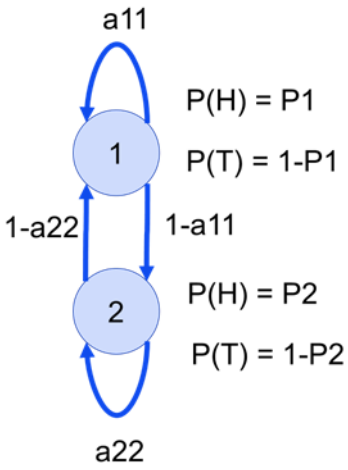


Figure 2.3: 2-coin 2-state Markov model with each state corresponding to a coin

or remaining in the same state at each time step, and the probability distribution of seeing a particular observation given that the model is in a particular state. A third possible explanation (Figure 2.4) is 3-state model with three biased coins. The same event can thus be represented in infinite ways.

Given the three possible models for explaining the observation sequence, the next question is to determine which is better. This can be determined by using a concept called an objective function. An objective function (also called a loss function or cost function) measures the quality of a given model, often with respect to the complexity of the model and some training/evaluation data. If the model is too simple (for instance the 1-coin model), then it may not be able to represent more complicated events. But with more variables

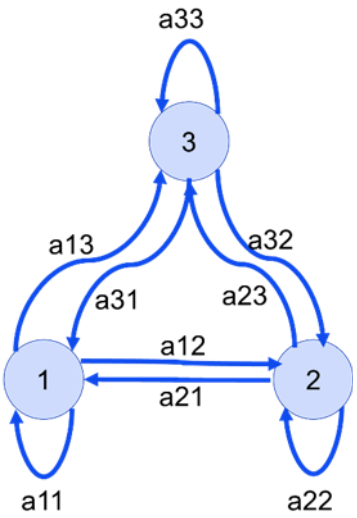


Figure 2.4: 3-coin 3-state Markov model with each state corresponding to a coin

and greater degrees of freedom the model would be able to represent more sophisticated events. However, some practical considerations like computational complexity, places limits on the maximum possible size of the model [118].

2.1.1.4 Factor Graphs

A factor graph is a probabilistic model that generalises many kinds of probabilistic models including Bayesian networks. As with all probabilistic models, a factor graph defines a joint probability distribution over a set of random variables. A factor graph is a "graphical" model in that the graph structure of the model fully captures conditional independencies between random variables [86] [74].

Probabilistic Graphical Models can be directed or undirected. In directed models, a directed edge from one node to the next indicates that the second node is conditioned on the first. As mentioned in the previous sub-section, the Bayesian Network is one example of such a directed model. On the other hand, in an undirected model the edges indicate some sort of dependence between random variables. A Markov Random Field (MRF) is an example of an undirected graphical model. Factor graphs can be used to represent both directed and undirected models [51]. Every Bayesian Network and Hidden Markov Model is a factor graph.

Every factor of a factor graph is associated with one or more random variables of the factor graph. The factors represent functions from the possible states of the random variables to a real number. In a Bayesian Network, the factors are conditional probability tables

(CPTs). But in general, factors represent the weights associated with each possible combination of random variables (i.e., worlds). The weight of a possible world is represented using the function ϕ. The sum of weights over all possible worlds is called the partition function, designated z.

Let $X = \{X_1, X_2, \ldots\}$ represent a collection of discrete random variables. The domain of each random variable, $\mathbb{D}(X_i) = \{x_{i,1}, x_{i,2}, \ldots\}$, is the set of possible states for the random variable. By using $|.|$ to denote set size, the number of possible states for random variable X_i is $|\mathbb{D}(X_i)|$. The distinction between a single random variable and a collection of random variables can be blurred, given that a collection is equivalent to a single random variable with states derived from the Cartesian product of the domains. That is $\mathbb{D}(X) = \mathbb{D}(X_1) \times \mathbb{D}(X_2) \times \ldots$.

Let P be a joint probability over X. $P(X = x)$ is the probability of the random variables X being in state $x \in \mathbb{D}(X)$. Each factor is associated with a subset of random variables. A factor provides a non-negative real number for each combination of possible states of its random variables. A factor graph's set of factors, F, defines a joint probability distribution over its random variables, X.

$$P(X = x) \stackrel{\text{def}}{=} \frac{1}{z}\phi(x) \tag{2.2}$$

$$\phi(x) = \prod_{f \in F} f(x) \tag{2.3}$$

$$z = \sum_{x \in \mathbb{D}(X)} \phi(x) \tag{2.4}$$

where z is the partition function for the distribution.

2.1.2 Inferencing and Learning

Inferencing refers to constructing a reasoning engine that can answer queries about the given PGM. Examples of some queries are given in the following section. Such reasoning engines are necessary because a given PGM may not directly make available probability values for required propositions (within the domain of the model). The purpose of an inference engine is to calculate probability values of propositions and provide joint values for random variables meeting some probabilistic requirement.

There are two broad classes of inferencing algorithms. The first class of algorithms perform exact inference. They guarantee to produce correct answers but may be computationally demanding. The second class of algorithms perform approximate inference. They relax the need to be correct but provide results that may be sufficiently close to a correct answer – this relaxation aims to ease computational requirements

There are three main exact inference algorithms for Bayesian networks: (1) inference by variable elimination, (2) inference by factor elimination, and (3) inference by conditioning [39]. The underlying basis for inference by variable elimination is to reduce the number of variables by summing probabilities over states of random variables which are not relevant to the query at hand, while retaining states that are relevant to answering the query of interest [39]. Inference by factor elimination is usually implemented using the jointree algorithm. This works by eliminating all factors (by summing probabilities over states) except ones containing the variables of interest. The final class of algorithms are inference by conditioning algorithms. Conditioning algorithms either reduce the network into a tree structure which is tractable, or decompose the network into smaller networks which can be solved recursively [39].

Within the class of approximate inference algorithms, belief propagation is a technique used for calculating the marginal distribution of all the unobserved random variables in a Bayesian network. Belief propagation was introduced as an exact inference algorithm on undirected trees, and then expanded to directed trees (or polytrees) as an approximate inference algorithm. While the algorithm is not exact on general graphs, it has been shown to be a useful approximate algorithm [108]. Another approximate inference technique in Bayesian networks is stochastic sampling. Stochastic sampling is a process for measuring the frequency of certain events, in a sequence of situations that are simulated based on their probability of occurrence, in order to estimate the probability of some event (or query). There are many algorithms for sampling a PGM.

The task of constructing a PGM by specifying its structure and/or parameter values from sample data is learning. In order to completely specify a PGM, both its structure and parameter values must be set.

2.1.3 Common types of probabilistic queries

Probabilistic models can be queried for different types of information. Here four types of queries are explained.

1. **Probability of state (Joint Probability)**: In this type of query, the model is asked to

provide the probability of a unique set of variable instantiations [40]. The unique set of random variable instantiations refers to a possible world represented by the combination of random variables. So, this probability represents the degree of belief in the given unique world. The given variables are called evidence variables. This type of query is one of the simplest queries that can be demanded from a probabilistic model.

2. **Prior and Posterior Marginals**: Marginal distribution can be defined as the joint distribution on a subset of the model's random variables involved [40]. If the marginal distribution is calculated with some given evidence then it can be referred to as a posterior marginal. This means that posterior marginal probability is always conditional. On the other hand, if the marginal distribution computation doesn't involve any given evidence, then it becomes a prior marginal [40].

3. **Most Probable Explanation (MPE)**: The next type of query considered here is the Most Probable Explanation (MPE) query. Here, given some evidence, the aim is to discover the most likely instantiation of all the model's random variables [40]. If the random variables are $X_1, X_2, X_3, \ldots X_n$ and the evidence is e, then the goal is to determine:
$$\underset{x_1,x_2,x_3,\ldots,x_n}{\operatorname{argmax}} \; P(X_1 = x_1, X_2 = x_2, X_3 = x_3, \ldots, X_n = x_n \mid e)$$

4. **Maximum a posteriori hypothesis (MAP)**: The Maximum a posteriori hypothesis (MAP) query is a generalisation of the MPE query. When given some evidence, calculating the most probable instantiation of a subset of the model's random variables is considered as an MAP query [40]. If the given evidence is e, then the aim is finding the most probable instantiation m of M network variables. This means maximizing $P(M \mid e)$. If $m = M$, then the MAP query becomes an MPE query. An important note here is that it is easier to algorithmically compute an MPE than a MAP query.

5. **Sensitivity**: A sensitivity query determines how changing a parameter value (in a factor) will change a probability. This includes the related query of how to minimally change parameter values to respect a given probability constraint, or to quantify the magnitude of change to probabilities in response to multiple constraints.

6. **Sampling**: Sampling can also be a direct use of a model (not just a part of approximate inference). For example, a PGM may be used to generate synthetic data for training, marketing or other purposes. Sampling refers to drawing samples from the probability distribution defined by a PGM (possibly conditioned on evidence). The

joint probability distribution from sampling should approach the actual PGM joint probability distribution as the number of samples approaches infinity. Sampling will be explored in greater detail in following sections.

2.1.4 Parameter Learning and Structure Learning

Parameter learning is an optimization technique for learning the distributions of a Probabilistic Graphical Model (PGM). Consider a factor graph that describes a joint probability distribution over a set of random variables. The structure of the factor graph determines what factors are connected to what random variables. The factors provide a value for each possible combination of states of its connected random variables which are known as the parameters. The factor graph is complete when the parameters has their values. Learning the values of these parameters is called parameter learning. In a Bayesian network, this would mean learning the probability distribution for each node given the values of its parent nodes [64]. The parameters to be learned in Hidden Markov Models (HMMs) are the optimal set of state transition and observation symbol probabilities for the output observation sequence [14].

Maximum likelihood estimation is a process for learning the parameter values of a model. It finds values for the parameters of the model such that they maximize the probability of observing the given data set [64]. This is done by maximizing a likelihood function such that the observations given are most probable for the model described. Considering a dataset of instances D and a probabilistic model M which can give probabilities for instances, $P(x \mid M)$, then the likelihood of data is given by:

$$L(D \mid M) = \prod_{x_i \in D} P(x_i \mid M) \tag{2.5}$$

Direct estimation using max likelihood is typically done in Bayesian networks when there are no unobserved variables [64]. Of course, since by definition there are latent variables in HMMs, direct estimation is not possible [14].

In Bayesian networks, if there are unobserved variables an iterative method called expectation maximization (E-M) can be used [64]. This iteration switches between an expectation (E) step and a maximization (M) step. The expectation step works by creating a function for calculating the expected values of latent variables conditioned on the observed data. Then the maximization step calculates the parameters assuming that the previously determined expected values of latent variables are correct [64]. Hidden Markov Models (HMMs) use a special case of the E-M algorithm called the Baum-Welch algorithm.

If the system being modelled contains known logical constraints, then there is no need to expect a learner to learn the constraints. This is because logical constraints translate to zero values for corresponding parameters [154]. Specifically, if a logical constraint implies that a particular combination of states of random variables is impossible, then the probability of that combination is zero, and therefore parameters which relate to that probability are zero [154]. In other words, a logical constraint for one or more random variables is a factor between those variables, where the potential function has values exclusively 0 or 1. For example consider two random variables 'age' and 'driver's license'. The 'driver's license' random variable is conditionally dependent on 'age'. Since the 'driver's license' random variable models whether a person has a license or not, it only has Boolean values. Since someone below the age of 18 is ineligible for a license, the random variable 'driver's license' would always have a zero value for the 0-18 age group. This would be an example of a logical constraint inherent to the network.

A functional factor is one where the state of one of the random variables (the dependant variable) is a function of the states of the other variables (the independent variables) [154]. It is a special case of the logical constraint factor, where for each combination of states of the independent variables, there is exactly one state of the dependant variable for which the potential function value is 1 [154]. An example here would be a random variable 'age' dependent on another random variable 'age-group'. The 'age' random variable has individual states for any age that a human being can have, whereas the 'age-group' random variable has binned groups for different age ranges (e.g. 0-5, 6-12, 12-18, 18-25, ...etc). In all cases, 'age-group' is a function of age. Parameter learning would not need to learn such a functional constraint.

Structure learning is a learning task focused on finding the actual structure of a factor graph. A manual way of learning the structure is for it to be specified by an expert. The synthetic data generator proposed in this thesis relies on the manual approach. But the structure can also be learned by means of an algorithm. When learning the structure of a PGM from data, the learning algorithm typically needs to learn the parameter values as well [40]. Various structure learning algorithms have been proposed over the years [25] [121] [29] [132] [131] [111].

Mutual information is a crucial concept in probability theory necessary for determining the usefulness of a learned structure [40]. Mutual information quantifies the impact of observing one random variable on the uncertainty of another. The mutual information between two random variables X and Y is given by:

$$I_D(X;Y) \stackrel{\text{def}}{=} \sum_{x,y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)} \tag{2.6}$$

2.1.5 Tractable Inference in PGMs: Knowledge Compilation

The computational complexity of an algorithm refers to the amount of resources needed to run it. Time complexity of an algorithm is a measure of how a change in inputs affects the time required for reaching a solution. Similarly, space complexity expresses the memory requirement to run an algorithm. Complexity is always specific to algorithms. There are some problems that can be solved in theory if given sufficiently large but finite time and memory resources. However, in practice the solution for these problems cease to be useful if the resource requirement is too high. Such problems are termed intractable. On the other hand, if the problem can be solved in practice it is termed tractable. The goal of knowledge compilation is to make certain problems more tractable.

2.1.5.1 Basics of Knowledge Compilation

The basics of knowledge compilation can be traced back to some research done in computational intractability of general propositional reasoning by various researchers in the late 1980s and entire 1990s [122] [43] [153] [92] [133] [9] [71] [36]. Some research by Darwiche et al [36] [42] solidified these concepts further.

The idea behind knowledge compilation is to convert graphical models into tractable representations. Essentially the knowledge compilation process takes into account the various features in the structure and parameters of the graph. Certain regularities or patterns in these features are exploited to create a re-representation of the original factor graph. The compilation process may be expensive computationally but if the resulting re-representation is used multiple times, the cost is ameliorated.

Typically, the factor graph is compiled into an arithmetic circuit or logic circuit. An arithmetic circuit is a function represented by a directed acyclic graph (DAG) where a directed arc points from nodes to its parent nodes. Leaf nodes are function inputs or constants. All other nodes represent arithmetic operations, typically summation and multiplication over the entire set of real numbers, or other semi-ring.

The idea of compiling a Bayesian network into an arithmetic circuit originated from a few papers published in the early 2000s [37] [38] [39]. The paper “A differential approach to inference in Bayesian Networks” [39] by Darwiche is particularly important because it

lays the foundation for doing inference using arithmetic circuits. To understand arithmetic circuits properly it's important to consider three main properties:

1. **Decomposability**: Consider a node n that is part of an arithmetic circuit computing a function η. The term vars(n) indicates all the variables in n. All variables are instantiations of X with some indicator variables $\lambda_{i,j} \in \Lambda$ appearing at or under the node. The entire circuit becomes decomposable if and only if every pair of children c_1 and c_2 of all product nodes satisfy the criterion $\text{vars}(c_1) \cap \text{vars}(c_2) = \emptyset$.

2. **Smoothness**: The arithmetic circuit can be called smooth if and only if it has a minimum of one indicator variable $\lambda_{i,j}$ for each random variable $X_i \in X$, and $\text{vars}(n) = \text{vars}(c)$ for every child c of a sum node.

3. **Determinism**: The arithmetic circuit is deterministic if and only if each sum node has a maximum of one non-zero input value while it is evaluated under any particular joint state $x \in \mathbb{D}(X)$.

The original work [38] [37] [39] that proposed the compilation of Bayesian Networks into arithmetic circuits assumed these three circuit properties. Later Poon and Domingos [113] introduced a group of arithmetic circuits that do not necessarily satisfy determinism called Sum-Product Networks (SPN). An SPN is a kind of arithmetic circuit where the nodes (operations) form layers, and the layers alternate between all sum nodes and all product nodes. Vergari et al [157] explains how to visualize and understand SPNs in greater detail. Their aim was to "uncover the inner workings of these probabilistic models" [157]. They introduced an interpretation of SPNs as sparse Multi-Layer Perceptrons whose layers are arranged in a directed acyclic graph.

The details surrounding relaxing determinism and its implications to arithmetic circuits was further formalised in 2017 by Choi and Darwiche [21]. They propose relaxing determinism by allowing sum nodes to be non-deterministic, meaning their children can simultaneously evaluate to non-zero values for the same joint state x. A positive outcome of relaxing determinism, identified by the authors, was that it led to exponentially smaller arithmetic circuits. They also identified a negative implication which resulted in the presence of dead circuit nodes. These are nodes that evaluate to zero for all possible states in $\mathbb{D}(X)$. Understanding this requires properly defining a subcircuit of an arithmetic circuit. A complete subcircuit α of an arithmetic circuit is obtained by traversing the circuit top-down, while choosing one child of each visited sum node and all children of each visited product node. The term of α is the set of states x for which the corresponding indicator

variables $\lambda_{i,j} \in \Lambda$ appear in α. The coefficient of α is the product of all parameters in α. A complete subcircuit consistent with joint state x is called an x-subcircuit. The authors [21] show that all complete subcircuits of a smooth and decomposable arithmetic circuit are x-subcircuits for some instantiation of variables in $\mathbb{D}(X)$.

The standard technique for compiling Bayesian networks into arithmetic circuits limits the allowed circuit parameters to those in the model factors $f \in F$. This limit is lost if the arithmetic circuit is decomposable and smooth, but not deterministic. The authors [21] demonstrate this by considering an arithmetic circuit computing a Boolean factor $f(x)$, where the circuit's parameters are restricted to the values $\{0,1\}$. If the circuit does not have a sum node, then it is vacuously deterministic. Otherwise, it has at least one sum node. Consider the case where the circuit is only decomposable and smooth but not deterministic. This implies there is a sum node n that violates determinism. There must be a complete subcircuit with a non-zero coefficient that contains the node n (otherwise, n is a dead node). Because n violates determinism, there will be two distinct x-subcircuits with non-zero coefficients that differ by the branch selected at n. Since the arithmetic circuit must calculate the factor $f(x)$, the coefficients of all x-subcircuits must add up to $f(x) = 1$. This implies there exists an x-subcircuit whose coefficient is in the exclusive interval $(0,1)$. In other words, the arithmetic circuit has a parameter not present in $\{0,1\}$. Therefore, the restricted circuit cannot compute $f(x)$ if it is smooth, decomposable, and free of dead nodes, but not deterministic. This is referred to as parametric incompleteness.

The conclusion from this is that in order to calculate the marginals of a Boolean factor, the circuit must be decomposable, smooth and non-trivially deterministic. This means that circuit parameters cannot be limited to those already present in the model factors, otherwise a valid circuit may not exist. Therefore, although relaxing determinism can lead to much smaller circuits, finding these circuits becomes more complex because it may require searching for additional parameters [21].

2.1.5.2 Learning Arithmetic Circuits

In order to compile a factor graph into an arithmetic circuit, the compiler uses the theoretical framework of a network polynomial [39]. A network polynomial is a multilinear function over evidence indicators and network parameters.

The following is the representation of a network with four random variables and three factors:

$$\omega(\lambda_{A,0}, \lambda_{A,1}, \lambda_{B,0}, \lambda_{B,1}, \lambda_{C,0}, \lambda_{C,1}, \lambda_{D,0}, \lambda_{D,1}, \lambda_{D,2}) =$$

$$\lambda_{A,0}\lambda_{B,0}\lambda_{C,0}\lambda_{D,0}f_{AB}(0,0)f_B(0)f_{BCD}(0,0,0)+$$
$$\lambda_{A,1}\lambda_{B,0}\lambda_{C,0}\lambda_{D,0}f_{AB}(1,0)f_B(0)f_{BCD}(0,0,0)+$$
$$\lambda_{A,0}\lambda_{B,1}\lambda_{C,0}\lambda_{D,0}f_{AB}(0,1)f_B(1)f_{BCD}(1,0,0)+$$
$$\lambda_{A,1}\lambda_{B,1}\lambda_{C,0}\lambda_{D,0}f_{AB}(1,1)f_B(1)f_{BCD}(1,0,0)+$$
$$\vdots$$
$$\lambda_{A,1}\lambda_{B,1}\lambda_{C,1}\lambda_{D,1}f_{AB}(1,1)f_B(1)f_{BCD}(1,1,1)$$

The λ variables are the indicators which define a set of possible worlds. Here 'world' just means a unique row in a conditional probability table where all the random variables have set values. The network parameters here are the factors.

An indicator variable may be set to 1 if that indicator is true in the world of interest. It is set to zero otherwise. If the entire distribution is written as a table, indicator variables are associated with the rows of this weight table where the states are true.

Considering some world w:

$$[\![\lambda_{v,i} \mid w]\!] = \begin{cases} 1 & \text{if } v = i \text{ in world } w \\ 0 & \text{otherwise} \end{cases}$$

Any factor graph can be thought of as a network polynomial. Since the network polynomial only has sum and product nodes it can be represented as an arithmetic circuit. The compiler constructs an arithmetic circuit that is guaranteed to be functionally equivalent to the network polynomial of the input factor graph.

It should be noted that the network polynomial is never computed completely. When the number of factors in a factor graph increases, the number of terms in the polynomial goes up exponentially. Computing it completely is often intractable. Instead certain patterns in the terms of the polynomial and the factors of the factor graph are utilised to discard terms which are unnecessary (eg: for instance, if a factor is zero, all terms that have that factor will be zero). This idea of a network polynomial is to provide the mathematical concepts to describe the compilation process and to prove correctness. The network polynomial also helps us understand how queries work on the circuit.

As mentioned before, the structure of an arithmetic circuit is a directed acyclic graph (DAG) where directed arcs point from nodes to parent nodes. A node may be either a constant value node (leaf node), an input variable node (leaf node), or an operation node (interior node). An operations node represents an arithmetic operation of its children. Each node of an arithmetic circuit represents a function of the input variables nodes. Normally,

the focus is solely on the function represented by a selected 'top' node. Once an input value is supplied for each input variable node, then the value of any node is defined. The values of the leaf node are trivially defined. The value of any operation node is its operation applied to the value of its child nodes.

2.1.5.3 Probabilistic Circuits

Probabilistic Circuits (PC) are computational graphs that represent a joint probability distribution as product units and sum units of simpler distributions [24]. They can encode many layers of hidden variables into graphs with large numbers of parameters and connections. They encode functions that describe a distribution, for example a Probability Mass Function (PMF) or a Probability Density Function (PDF). PCs were introduced to reconcile and abstract from the various syntactic and graphical formalisms of some existing models including arithmetic circuits. PCs aim to represent and compute probability distributions efficiently; where as arithmetic circuits compute general polynomial functions without a probabilistic interpretation [24]. Arithmetic circuits can be used to represent probabilistic computations when compiled from probabilistic graphical models. But unlike PCs, they do not have nodes inherently capable of supporting various probabilistic queries without modification. Even though they have a probabilistic interpretation, PCs are not PGMs because they do not have representational semantics. The semantics of PCs are operational. The nodes of PGMs represent random variables. However, the units within computational graphs of probabilistic circuits explicitly encode instructions for evaluating their probability distributions and responding to probabilistic queries.

Probabilistic Circuits can encode the computation required to answer the following queries discussed in section 2.1.3: (1) probability of state or complete evidence queries (2) marginal queries (3) Most Probable Explanation (MPE) queries (4) Maximum a Posteriori hypothesis (MAP) queries. The following three computational units are needed to construct a probabilistic circuit.

1. **Distribution Unit**

   A distribution unit represents a whole probability distribution over a collection of random variables. It is also referred to as an input unit. The calculation performed by a distribution unit, i.e., the output it produces for a given input, depends on the type of query being addressed and the parametric form of the distribution it represents [24].

For a computational node acting as a distribution unit in a PC, the node type does not need to be switched to answer different queries from various classes. Here 'switching node type' refers to changing the computational behavior of a node, such as from performing addition to performing multiplication, or from one kind of probabilistic operation to another, to suit the needs of a specific query or computation. Computational models including factor graphs, BNs and arithmetic circuits require node switching because the nodes are not inherently versatile and are designed to perform specific, fixed operations. For example, an arithmetic circuit might be compiled to compute marginal probabilities efficiently but may not be optimized for computing MAP estimates. Since distribution unit in PCs encapsulate probability distribution functions and act as black boxes, this type of node switching is not required.

Probabilistic circuits also have sum units to represent mixtures and product units to represent factorization.

2. **Product Unit**

   Product units are used to decompose a joint probability distribution into smaller distributions via factorization. Consider the following definition for factorized models [24]. Assume that a probabilistic model m representing a joint distribution for a group of random variables $X = \bigcup_{i=1}^{k} X_i$ is separated into a collection of disjoint sets $X_i \cap X_j = \emptyset$ for any $i \neq j$ in $1, \ldots, k$ where $k > 1$. Then m can be considered factorized only when:

   $$P_m(X) = \prod_{i=1}^{k} P_{m_i}(X_i)$$

   Here every P_{m_i} is a probability distribution over the subset of random variables X_i.

   Classical PGMs like Bayesian Networks (BN) work on the principle of decomposing a joint distribution into smaller factors where the dependency structure of the random variables decides the way in which the distribution factorizes [72]. The base case of factorized models are the fully factorized distributions (happens when all the random variables in the graph are not connected). In this case, the joint probability distribution decomposes into individual univariate marginal distributions.

   A fully factorized model over any computational graph can be represented as a product unit connected to some input distribution units. Such product units representing factorizations are important for several tractable inference scenarios where complex inference problems can be broken down into simpler ones [24]. However, it is often

necessary to represent a model that is factorized but not fully factorized. Such models, which are more expressive, require product units to receive input from both from distribution units and from another type of unit called sum unit.

3. **Sum Unit**

   Sum units are essentially mixture models [109] that combine a number of simple distributions into a single model [24]. For the purposes of probabilistic circuits, the focus is on finite mixture models with positive weights. This type of mixture model can be defined as follows. Assume that $\{P_{m_i}\}_{i=1}^{k}$ is a finite collection of probabilistic models. All of those models are described over a group of random variables X. A mixture model is a probabilistic model that can be described as the convex combination:

   $$P_m(X) = \sum_{i=1}^{k} \theta_i P_{m_i}(X)$$

   for a set of positive weights (mixture parameters) $\theta_i > 0, i = 1, \ldots, k$ and $\sum_{i=1}^{k} \theta_i = 1$.

   Mixture models composed of fully-factorized components can encode distributions that are not fully factorized because every mixture implicitly encodes a categorical latent variable. This latent variable functions as a selector over the different components in the mixture and therefore they introduce correlations into the various component distributions. This means that the weights in a mixture density represent the initial probability of assigning such a latent variable to a particular value denoting a component. The component distributions can be interpreted as conditional distributions if conditioning occurs on a certain value.

   A mixture model's distribution may be naturally depicted as a computational graph by incorporating a sum unit which calculates a weighted average of its inputs [24]. Because these weights represent mixture components, they are shown on the edges linking the sum unit to its own input nodes.

Now that all the computational units in a PC have been introduced, a proper definition can be formulated. A PC C over random variables X is defined as a pair (G, θ), where G represents the computational graph (or circuit structure) and θ denotes the parameters of the circuit (or circuit parameters). The computational graph G is in the form of a rooted DAG consisting of the aforementioned computational units. The probabilistic circuit C calculates a function which describes a distribution $P(X)$.

2.1.5.4 Advances in Tractable Inference

Tractable probabilistic inference has seen a lot of advancements in recent years. Arithmetic circuits (AC) and Probabilistic Circuits (PC) are defined by a graph that allows for the exact and efficient evaluation of probabilistic queries, specifically marginals and conditionals. Recent advancements in PCs include hardware-centric scaling by moving away from sparse graph representations to dense, tensorized operations compatible with modern GPUs. This shift started with the introduction of Einsum networks by Peharz et al [110]. The authors proposed Einsum Networks (EiNets) - a method to combine large numbers of arithmetic operations into a single monolithic einsum operation. An einsum (Einstein summation) is a mathematical convention used to express tensor operations concisely. It eliminates the need for explicit summation symbols ($\sum$) by determining operations based on index placement. It can be found in numerical libraries like NumPy [144], PyTorch [146], and TensorFlow [147] that evaluates tensor operations using index notation.

Memory bandwidth bottlenecks are a problem in very large, deep circuits because the sheer volume of intermediate activations in a massive tensorized circuit can overwhelm GPU memory. Liu et al [82] developed PyJuice, a general GPU implementation design for probabilistic circuits (PCs) to address this problem. The authors identify that standard deep learning frameworks are not optimized for the specific sparsity patterns of PCs. PyJuice introduces a Just-In-Time (JIT) compilation phase that analyzes the logical structure of a PC and compiles it into a compact, block-sparse memory layout. This layout is specifically designed to minimize Input/Output (IO) operations between the GPU memory and the compute cores. This makes it possible to efficiently leverage Tensor Cores available in modern GPUs [82]. Similarly Dang et al [34] created an open-source Julia package providing tools for logic and probabilistic reasoning and learning based on PCs.

Zhang et al [170] introduced a scaling method that leverages structured sparse parameterizations for the sum blocks in PCs. The authors note that advancements have so far improved the scalability of PCs by either leveraging their sparse properties or through the use of tensorized operations for better hardware utilization. However, no existing method fully exploits both aspects simultaneously. The contribution here is the parameterization of the weight matrices of sum nodes as products of sparse Monarch factors. A Monarch matrix is a product of block-diagonal permutation matrices. This structure allows the layer to represent a dense linear transform while storing only a fraction of the parameters. Theoretically, the authors demonstrate that this parameterization arises naturally when multiplying multiple smaller circuits together. In an empirical evaluation, they demonstrate state-of-the-art generative modelling performance and superior scaling behaviour.

Sensitive data often reside in decentralized silos (eg. hospital records) due to privacy concerns. So, there is a need for some type of federated learning for PCs. Seng et al [135] recently proposed federated circuits (FCs) to provide a unified framework for this. It allows us to scale PCs on distributed learning environments, train PCs faster and unifies different types of federated learning for PCs in one framework. In another recent paper Liu et al [84] introduces latent variable distillation - a framework to induce semantics-aware assignments for latent variables in PCs. They leverage less tractable but more expressive deep generative models (eg. GANs, VAEs) to provide extra supervision to PC optimizers over the latent variables of PCs. This mitigates the failure of existing optimizers to utilize the expressive power provided by large PCs.

Initially, PCs were primarily viewed as estimators of Probability Mass Functions (PMFs) or Density Functions (PDFs). However, a recent work by Broadrick et al [11] demonstrates that PCs are efficient polynomial evaluators. Various polynomial semantics have been considered in the literature for probabilistic circuits (PCs) and arithmetic circuits (ACs). The standard network polynomial discussed in subsection 2.1.5.2 is one example. Others include likelihood polynomials, generating functions and Fourier transforms. Broadrick et al [11] show that for binary variables they are all equivalent in the sense that a circuit in one semantics can be transformed into a circuit in any of the others with at most a polynomial change in size. This is an important contribution because it demonstrates that they are all tractable for marginal inference on the same class of distributions. They also explore an extension of one such polynomial semantics called probabilistic generating circuits (PGCs). Standard PCs (like SPNs) are restricted to non-negative weights to ensure they define valid probability distributions. PGCs can allow negative parameters within the circuit, provided the final expanded coefficients (the probabilities) remain non-negative [171]. The utility of negative parameters is further explored by Wang and Van den Broeck [158].

In another paper, Broadrick et al [10] examines what happens when PCs compute Cumulative Distribution Functions (CDF). Typically they are used to compute Probability Mass Functions (PMFs) or Density Functions (PDFs). For binary random variables, the authors identify an exact equivalence between the multilinear polynomial representing the CDF and the Probability Generating Function (PGF). This structural identity implies that circuits computing the PMF and CDF are equally succinct and enables polynomial-time transformations between the two representations. For continuous random variables, the authors demonstrate that PCs satisfying the smoothness and decomposability properties allow for efficient conversion between PDFs and CDFs. To extend this tractability to finite discrete variables, the authors introduce the Less-Than Encoding, a method that maps dis-

crete values to binary indicators while preserving the elementwise order relations required for CDF semantics.

In general, various applications of PCs depend on the ability to multiply two circuits. Zhang et al [172] proposes polynomial-time algorithms to restructure a PC to conform to a target v-tree (variable orderings). Another recent work by Leland and Choi [76] investigate whether the strict structural constraints of PCs like decomposability and determinism can be relaxed if one accepts an approximate answer rather than an exact one. Characteristic Circuits (CCs) - introduced by Yu et al [169] - define the circuit over the characteristic function of the respective probability measures instead of the density function. The characteristic function is the Fourier transform of the probability measure. CCs provide a unified formalization of distributions over heterogeneous data in the spectral domain.

The use of machine learning algorithms for decision making is becoming more common. So, researchers are taking a closer look at the limitations of standard algorithmic fairness notions. For instance, most existing work on algorithmic fairness assumes complete feature observation at prediction time. However, this assumption is insufficient for models capable of inference with partial observations. Relying on complete data masks bias patterns and leads to false fairness certifications [134]. Previous work on Naive Bayes Classifiers [23] has introduced a new notion of fairness that asks whether the model exhibits any discrimination pattern when one or more sensitive attributes of a given individual is disclosed. Here the individual or group of individuals is characterized by some partial assignments to the features. The individual(s) may see a major discrepancy in the prediction by the model after revealing one or more sensitive attributes like race or gender. This method explicitly accounts for partial observations, thereby providing a more fine-grained notion of fairness. Selvam et al [134] extends this notion of fairness from Naive Bayes Classifiers to Probabilistic Circuits (PCs). The authors introduce specific sub-classes of discrimination patterns, specifically minimal, maximal, and Pareto optimal patterns, to efficiently summarize the potentially exponential number of biases into interpretable categories. The authors also develop two distinct algorithmic approaches exploiting the structural properties of smooth and decomposable circuits.

Ventola et al [156] examined how well PCs work with out-of-distribution (OOD) data. They show that OOD data is a failure mode for standard PCs. In neural networks, dropout is used to estimate uncertainty by running the model multiple times with different neurons dropped out. This is computationally expensive. The authors show that for PCs, the moments of the dropout distribution can be computed analytically in a single pass. This provides a fast, tractable measure of model uncertainty. A paper by Sidheekh et al [137]

examines the problem of late multimodal fusion for discriminative learning. In robust perception, systems must handle sensor failures. Sidheekh et al [137] uses PCs to fuse data from multiple modalities (e.g., audio and video). The PC models the joint distribution of the expert predictions.

There has been research done to address the integration of PCs with deep learning [3] [4] [90] [70] [91]. The goal here is using PCs to inject reasoning capabilities into neural networks. Neural networks are excellent function approximators but struggle with hard logical constraints. PCs can help with this [4]. There has also been work done to combine the structural advantages of PCs with the generative power of deep learning [137] [168]. Standard probabilistic models capture correlation, not causation. Some recent literature explores PCs as a tractable substrate for causal modeling [12] [114]. Liu et al [83] applies PCs to data compression. Since PCs provide exact likelihoods, they can be directly used with arithmetic coding for lossless compression. The authors show that PCs achieve state-of-the-art compression rates on tabular and image data [83]. Large PCs are susceptible to overfitting. A paper by Shih et al [136] addresses this problem by introducing HyperSPNs. This is a method for generating the mixture weights of large PCs using a small-scale neural networks.

The above sub-sections described the structure and properties of different graphical models. Inferencing and learning of PGMs were also briefly explained. Overall, this section covered all the necessary 'Background' for my thesis. This 'Background' section included all the fundamental concepts necessary for my research in synthetic data generation. Synthetic data generation from PGMs require a mechanism for sampling and a mechanism for privacy preservation. Section 2.2 is a literature review on the existing sampling algorithms for sampling from PGMs. Section 2.3 contains a literature review on the differential privacy mechanism for privacy preservation. Finally, Section 2.4 provides a literature review on the existing state-of-the-art synthetic data generators.

2.2 Sampling of Probabilistic Graphical Models

Sampling a probabilistic model involves drawing samples from the model, possibly conditioned on assumptions about states of some random variables. Importantly, as the number of samples approaches infinity, the empirical distribution of the samples approaches the distribution defined by the model. This property is guaranteed by design of the sampling method even when the model's joint probability distribution is not directly computable. There are two main groups of sampling methods for probabilistic models, independent

and dependent [53]. Independent sampling refers to the process of drawing individual samples from a model in such a way that each sample is generated independently of the others. Methods include forward sampling, rejection sampling and inverse transform sampling. Dependent sampling methods involve drawing a sequence of samples, where adjacent pairs of samples are related. Methods include Metropolis-Hastings, Gibbs and other Markov Chain Monte Carlo (MCMC) samplers.

2.2.1 Forward Sampling

The simplest approach to generate samples from the joint distribution of a Bayesian Network is forward sampling. Typically, in forward sampling the topological ordering of all the nodes is identified. A topological ordering of a directed acyclic graph (DAG) is a linear ordering of all its nodes such that, for every directed edge from node A to node B, A appears before B in the ordering. This ensures that each node is only sampled after all its parents [72]. A simple algorithm like depth-first search can be used to find the topological ordering. After the topological order is obtained, each node X_i is sampled conditioned on its parents.

$$X_i \sim P(X_i | \text{Parents}(X_i))$$

The samples for all the nodes obtained by passing once through the network collectively constitute a sample from the joint distribution defined by the network. Because each variable is sampled in sequence according to its parents' (already-sampled) values, the joint distribution is respected automatically. The key advantages of forward sampling are its simplicity and its direct use of the structure of the Bayesian network. The disadvantage here is that conditioning requires it to be combined with rejection sampling (explained below), which can be computationally inefficient for low probability conditions.

2.2.2 Rejection Sampling

Rejection sampling is used to generate independent samples from a complex target distribution by sampling from a simpler proposal distribution, and accepting each sampled point with probability proportional to the ratio between the target and proposal densities. Rejection sampling can be used to implement sampling from a conditioned distribution, when the proposal distribution is a Forward sampler. Forward sampling, as described above, only samples from an unconditional (joint) distribution. However, it is often necessary to estimate conditional distributions given some evidence E. Here E represents some observed variables in the Bayesian Network. Rejection sampling extends forward sampling

by retaining only those samples in which the evidence E takes the observed values e [72]. In other words, reject any sample that conflicts with the observed evidence.

The problem with rejection sampling is that for a sufficiently large state space the number of unrejected samples can be quite small. The number of candidate samples that are not rejected from a total sample set of size M is $MP(e)$. For instance, consider $P(e) = 0.001$, then even for an M of 10000, the expected number of useful samples is 10. This high attrition rate limits the use of rejection sampling in many real-world applications.

2.2.3 Markov Chain Monte Carlo (MCMC) Sampling

Markov Chain Monte Carlo (MCMC) represent a class of dependent sampling algorithms that are used to draw samples from a probability distribution. Typically, they are faster than independent samplers like rejection or inverse-transform. MCMC samplers construct a Markov Chain whose elements' distribution approximate the target distribution.

The Metropolis-Hastings algorithm is an MCMC sampling method that generates the Markov chain based on a proposal distribution [61]. The sampler starts by picking an initial state for the network (e.g. by randomly assigning values to each random variable). A candidate state is then proposed based on the proposal distribution. The state is either accepted or rejected based on a pre-defined acceptance ratio. If the state is accepted, then the chain transitions to proposed state. As the chain evolves, the fraction of time it spends in each region of the state space converges to the target distribution, thereby enabling efficient sampling from otherwise intractable or high-dimensional distributions [61].

Gibbs sampling is a special case of Metropolis-Hastings. The Gibbs sampler updates each random variable one at a time, sampling from its conditional distribution given all other random variables (its Markov blanket in a Bayesian network) [18]. This is typically simple in a Bayesian Network because the conditional distribution for each random variable is known. In a standard Gibbs sampler, the acceptance ratio is 1. This is because of the fact that the Gibbs sampler draws directly from the true conditional i.e. there is no proposal mechanism that might need rejecting or accepting with some probability.

In dependent samplers, since nearby samples are correlated and are not independent, the samples from the beginning of the chain may not accurately represent the model distribution and are discarded, forming the so-called burn-in period. Determining appropriate burn-in periods can be challenging, and large supports generally necessitate lengthy burn-in periods leading to inefficient sampling [54] [66]. An additional problem is that multimodal distributions may cause dependent sampling methods to become stuck in modes

with high probability. Random restarts can alleviate this issue, but each restart requires an additional burn-in period, further diminishing efficiency.

2.2.4 Inverse Transform Sampling

Inverse transform sampling is an independent sampling technique used to generate random samples from a chosen probability distribution by leveraging uniform random numbers [72]. At its core, the method hinges on the cumulative distribution function (CDF) of the target distribution. This method requires an ordering over possible states of the model; a point on a cumulative probability distribution $c(x)$ is defined by the sum $\Sigma_{y \leq x} P(X = y)$. Inverse transform sampling draws a random variate u uniformly from $[0,1]$, then determines the corresponding joint states x using the inverse cumulative distribution, i.e., x that minimises $c(x)$ subject to $c(x) \geq u$. When the support is large, this approach is not practical unless the sum can be efficiently determined. Because programming libraries readily provide uniform random numbers, this technique simplifies the sampling process for a broad array of probability distributions.

In practice, inverse transform sampling is commonly used in scenarios ranging from basic Monte Carlo simulations to more specialized applications like generating synthetic data or running tests that require a certain probability distribution. In continuous distributions, it is especially valuable when a closed-form expression for the inverse CDF is available, as the method then becomes both straightforward and computationally feasible. In discrete distributions, the inverse CDF can easily be calculated. When the inverse CDF cannot be expressed in a simple form, numerical methods or approximations may be employed, maintaining the broad utility of this approach.

2.2.5 Sampling for synthetic data generation

Sampling from a PGM has many applications. A typical use-case is executing the approximate inference procedure described in section 2.1.2. When exact computation of marginals or posteriors is intractable, methods like MCMC sampling can be used for approximating probabilities by averaging over many samples. Sampling from a PGM also has applications in predictive modeling and forecasting. By drawing samples conditioned on observed data, it is possible to generate future or missing data points, quantify prediction intervals, and stress-test scenarios. Similarly, sampling has applications in anomaly detection and generative modeling as well.

However, for this thesis, the application of interest is synthetic data generation. A synthetic data generation process involves designing a PGM that produces data in a way that captures the important statistics of the reference database. As discussed, independent sampling methods generate each sample from scratch without using information from previous samples. This can make them computationally costly for synthetic data generation. However, faster dependent sampling methods are not suitable for synthetic data generation because the sampler can introduce additional statistical relationships, beyond that which is in the model. This would make it difficult to determine whether a relationship apparent from the data is the effect of the model or the sampling algorithm. Here the need for an efficient independent sampling method is apparent.

The synthetic data is created by sampling the generative model (i.e. PGM). However, before creating the model itself, privacy preservation of sensitive data may be an important consideration. This may even be a requirement in certain critical government or corporate applications. The next section of this thesis explains privacy preservation using differential privacy. I will explain different types of differential privacy, different algorithms for differential privacy and different properties of differential privacy.

2.3 Privacy Preservation of Sensitive Data

Synthetic datasets are a valuable tool in protecting privacy in critical domains like healthcare and law enforcement. There have been multiple techniques developed for synthetic data generation over the past few years. The idea of creating and releasing fully synthetic data was first proposed by Rubin [128]. In the previous section, the discussion focused on different methods for sampling from PGM models to generate synthetic data. It is necessary to ensure that the models being sampled do not disclose any private information. This section expands on the idea of privacy preservation in synthetic datasets.

2.3.1 Differential Privacy

Differential Privacy provides a mathematical framework for preserving the privacy of individuals in sensitive datasets. Consider a scenario with n individuals denoted $X_1, X_2, X_3, \ldots, X_n$. Every individual has their own datapoint. The raw data of the individuals is sent to an algorithm M, True differential privacy means that the presence or absence of any individual's data does not influence the algorithm's output.

Formally the algorithm $M : X^n \rightarrow Y$. Assume that the two datasets $X, X' \in X^n$ vary in just one entry. These types of datasets will be referred to as neighbouring datasets and will be denoted by $X \sim X'$. M is $\varepsilon-$(*pure*) *differentially private* ($\varepsilon-$(pure) DP) if for all neighbouring X, X' and all $T \subseteq Y$, there is

$$Pr[M(X) \in T] \leq e^{\varepsilon} Pr[M(X') \in T] \tag{2.7}$$

Here the randomness is defined by the decisions in the algorithm M. Dwork et al [48] originally introduced this definition of differential privacy.

Wasserman and Zhou [160] provide a slightly different definition for differential privacy. Here, the authors [160] define it with regards to an adversary trying to guess between X and X'. One of the given two hypothesis is guaranteed to hold.

H_0: the underlying dataset is X.

H_1: the underlying dataset is X'

Here H_0 is the null hypothesis and H_1 is the alternate hypothesis of the adversary. Let p be the probability that the adversary predicts H_1 when H_0 is true (a "false positive") and q be the probability that the adversary predicts H_0 when H_1 is true (a "false negative"). ε-differential privacy has two implications here.

$$p + e^{\varepsilon} q \geq 1 \tag{2.8}$$

$$e^{\varepsilon} p + q \geq 1 \tag{2.9}$$

It is clear that when $\varepsilon = 0$ the adversary can only ignore the data and make a random guess. A value greater than zero allows some statistical advantage over random chance [160]. This idea was also explored in [69] and [7].

Both of these definitions provide the basic framework for determining $\varepsilon-$(*pure*) *differential privacy* of a dataset. Approximate differential privacy will also be defined in a following subsection. First, however, an algorithm for calculating differential privacy, called the Laplace algorithm, is explored.

2.3.2 Laplace algorithm

The following definition of the Laplace algorithm (or alternatively the Laplace mechanism) is from the book "The algorithmic foundations of differential privacy" by Dwork [49]. An important and often used type of database query is the numeric query $f : \mathbb{N}^{|X|} \rightarrow \mathbb{R}^k$. They map databases to k real numbers. The Laplace algorithm can solve all types of numeric

queries. To understand the Laplace algorithm, an important parameter called l_1 sensitivity must be examined first.

The l_1-sensitivity of a function $f : \mathbb{N}^{|X|} \rightarrow \mathbb{R}^k$ can be defined as:

$$\Delta^{(f)} = \max_{X,X'} ||f(X) - f(X')||_1 \tag{2.10}$$

Here X and X' are *neighbouring datasets*.

The Laplace algorithm for differential privacy works by calculating f and adding noise from the Laplace distribution to each coordinate individually. The noise should be proportional to the l_1-sensitivity.

Formally for a function f, the Laplace algorithm is defined as:

$$M(X) = f(X) + (Y_1, \ldots, Y_k) \tag{2.11}$$

Here Y_i are independent $\Delta^{(f)}/\varepsilon$ random variables.

As noted by Dwork and Roth [49], the Laplace algorithm is ε - differentially private.

2.3.3 Approximate Differential Privacy

The concept of $\varepsilon-$(*pure*) *differential privacy* is useful in understanding the basic idea behind differential privacy. However, when ε is small it provides meaningless accuracy and when ε is large it delivers weak privacy [81]. Hence the notion of *approximate* (ε, δ) - *differential privacy* is introduced. This notion was first introduced by Dwork et al [47]. Formally, approximate differential privacy is defined as follows.

An algorithm $M : X^n \rightarrow Y$ is (ε, δ) - differentially private if, for all neighbouring datasets $X, X' \in X^n$ and all $T \subseteq Y$,

$$Pr[M(X) \in T] \leq e^{\varepsilon} Pr[M(X') \in T] + \delta \tag{2.12}$$

Equation (2.12) can be interpreted using the privacy loss random variable.

If Y and Z are two random variables, the privacy loss random variable $L_{Y||Z}$ is distributed by drawing $t \sim Y$, and outputting $\ln \dfrac{Pr[Y = t]}{Pr[Z = t]}$. Privacy loss is undefined if supports of Y and Z are not the same.

Consider the typical scenario described earlier, where M is the algorithm and X, X' are the neighbouring datasets. When Y and Z are equal to $M(X)$ and $M(X')$, it can be surmised that the privacy loss random variable specifies the likelihood of X being the input dataset instead of X', given the observation of $M(X)$.

Note that pure differential privacy (ε-DP) corresponds to $|L_{M(X)||M(X')}|$ being bounded by ε for all values of X, X'. This means that in the case of pure differential privacy, the absolute value for the privacy loss random variable will be bounded by ε with probability 1. In Chapter 3 Lemma 3.17 of their book, Dwork and Roth [49] show that in the case of approximate differential privacy the absolute value for the privacy loss random variable will be bounded by ε with probability 1 - δ. It should be noted that smaller values of δ correspond to stronger privacy.

2.3.4 Gaussian algorithm

The Gaussian algorithm (or alternatively the Gaussian mechanism) privatizes the statistic by adding Gaussian noise to every coordinate. In order to define the Gaussian algorithm, the l_2-sensitivity parameter must be defined.

Let $f : \mathbb{N}^{|X|} \to \mathbb{R}^k$. The l_2-sensitivity of f is:

$$\Delta_2^{(f)} = \max_{X,X'} ||f(X) - f(X')||_2, \tag{2.13}$$

Here X and X' are *neighbouring datasets*.

For the Laplace algorithm, the noise was proportional to the l_1-sensitivity. Here the l_2-sensitivity is used. The l_1 and l_2 norms have this relationship: for a given vector $x \in \mathbb{R}^d, ||x||_2 \leq ||x||_1 \leq \sqrt{d}||x||_2$. Hence l_2-sensitivity may be up to a factor $\sqrt{d}$ less than l_1-sensitivity. Dwork and Roth [49] show that this relationship means that the Gaussian algorithm adds $O(\sqrt{d})$ less noise. This may result in a marginally weaker privacy guarantee. So Gaussian algorithm is more useful for multivariate problems. The Gaussian algorithm is defined as follows.

Let $f : X^n \to \mathbb{R}^k$. Then

$$M(X) = f(X) + (Y_1, \ldots, Y_k) \tag{2.14}$$

Here the Y_i are independent $N(0, 2\ln(1.25/\delta)\Delta_2^2/\epsilon^2)$ random variables.

2.3.5 Properties of Differential Privacy

Post-processing

An interesting property of the differential privacy algorithms discussed above is that once a quantity is privatized, the privatization cannot be reversed without using the original data again [49] [81]. A given algorithm that is (ε, δ) - differentially private will remain so irrespective of any post-processing done on it. Formally this is stated as:

Let $M : X^n \rightarrow Y$ be (ε, δ) - differentially private, and let $F : Y \rightarrow Z$ be an arbitrary randomized mapping. Then $F \circ M$ is (ε, δ) - differentially private.

This property is important for the contributions of this thesis.

Group Privacy

Group Privacy discusses what happens when datasets differ in k entries instead of 1. Even though ε linearly scales with k, δ has an additional factor of $e^{(k-1)\delta}$. Let's formally define this.

Let $M : X^n \rightarrow Y$ be an (ε, δ) - differentially private algorithm. Consider two datasets X and X' that differ in exactly k positions. Then for all $T \subseteq Y$, there is

$$Pr[M(X) \in T] \leq exp(k\varepsilon) Pr[M(X') \in T] + ke^{(k-1)\varepsilon}\delta$$

Basic Composition

Next, the effects of running k differentially private algorithms on the same dataset are examined. Obviously the overall privacy guarantee will decay. Summing all the ε and δ values provides the overall privacy guarantee [49] [125].

Consider a sequence of algorithms $M = (M_1, \ldots, M_k)$. Here any given M_i is $(\varepsilon_i, \delta_i)$ - differentially private. The M^i values can be chosen sequentially and adaptively. Then M is $(\sum_{i=1}^{k} \varepsilon_i, \sum_{i=1}^{k} \delta_i)$ - differentially private.

Advanced Composition

Compared to basic composition, which simply adds the privacy loss of each algorithm, advanced composition offers tighter bounds on how much privacy is leaked with each additional analysis. Basic Composition can be overly conservative in its approach to calculating privacy loss. While advanced composition offers tighter bounds, it can be more complex to implement and analyze compared to basic composition. The bounds for δ can be tightened by including some slack with the term δ'. Kairouz et al [69] and Gopi et al [57] formally define advanced composition.

For all ε, δ, $\delta' > 0$, let $M = (M_1, \ldots, M_k)$ be a sequence of (ε, δ) - differentially private algorithms; where M_i's can be chosen sequentially and adaptively. Then M is $(\bar{\varepsilon}, \bar{\delta})$ - differentially private, where $\bar{\varepsilon} = \varepsilon\sqrt{2k\ln(1/\bar{\delta})} + k\varepsilon\dfrac{e^{\varepsilon}-1}{e^{\varepsilon}+1}$ and $\bar{\delta} = k\delta + \delta'$.

Kairouz et al [69] provide an optimal composition theorem for differential privacy. They generalise this result so that it can applied when the k different algorithms have different values for ε and δ.

2.3.6 Exponential algorithm

The Laplace and Gaussian algorithms were both designed for the privatization of numeric queries. This means that the algorithm outputs a value. However in many natural situations this approach may not be suitable. Consider the following example based on the paper by McSherry and Talwar [94].

A digital goods retailer can have unlimited copies of the items that they are selling. This could be digital copies of movies, books, software etc. Imagine n individuals, each wanting exactly one copy of an item. Any given individual i can pay up to a maximum v_i valuation. In order to get maximum revenue, the retailer's best option for pricing would be to select p which maximizes revenue $\sum_{i:p\leq v_i} p$. This can lead to scenario where a small sub-group of buyers have a high valuation for the item(s).

Here applying the differentially privacy algorithms, meant for just privatizing numeric queries, and adding noise to the data, can be an ineffective approach. To see why, consider a scenario where number of individuals n=3 with valuations v_1=100, v_2=100 and v_3=301. If the price is 100, then the revenue will be 300. But if the price is increased marginally to to 101, then the revenue drops to 101. Only if the price is increased all the way to 301 will the revenue go up. The revenue would drop to zero if the price goes to 302. This shows that the data is extremely sensitive to noise and this sensitivity makes the aforementioned differential privacy techniques ineffective. This can be resolved by thinking of prices as objects instead of values.

The exponential algorithm considers three items: (1) A dataset $X \in X^n$, (2) A set of objects H, and (3) A score function $s : X^n \times H \to \mathbb{R}$ [49].

The score function s is designed to accept a dataset, denoted by X, and an object, denoted by $h \in H$. The function then outputs a quantitative assessment of the 'goodness' of h with respect to X [49]. For the example provided above, the individuals' valuations would form the dataset, the set of all probable price values would be the objects and thus the score function would determine the revenue received at a given price from the set of all individuals.

In this setting, the dataset X is the information for which privacy needs to be preserved. Both the score function and the set of objects are publicly known. From this, it is possible to determine the sensitivity of the score function w.r.t the dataset only [49].

$$\Delta s = \max_{h\in H} \max_{X,X'} |s(X,h) - s(X',h)| \tag{2.15}$$

Here X and X' are neighbouring datasets.

With this, the exponential algorithm can be defined as follows.

The exponential mechanism (or algorithm) M_E with input values X, H, s selects and then outputs an object $h \in H$, where the probability of a specific h is selected as being proportional to $\exp(\frac{\varepsilon s(X,h)}{2\Delta s})$.

Here it is necessary to compute this quantity for every $h \in H$ to define the distribution. This may be computationally costly when X is large or even impossible when it is infinite. Nevertheless it theoretically proves that a certain error rate or sample complexity is possible. In their book, Dwork and Roth [49] prove that this algorithm M_E is *ε-differentially private*.

M_E can be used to select an object with a score close to the best possible score. The small additive amount lost will depend on the total number of candidate objects and the sensitivity ε. The utility of M_E can be described as follows.

Consider a dataset X with an optimal score $OPT(X) = \max_{h \in H} s(X,h)$ attained by the best object h w.r.t X. Let $H^* = \{h \in H : s(X,h) = OPT(X)\}$ be the set of objects that obtain this score. Then it follows:

$$Pr\left[s(M_E(X)) \leq OPT(X) - \frac{2\Delta s}{\varepsilon}(\ln(\frac{|H|}{|H^*|}) + t)\right] \leq exp(-t) \tag{2.16}$$

The following corollary can also be derived in the same setting since $|H^*| \geq 1$.

$$Pr\left[s(M_E(X)) \leq OPT(X) - \frac{2\Delta s}{\varepsilon}(\ln(|H|) + t)\right] \leq exp(-t) \tag{2.17}$$

The proof for this can be found in the book by Dwork and Roth [49]. The Laplace algorithm described earlier is an instantiation of the exponential algorithm.

2.4 Synthetic Data Generation Techniques

An important technique for maintaining privacy in synthetic datasets was discussed in the previous section. This section explores some recent advances in the field of synthetic data generation. Broadly, synthetic data research goes in two possible directions: (1) Exploration of new synthetic data generation mechanisms [107] [112] [117] [166] [173] [93] [175] [67] (2) Real-life efficiency of such generators [120] [63] [101] [62] [33] [60] [8] [35]. The differential privacy algorithms explained earlier are used in many of these synthetic data generation techniques. This section begins by exploring various new synthetic data generation mechanisms. Specifically, mechanisms relying on graphical models, copulas, marginals, and Generative Adversarial Networks (GAN) are examined.

2.4.1 DataSynthesizer (DS)

The first synthetic data generator to consider is called DataSynthesizer and it was developed in 2017 by Ping et al [112]. It does not natively support multi-table relational datasets. However, it provides certain privacy guarantees via differential privacy. It consists of three main modules: (1) DataDescriber, (2) DataGenerator and (3) ModelInspector. Overall these three modules make up an end-to-end system that generates synthetic datasets by taking private datasets as inputs.

The DataDescriber module takes the private input dataset and it infers the domains or estimates of distributions of the various attributes [112]. The inferred attributes are then saved in a dataset description file. Here the domains are the set of permitted values. DataSynthesizer supports the 'integer', 'float', 'string' and 'datetime' datatypes. All attributes are either specified by the user or inferred by the DataSynthesizer. For all categorical attributes, DataDescriber computes the frequency distribution of values and represents it as a bar chart. Categorical attributes with large domains (like names) are treated as non-categorical strings. This allows DataSynthesizer to generate realistic-looking synthetic data during synthesis, including creating random strings that mimic the length and style of real names and IDs. For non-categorical datetime and numerical attributes, the DataDescriber represents the distribution by an equi-width histogram. For every non-categorical string attribute, their maximum and minimum lengths are stored.

The DataDescriber can be called in an independent attribute mode. In this mode, the module performs frequency-based estimations of all the unconditioned probability distributions of the various attributes [112]. Bar charts are used to capture the probability distributions for categorical attributes. On the other hand, the distributions of numerical attributes are stored as histograms. The Laplace algorithm for differential privacy is implemented by the DataDescriber for adding controlled noise to the calculated distributions. Since the values of the attributes are often correlated, the DataDescriber can also be called in correlated attribute mode [112]. In this mode, the DataDescriber uses a Greedy Bayes algorithm to create a Bayesian Network that models the correlated attributes. The Greedy Bayes algorithm is based on a paper by Zhang [173].

The dataset description file created by the DataDescriber is provided to the DataGenerator which samples synthetic data from the probability distributions specified in the file. The size of the output synthetic dataset needs to be specified by the user. DataGenerator can be called in random mode, independent attribute mode or correlated attribute mode. Random mode is used to generate type-consistent random values for all the attributes. In independent attribute mode, the probability distributions in the dataset description file

are uniformly sampled to generate samples. When called in correlated attribute mode, the attribute values are sampled in the correct order from the Bayesian network created by the DataDescriber. As discussed earlier, the performance of differential privacy degrades with repeated queries. In order to avoid data leakage, unique random seeds can be provided for all users that need to generate synthetic datasets [112].

In order to draw useful inferences from synthetic data, the synthetic dataset needs to have similar probability distributions to the original input dataset. The ModelInspector module is used to examine this similarity. Visual comparison of the attributes is permitted by the ModelInspector in independent attribute mode. In correlated attribute mode, it displays "before" and "after" pairwise mutual information matrices and depicts Bayesian networks, permitting rapid evaluation of statistical fidelity between the original and synthetic datasets.

2.4.2 Synthetic Data Vault (SDV)

The next data generator to consider is the Synthetic Data Vault (SDV) developed by Patki et al in 2016 [107]. It natively supports multi-table relational datasets. In order to understand how the SDV works, first the concept of a Gaussian copula must be described. A copula is a mathematical tool that separates the modeling of marginal distributions from the modeling of their dependence structure. In particular, the Gaussian copula uses the multivariate Gaussian normal distribution for describing the aforementioned dependence structure. This is done by creating the joint probability distribution of all the random variables. First the marginal distributions of all the random variables are converted into standard normal distributions with the help of their cumulative distribution functions (CDF). The dependence between these random variables are then modelled using a covariance matrix. The Gaussian copula combines these new distributions with the covariance matrix to construct the joint probability distributions of the original random variables. The formal mathematical definition for this can be found in the book by Nelson [98] and in this article by Trivedi et al [150].

The SDV estimates the joint distribution of the population using this Gaussian Copula [107]. Consider a population P and a private dataset D drawn from P. The population's CDF F is modelled by the Synthetic Data Vault (SDV) from the sample. This is done by specifying the marginal distributions of the different attributes $F_1, \ldots, F_d$ and the dependency structure among them using the Gaussian Copula. The sample D can be used to determine the marginal distributions and the Gaussian Copula is represented as a function of the covariance matrix. This works well when only a standalone table needs to be modelled.

If there are multiple related tables as in a relational database, the authors [107] propose a technique called Conditional Parameter Aggregation (CPA) that enumerates how a child table's information must be included into the main table. This is only needed if the table being considered is not a leaf table (i.e., when at least one other table contains a column that refers to rows in the current table). CPA can be calculated using the following iterative process.

1. Select the first row.

2. A conditional primary key lookup is performed through the whole database using the ID of the first row. Suppose there are m foreign key columns that refer back to the present table, the conditional primary key lookup will return m sets of rows. These are the sets of conditional data.

3. The Gaussian Copula process is performed on all sets of conditional data. It can be observed that this will return m distribution sets and m covariance matrix sets. When provided a parent ID, these *conditional parameters* will constitute parameters of a model for a subset of data from a child.

4. New columns are added to the original table to place these conditional parameters along the same row. The authors [107] refer to these new columns as derived columns.

5. The total number of children for every parent is added as a new derived column.

6. Iterate through to the second row and onwards.

The CPA and the Gaussian Copula require all attributes to be numerical. When they are not numerical a pre-processing step is required to convert all categorical attributes to numerical values in the range $[0, 1]$. In order to convert datetime values to numerical attributes, the number of seconds from the given timestamp is counted. All the missing values are taken to be null values [107].

The CPA approach works when there is a parent-child relationship. But the children can have more children as well. Hence, the authors [107] proposed a Recursive Conditional Parameter Aggregation (RCPA) approach where the CPA is done recursively through all the parent's descendants. The RCPA is applied by the Synthetic Data Vault (SDV) to create an overall database model.

The authors [107] provide two approaches to generating synthetic data based on the database model. They are the model based and knowledge based approaches. In the model

based approach, synthetic data can be generated based on the model that was just calculated. The data is generated by using a sampling algorithm that samples data from the aforementioned joint probability distribution and covariances.

The sampling algorithm takes two inputs: (1) F - the joint probability distribution that needs to be sampled (2) Σ - the covariance matrix. An n-dimensional Gaussian vector is randomly sampled. This results in a vector of random numbers that follow a normal distribution. Here n is the number of columns in the data. The next step is to find the Cholesky decomposition of the covariance matrix. Cholesky decomposition is a way to break down the covariance matrix into the product of a lower triangular matrix and its transpose (i.e. $LL^T = \Sigma$). Then the random Gaussian vector is multiplied by the lower triangular matrix L. This gets us a vector u in Copula space. The inverse cumulative distribution function (referred to as Φ) is then applied to the transformed vector u. Since this step involves applying the inverse of the Gaussian Copula, it converts u back into its original space. The resulting vector x represents a sample from the distribution. Of course this needs to be post-processed to look like the original data.

The authors [107] point out that this sampling algorithm works well for model-based synthesis of data. However, in the knowledge based approach, this sampling algorithm is not sufficient. Knowledge based approach incorporates prior knowledge the user may have to generate the synthetic data. It can involve generating additional rows for a group that is underrepresented in the data. This requires modifications to the sampling algorithm because some of the values in Σ and F may already have been observed. So an update needs to be performed to get new F' and Σ' for the unobserved values. The parent would also need to be inferred based on the value provided by the user.

The observed data is represented by updating the Σ matrix and the vector u. At first, $u = 0$ as specified by the Gaussian Copula method. The Σ matrix and the u vector can be rearranged to get the unobserved (unknown) variables to the top. Then Σ and u can be updated using the known observations to determine new values (Σ' and u') for the unknown variables. The parent row also needs to be inferred if the user has observed certain values for a row and the row has parents. The authors [107] point out that the Synthetic Data Vault selects the parent row of a variable from the weighted distribution of the Gaussian Copula.

Apart from Synthetic Data Vault (SDV) [107], there are multiple synthetic data generators that use copulas [77] [6] [52]. These approaches focus on generating differentially private synthetic datasets by modeling the joint dependence structure of the attributes separately from their marginal distributions. Specifically, frameworks such as DPCopula [77] and the method proposed by Asghar et al [6] utilize Gaussian copulas to capture pairwise

correlations. In contrast, the framework by Gambs et al [52] implements vine copulas to model more complex and nested conditional dependencies.

2.4.3 Synthetic Data Generation Using Graphical Models

There has been some research where the authors use different Graphical Models to generate synthetic data. The DataSynthesizer [112] discussed in subsection 2.4.1 uses a differentially private Bayesian Network to model the joint probability distribution. It was based on the PrivBayes paper by Zhang et al [173]. The differentially private Bayesian Network is a PGM that is represented by a directed acyclic graph. The random variables (or nodes) of the Bayesian Network are attributes of the dataset. The edge from one node u to another node v represents the probability of v being caused by u (i.e. $P(v|u)$). There can be multiple values for every attribute, represented by a Conditional Probability Table (CPT). All possible values in $P[v = y|u = x]$ are necessary. If the node v is dependent on more than one node (i.e. $P[v|U]$ where $U = u_1, \ldots, u_k$), then $P[v|U]$ is needed to sample v. The sampling process starts with the node that has in-degree 0. The graph is then traversed in the generation order specified by the Bayesian Network to calculate the remaining attributes.

In PrivBayes [173], the Bayesian Network is learned in two steps. First, all the joint probability distributions $P[v, u]$ must be determined. The conditional distributions $P[v|u]$ can be determined from the aforementioned joint distributions. Based on the composition property of differential privacy discussed in subsection 2.3.5, it can be observed that having more marginals results in increased noise in each. Hence, the authors decided to select the minimum number of marginals needed. PrivBayes [173] selects these marginals by building the network. The first node of the network is randomly assigned an attribute. The Exponential algorithm (or mechanism) is then used to select the remaining attributes one by one. Typically, the mutual information metric would be used to find the most correlated marginals. However, in the setting of differential privacy, mutual information has very high sensitivity. So the authors [173] introduced a function similar to mutual information. After determining the marginals, the second step is to add Laplacian noise in order to satisfy differential privacy.

A paper by McKenna et al [93] uses the Markov Random Field (MRF) graphical model instead of a Bayesian Network. Unlike a Bayesian Network, the Markov Random Field uses undirected graphs. Every edge u, v has the joint probability distribution $P[v, u]$. MRF permits both cycles and cliques. This will inevitably make the structure more complicated; thereby making the making the sampling process harder. However, these complicated structures can be used to capture higher dimensional correlations. Here the sampling process

works by joining cliques into nodes and creating a junction tree structure [93] [175]. McKenna et al [93] doesn't provide a way for automatically determining the structure of the graph.

Another more recent paper [175] proposes a new method called PrivSyn for generating synthetic data without using graphical models to represent the dataset. Instead PrivSyn uses a set of large number of low-degree marginals to represent the dataset. The authors [175] distinguish PrivSyn from the previous approaches by pointing out that graphical models are parametric while PrivSyn is a non-parametric approach. An important limitation of the previous two graphical model-based approaches [173] [93] is that once the model structure is fixed, it places conditional independence assumptions that may not exist in the original dataset. The PrivSyn approach mitigates this problem by not using graphical models. Another limitation of the PrivBayes approach [173] is that the low-sensitivity proxy used instead of mutual information has high computational complexity ($O(n^2)$ where n is no. of users). PrivSyn proposes a novel metric called Independent Difference (InDif) to efficiently measure correlation levels among attributes. InDif is meant to mitigate this problem of sensitivity vs computational complexity.

When trying to obtain differentially private marginals, there are two major error sources: information loss and noise error. Information loss happens when some important marginals don't get selected. This can happen in PrivBayes [173] because as mentioned it chooses a limited number of marginals. In PrivSyn, the authors [175] try to mitigate this by using this InDif method. Consider two attributes a and b. InDif is the l_1 distance between the $M_{a,b}$ marginal and and the 2-way marginal calculated assuming independence $M_a \times M_b$. The authors suggest that InDif is a more efficient way to capture the correlation information between attributes; thereby enabling the selection of more appropriate marginals. After selecting the appropriate marginals, the authors added weighted Gaussian noise to privatize the dataset. Afterwards they use the Gradually Update Method (GUM) to generate the synthetic dataset iteratively by matching it to the target set of marginals. The details for GUM can be found in the PrivSyn paper [175]. There are other synthetic data generators of that use graphical models or marginals as well [174] [89] [13] [79].

2.4.4 Synthetic Data Generation Using GANs

Generative Adversarial Networks (GAN) have also been used as synthetic data generators. A GAN consists of two competing networks: a generator that creates artificial data and a discriminator that attempts to distinguish this artificial data from real data. Through this continuous competition, the generator learns to produce increasingly realistic outputs. Early GAN research focused heavily on image generation, which is not the focus of this thesis.

Choi et al [22] first proposed an approach to generate realistic synthetic patient records (i.e. discrete tabular data) using GANs. They proposed medical Generative Adversarial Network (medGAN), that can generate high-dimensional discrete variables by coupling a GAN with an autoencoder. The autoencoder assists the original GAN to learn the distribution of multi-label discrete variables by generating continuous distributed representations of patient records rather than outputting discrete records directly.

Similarly, Park et al [106] introduced a framework called TableGAN that can accurately capture inter-column relationships. TableGANs consist of three neural networks: generator, discriminator and classifier. The generator and discriminator perform the same function as in a typical GAN. The newly added classifier network increases semantic integrity by eliminating semantically incorrect records. The classifier is trained using the labels from the original source table. This training allows it to learn correlations between specific labels and other attribute values. During the generation process, the classifier evaluates the synthetic records to determine if they are semantically correct. TableGAN also introduced the concept of model compatibility as a primary evaluation metric. Model compatibility is the idea that machine learning models trained using carefully synthesized tables show behavior similar to that of models trained using the original table.

Tabular data often contain a mixture of continuous and discrete variable columns. Xu and Veeramachaneni [165] proposed a method to handle continuous variables with complex distributions. Their framework called TGAN (Tabular GAN) introduced mode-specific normalization, a preprocessing technique where continuous variables are modeled using a Gaussian Mixture Model (GMM). This was important because real-world continuous data often do not follow a simple Gaussian distribution; they are often multimodal, with probability mass concentrated around specific values. In TGAN, the generator does not output a single scalar value for a continuous column. Instead, it outputs a probability distribution over the GMM modes and a scalar value within that mode. For discrete variables, TGAN uses the Gumbel-Softmax distribution to allow for differentiable sampling of categorical data.

Class imbalance can be a significant issue for discrete columns. This is because, in a standard GAN, the generator minimizes the global distance between distributions, often ignoring rare events that are statistically insignificant but semantically critical. Xu et al [164] first proposed using conditional GANs for synthetic data generation to mitigate this issue. The authors used mode-specific normalization to overcome the non-Gaussian and multimodal distributions in continuous columns. In order to deal with the imbalanced discrete columns, they designed a conditional generator and a training-by-sampling mechanism.

This mechanism allows the training process to explicitly condition the generator on specific discrete categories. If a dataset contains a rare category, CTGAN's sampler ensures that the generator is frequently asked to produce rows matching that category, preventing the discriminator from simply rejecting minority classes. A modified conditional GAN variant called CTAB-GAN+ was proposed by Zhao et al [176].

GANs are prone to overfitting where the density of the learned generative distribution concentrates on the training data points. This results in the generator reproducing exact copies of training examples. To address this, the research direction shifted toward integrating differential privacy (DP) into the GAN training framework. Xie et al [163] proposed the differentially private GAN (DPGAN) model, in which differential privacy is achieved by adding carefully designed noise to the discriminator's gradients during the training procedure. Because of the post-processing property of differential privacy, the generator's parameters inherit the privacy guarantee from the discriminator [163]. Torkzadehmahani et al [149] extended this and created Differentially Private Conditional GAN (DP-CGAN). DP-CGAN generates both synthetic data and corresponding labels and leverages Renyi differential privacy (RDP) accountant to track the spent privacy budget. RDP is a new method for quantifying the privacy guarantees of algorithms that process sensitive data introduced by Mironov [96].

An alternative to gradient perturbation is the Private Aggregation of Teacher Ensembles (PATE) framework. A paper titled "PATE-GAN: Generating Synthetic data with Differential Privacy guarantees" by Jordan et al [67] explores this idea. Instead of a single discriminator, the framework trains an ensemble of k teacher discriminators, each on a disjoint partition of the sensitive data. When the generator produces a sample, the teachers vote on whether it is real or fake. The consensus vote is perturbed with Laplacian noise before being revealed to the generator (i.e. student). Privacy cost is incurred only when the teachers vote. If the teachers overwhelmingly agree, the privacy cost is low [67]. Long et al [87] identified scalability issues in PATE-GAN when dealing with high-dimensional data. They introduced G-PATE, which utilizes a gradient aggregation mechanism. Instead of voting on labels, G-PATE aggregates the gradients of the teacher discriminators.

Mukherjee et al [97] proposed the PrivGAN architecture designed to defend against membership inference attacks while maintaining high data utility. Standard GANs tend to memorize their training datasets, which leaves them vulnerable to membership inference attacks. Existing privacy-preserving alternatives (such as DPGANs mentioned earlier) suffered from severe degradation in synthetic sample quality. PrivGAN introduces a privacy discriminator that tries to infer membership of the training data. In PrivGAN, the train-

ing dataset is randomly partitioned into N non-overlapping, equal-sized subsets in order to train N generator–discriminator pairs. The privacy discriminator is trained to identify which specific generator produced a given synthetic sample. The generators are optimized to simultaneously fool their standard discriminator and the privacy discriminator. This structure acts as a regularizer that prevents the generators from memorizing the specific samples in the training data.

Gueye et al [58] proposed the Row Conditional Tabular Generative Adversarial Network (RC-TGAN) architecture that extends the tabular GAN to support modeling and synthesizing relational databases. The core innovation is conditioning the generation of the child table on the features of the parent table. When generating a child, the generator takes the feature vector of the parent as a condition [58]. Agren and Sosa [2] proposed another architecture, the Hierarchical Conditional Tabular GAN (HCTGAN), to synthesize multi-tabular data from complex multi-tabular datasets. They addressed datasets with deep hierarchical structures. HCTGAN introduces a hierarchical training algorithm where models are trained sequentially down the hierarchy chain. It uses a novel sampling algorithm that enforces referential integrity, ensuring that every foreign key generated in a child table corresponds to a valid primary key in the synthesized parent table [2].

Standard GANs learn correlations, not causation. This can lead to synthetic data that is statistically valid but causally impossible. Wen et al [161] proposed the Causal-TGAN; a causally-aware generator architecture that can capture the relationships among variables by explicitly modeling the pre-defined inter-variable causal relationships. The generator architecture is structured according to the causal graph. Variables are generated in topological order; a variable is generated only after its causal parents have been generated, and it is conditioned on those parents. This method can handle discrete, continuous, and mixed data types [161]. However, the training process in Causal-TGAN remains identical to standard GANs, which relies only on distinguishing real from synthetic samples without encouraging causal alignment. Therefore, the Causal-TGAN's generator may not generate synthetic data that fully captures the causal relationships in the original data. Nguyen et al [99] proposed CA-GAN (Casually Aware GAN), a robust framework that integrates causal awareness in both training and generation phases.

Jia et al [65] integrated marginal-based methods (discussed in sub-section 2.4.3) into a GAN framework. Pure GANs suffer from severe utility degradation when trained under differential privacy. On the other hand, marginal-based differentially private data synthesis methods, such as PrivBayes [173], achieve high utility but rely on low-dimensional structural assumptions. These structural assumptions are difficult to verify and to adaptively

match to the true unknown degree of low-dimensionality. To resolve these issues, PrAda-GAN [65] integrates GANs with low-dimensional structural modeling. The framework utilizes an autoregressive approach with d sub-generators to model the joint data distribution. Each sub-generator captures the conditional distribution of a variable given its parent nodes, represented as a directed acyclic graph in a Bayes network. To dynamically learn the network structure, PrAda-GAN introduces adaptive regularization via a group lasso penalty applied to the generator's objective function. This penalty implicitly recovers the underlying Bayes network by shrinking the weights associated with unimportant features to zero. To ensure privacy, the discriminator is trained using Differentially Private Stochastic Gradient Descent. The authors of PrAda-GAN establish diminishing bounds on the parameter distance, variable selection error, and Wasserstein distance between the generated and true data distributions. They demonstrate that leveraging dependency sparsity significantly improves the convergence rate.

Synthetic data must not propagate or amplify societal biases present in the real data. Rajabi and Garibay [119] introduced TabFairGAN, a fairness-aware GAN that includes a modified loss function with a fairness constraint. It utilizes a second discriminator (or a modification of the primary one) that specifically checks for discrimination metrics in the synthetic data. Sarmin et al [129] proposed PF-WGAN, which attempts to satisfy both differential privacy and fairness simultaneously. Kotal and Joshi [73] addressed the issue of implicit rules in data. Synthetic data generation methods often fail to capture complex, implicit rules that relate different elements of the data. For example, in healthcare, prescription guidelines often dictate that certain medications should not be prescribed to people with certain medical conditions. The ContextGAN framework proposed by Kotal and Joshi [73] utilizes a constraint matrix that encodes these types of explicit domain rules. This matrix is integrated into the discriminator. If a generated sample violates a rule in the constraint matrix, the discriminator penalizes it heavily (or rejects it outright).

Time-series Generative Adversarial Networks (TimeGAN) were introduced by Yoon et al [167] as a way to preserve temporal dynamics while modelling. Standard autoregressive models accurately capture conditional distributions for sequence prediction but are fundamentally deterministic. Conversely, typical sequential GAN applications attempt to model the joint distribution directly but rely solely on binary adversarial feedback. This unsupervised feedback is often insufficient to ensure the network captures the stepwise temporal dependencies present in the training data. In order to effectively model the temporal dynamics of time series, the authors of TimeGAN [167] augment the standard unsupervised adversarial loss with a stepwise supervised loss. This explicitly forces the model to learn the

conditional distributions present in the original data. By extracting transition dynamics directly from real sequences, the framework leverages richer training signals than binary adversarial classifications alone provide. The authors also introduce an embedding network to create a reversible mapping between latent representations and features, thereby reducing the high dimensionality of the adversarial learning space. This takes advantage of the fact that temporal dynamics of complex systems are often motivated by fewer and lower-dimensional factors of variation. Simultaneous training of the embedding and generator networks minimizes the supervised loss. This joint optimization ensures the latent space improves parameter efficiency and explicitly helps the generator model temporal correlations.

Lin et al [80] addressed the complexity of using GANs for sharing networked time series data. Time series data often have static metadata (e.g., User ID, Region) that influences the dynamic time series. When dealing with such data, standard GAN approaches have fidelity and privacy issues. The authors designed a custom workflow called DoppelGANger [80] that decouples the generation of metadata and time series. It first generates the metadata, then conditions the time-series generation on that metadata. The authors demonstrate that this achieves better fidelity than baseline models. Deng et al [45] combined TimeGAN with privacy-preserving stability mechanisms and introduced a gradient penalized Conditional Time-series Generative Adversarial Network with a Multi-Scale Importance-aware Network (CTGAN-MSIN). CTGAN-MSIN is a hybrid framework combining data augmentation and feature classification to address fault diagnosis in rotating machinery under imbalanced data conditions. The approach aims to resolve training instability and mode collapse in standard TimeGAN architectures while improving the extraction of critical multi-scale features. Details of the architecture can be found in the CTGAN-MSIN paper by Deng et al [45].

2.4.5 More Synthetic Data Generators

So far, synthetic data generators that rely on graphical models, copulas, marginals, and Generative Adversarial Networks (GAN) have been explored. There are other synthetic data generation methods that do not rely on any of these. Variational Autoencoders (VAEs) perform synthetic data generation by encoding the heterogeneous, mixed-type input features into a lower-dimensional, continuous latent probability space, and subsequently reconstructing the data through a probabilistic decoder network. Abay et al [1] proposed DP-SYN a autoencoder architecture that assumes a supervised learning context and deterministically splits the training data according to existing class labels. They train a separate ded-

icated autoencoder for each class label utilizing Differentially Private Stochastic Gradient Descent. To synthesize data, it samples intermediate continuous representations from the latent space using a Differentially Private Expectation Maximization algorithm. The respective decoders then translate these representations back into the original tabular feature space. Gupta and Sethi [59] proposed another method that utilizes VAEs.

Diffusion models can also be used for synthetic data generation. They operate by progressively distorting original data by injecting Gaussian noise over a series of sequential timesteps (the forward process). A deep neural network is then trained to completely reverse this process by predicting and systematically removing the noise to recover the original data distribution (the reverse process). Because diffusion models rely on a fixed, incredibly smooth trajectory of Gaussian transitions, they provide training stability. This stability makes them highly suitable for strict differential privacy constraints. For instance, TableDiffusion [151] explicitly trains the network to predict the precise mathematical noise added at each timestep, rather than attempting to reconstruct mixed-type discrete variables directly. Another architecture DP-Fed-FinDiff [130] extend diffusion models into federated learning frameworks for the financial sector.

Castellon et al [19] proposed DP-TBART, an autoregressive, Transformer-based model specifically designed for tabular data synthesis under differential privacy. A transformer is a neural network architecture that relies on self-attention mechanisms to process sequential data by dynamically weighing the importance of each input element [155]. DP-TBART uses a three-layer decoder-only transformer architecture and is trained using DP-Adam (a differentially private optimization algorithm) to ensure privacy. Details can be found in the paper by Castellon et al [19]. Tiwald et al [148] proposed Tabular Auto-Regressive Generative Network (TabularARGN) - another autoregressive architecture designed to handle mixed-type, multivariate, and sequential datasets. It natively handles mixed data types by mapping all features into discrete categorical sub-columns and employing an any-order autoregressive architecture with a permutation masking layer to learn conditional probabilities for arbitrary feature subsets. It efficiently enforces differential privacy by combining Differentially Private Stochastic Gradient Descent with an early stopping mechanism that halts training when validation utility plateaus [148].

2.4.6 Evaluation metrics

This section explores the various evaluation/performance metrics that have been used to evaluate synthetic data generators. Broadly speaking, there are three aspects that need to be evaluated: utility, privacy and computational cost.

2.4.6.1 Utility

Utility measures are metrics that provide an understanding of the differences between the probability distributions of the original dataset and the generated synthetic dataset. If the synthetic and original datasets are similar, greater utility is assigned . Distinguishability metrics are popular utility metrics. One such well-known distinguishability measure is the propensity score. Propensity score is measured by creating a classification model that represents the probability of a given record being in either the original or the synthetic dataset [35]. The classification model would be capable of telling the difference between real and synthetic datasets records. If the utility value is high, the implication is that the model is unable to perform the distinction. The classification model is created by merging the original and synthetic datasets into one group, then assigning a binary indicator (0 or 1) to every record depending on whether the given record is real or synthetic. The propensity score $\hat{p}_i$ is calculated as a probability value for all of the records in the combined dataset. The mean squared difference between these estimated probabilities and the true proportion of records from the masked data in the combined dataset provides the utility statistic [35]. This utility statistic propensity score mean-squared error ($pMSE$) can be determined as follows.

$$pMSE = \frac{1}{N}\Sigma_i(\hat{p}_i - 0.5)^2 \tag{2.18}$$

where N is the size of the combined dataset.

A propensity score of 0 indicates no difference between the datasets. This is usually the result of the synthetic data generator overfitting with the original dataset and generating a fake that is identical to the original (resulting in $\hat{p}_i = 0.5$ for every entry). On the other hand, if both datasets are fully distinguishable, a propensity score of 1 for synthetic rows and 0 for original rows will be obtained. This results in an overall score of 0.25 [35].

Histogram Intersection (HI) and Kullback-Leibler Divergence are two other metrics for measuring the statistical similarity between two distributions. HI measures the distance between two histograms A and B as follows. If HI is higher, the datasets are more similar.

$$HI(A,B) = \Sigma_i \min(A(i), B(i)) \tag{2.19}$$

where $A(i)$ and $B(i)$ are values of the i^{th} bins of the histograms of A and B respectively.

The KL divergence of a distribution Q from a distribution P can be defined as follows.

$$KL(P||Q) = \Sigma_i P(i)\log\left(\frac{P(i)}{Q(i)}\right) \tag{2.20}$$

Here $P(i)$ and $Q(i)$ are the probabilities associated with the i^{th} event in the P and Q distributions respectively. A KL divergence value of zero indicates that P and Q are identical.

2.4.6.2 Privacy

Stadler et al [139] defines a metric called Privacy Gain. This can be used in addition to the theoretical DP guarantee discussed earlier.

The privacy gain of publishing a synthetic dataset S in place of raw data R for target record r_t, as the reduction in the adversary's advantage when given access to S instead of R, is given as follows.

$$PG = Adv(R, r_t) - AdV(S, r_t) \tag{2.21}$$

2.4.6.3 Computational Cost

This is simply calculating the CPU time taken to complete adding DP noise and to do the sampling process. The proposal for this thesis which will be covered in the following section involves using arithmetic circuits to speed up the sampling process. Hence calculating the computational cost would be relevant.

2.5 Conclusion

This chapter provided a comprehensive background and literature review for my thesis on synthetic data generation. I provided detailed background on different types of probabilistic graphical models (PGM) including Bayesian networks (BN), Markov models, factor graphs and arithmetic circuits. All of these models are relevant to my research. The PGM used as the generative process for the proposed synthetic data generator were BNs. They can be directly examined to determine the statistical properties of the data they generate. Also their directed acyclic graph structure can be leveraged to explore causal relationships among variables. The proposed synthetic data generation process involves converting the PGM into an arithmetic circuit. However, the conversion process used here works for any factor graph - which is a generalized representation that covers many kinds of probabilistic models including BNs. Hence factor graphs and arithmetic circuits are relevant to my research as well. The background section also covered inferencing, parameter learning and structure learning for PGMs.

The literature review section of this chapter included sections on sampling of PGMs, privacy preservation of sensitive data (via differential privacy) and various publicly available synthetic data generation techniques. Different independent and dependent samplers were considered here. The need for cost-efficient independent sampling in synthetic data generation was also discussed. The section for differential privacy (DP) included different

types of DP, different algorithms for DP and different properties of DP. Reducing computational cost is an important motivation for DP as well. Finally, existing literature on synthetic data generators were examined. This included covering publicly available generators like DataSynthesizer [112] and Synthetic Data Vault [107] (available as Python libraries) as well as prominent synthetic data generation methods like PATE-GAN [104] [105] and PrivSyn [175]. The literature review also included some prominent evaluation metrics used to test synthetic data generators.

The next three chapters of the thesis present the full scope of my research. First, I will propose two novel methods to draw independent samples from PGMs. The empirical evaluation of their performance provided will demonstrate the reduction in computational cost when compared to other prominent samplers. The next chapter will describe a novel decomposition scheme for reducing the computational cost of Laplacian noise perturbation process in DP. Noise generation time and accuracy of the novel decomposition scheme are evaluated by comparing the proposed method against the standard version without the novel decomposition scheme. The reduction in noise generation time is shown. Privacy gain is evaluated by comparing the synthetic data generated using this method against prominent state-of-the-art baselines. Finally, the proposed synthetic data generator will be described as a whole. It will be evaluated against DataSynthesizer and Synthetic Data Vault in terms of data generation time and referential integrity.

CHAPTER 3

SAMPLING OF LARGE PROBABILISTIC GRAPHICAL MODELS USING ARITHMETIC CIRCUITS

3.1 Overview

A synthetic data generation approach involves designing a generative model that captures the important statistics of the reference dataset and then sampling from that model. Sampling a probabilistic model involves drawing samples from the model, possibly conditioned on assumptions about states of some random variables. Importantly, as the number of samples approaches infinity, the empirical distribution of the samples approaches the distribution defined by the model. Samplers are broadly classified as dependent and independent. Independent sampling refers to the process of drawing individual samples from a model in such a way that each sample is generated independently of the others. Dependent sampling methods involve drawing a sequence of samples, where adjacent pairs of samples are statistically related. Details of different dependent and independent sampling methods are provided in section 2.2.

Independent sampling methods are computationally costly because each sample is generated from scratch without using information from previous samples. Every sample requires a full evaluation of the target probability density or likelihood, which can be expensive, especially when dealing with high-dimensional data. For synthetic data generation this computational cost may be impractically inefficient. Typically, faster dependent sampling methods are not suitable for synthetic data generation because the sampler can in-

troduce additional statistical relationships, beyond that which is in the model. This would make it difficult to determine whether a relationship apparent from the data is the effect of the model or the sampling algorithm. Synthetic data generation also requires a sampler capable of conditioning. This is required to model the various relationships in multi-table relational datasets.

Developing and evaluating a novel method to efficiently generate independent samples from probabilistic graphical models with impractically large supports will form the first part of this research. This method is a variation of inverse transform sampling. The proposed method may be applied to any factor graph that can be compiled to an arithmetic or probabilistic circuit. Here the method is demonstrated by transforming the factor graph into an arithmetic circuit with the smooth and decomposable properties. The following experiments evaluate the proposed method for large probabilistic graphical models (PGMs) with discrete distributions and a finite number of states. This part of the research was published in a conference paper I authored [141].

Section 2.1.1.4 defined factor graphs and their attributes. By blurring the distinction between a single random variable and a collection of random variables, $\mathbb{D}(X)$ was obtained. $P(X = x)$ is the probability of the random variables X being in state $x \in \mathbb{D}(X)$. Following this, the equations for the probability of state $P(X = x)$ (equation (2.2)), the likelihood function $\phi(x)$ (equation (2.3)) and the partition function z (equation (2.4)) were obtained.

The likelihood function ϕ can then be represented as an arithmetic circuit using the specific formulation described here. As mentioned previously, an arithmetic circuit η is a function of indicator variables Λ. An indicator variable $\lambda_{i,j} \in \Lambda$ describes the state $X_i = x_{i,j}$. Thus, the total number of indicator variables is equal to the sum of the domain sizes of all random variables. The likelihood of $X = x$ can be calculated using an arithmetic circuit by providing the input Λ, where

$$P(X = x) = \frac{1}{z}\phi(x) \tag{3.1}$$

$$\phi(x) = \eta(\lambda_{1,1}, \lambda_{1,2}, \ldots, \lambda_{2,1}, \lambda_{2,2}, \ldots)$$

$$\lambda_{i,j} = \begin{cases} 1 & \text{if } X_i = x_{i,j}, \\ 0 & \text{otherwise.} \end{cases}$$

Simplifying notation, write $\eta(\Lambda[\alpha])$, such that

$$\Lambda[\alpha] = (\lambda_{1,1}, \lambda_{1,2}, \ldots \lambda_{2,1}, \lambda_{2,2}, \ldots) \tag{3.2}$$

$$\lambda_{i,j} = \begin{cases} 0 & \text{if } \alpha \implies (X_i \neq x_{i,j}), \\ 1 & \text{otherwise.} \end{cases}$$

In this context, α denotes the logical condition ($P(x|\alpha)$) and $\Lambda[\alpha]$ specifies how to configure the arithmetic circuit's indicators to enforce α. From this, it can be surmised that $\alpha \implies (X_i \neq x_{i,j})$ is true only if α mandates that the random variable X_i cannot be in state $x_{i,j}$.

In order to effectively calculate the likelihood function of an arithmetic circuit for any marginalisation and evidence, the circuit must have the *smooth* and *decomposable* properties [39]. If the circuit is smooth and decomposable, then it can be directly used to calculate marginal probabilities in space and time linear to the size of the circuit. This can be generalized to any grouping of partial marginals. Let $s_i \subseteq \mathbb{D}(X_i)$ be some possible states. If there is s_i for each random variable X_i then

$$\begin{aligned} &P(X_1 \in s_1, X_2 \in s_2, \ldots) \\ &= \sum_{x_{1,j} \in s_1} \sum_{x_{2,j} \in s_2} \ldots P(X_1 = x_{1,j}, X_2 = x_{2,j}, \ldots) \end{aligned} \tag{3.3}$$

The arithmetic circuit equivalent computation is

$$P(X_1 \in s_1, X_2 \in s_2, \ldots) = \frac{1}{z}\, \eta \left(\Lambda[X_1 \in s_1, X_2 \in s_2, \ldots] \right) \tag{3.4}$$

A standard conditioned marginal probability is defined by assigning a unique value to the conditioned variable X_i (i.e., $|s_i| = 1$) and summing over the entire domain of X_j (i.e., $s_j = \mathbb{D}(X_j)$); thus, marginalisation is a special case of (3.4).

By applying (3.4), the conditional marginal distribution of X_i can be computed using $|\mathbb{D}(X)|$ calls to the circuit.

The computational complexity of calculating marginals from an arithmetic circuit can be improved further by using the differential approach developed by Darwiche [39] for calculating marginals. Using this approach, it is possible to calculate all marginal distributions with just two passes through the circuit. The first pass *up* is used to determine the result as explained above. The second pass *down* calculates the partial derivatives at every node [39].

The differential approach [39] requires the arithmetic circuit to be deterministic and decomposable. After the circuit is built, the observed evidence can be inserted by assigning zero to parameters that contradict the evidence and then do a straightforward bottom-up pass to compute the likelihood of the evidence, $P(e)$. The key idea for extracting posterior

marginals lies in taking partial derivatives of this circuit with respect to its parameters. For each CPT parameter θ, the derivative $\frac{\partial P(e)}{\partial \theta}$ effectively measures the expected count of the parent-child configuration associated with θ given the evidence. By doing a single forward pass to compute $P(e)$ and then a backward pass to compute these derivatives, all marginal probabilities can be derived.

3.2 Proposal

3.2.1 Theoretical Framing

The content for this section is taken from my published paper [141]. It is well known that one may generate a random sample from an arbitrary distribution using *inverse transform sampling*. Given a probability distribution p over ordered domain D with cumulative distribution $c(x) = \Sigma_{y \leq x; y \in D}\ p(y)$, the method works by drawing a uniform random variate u in $[0,1]$. The desired sample is x that minimises $c(x)$ subject to $c(x) \geq u$.

Given a factor graph, we order its domain by providing an arbitrary ordering over its random variables X, and providing an arbitrary ordering over the possible states of each random variable. Labelling each state of random variable X_i to be $0, 1, 2, 3, \ldots, m_i$, the ordered domain of the factor graph with n random variables, $X = (X_1, X_2, X_3, \ldots, X_n)$, is $(0,0,0,\ldots, 0), (1,0,0,\ldots,0), \ldots, (m_1,0,0,\ldots,0), (0,1,0,\ldots,0), \ldots, (m_1, m_2, m_3, \ldots, m_n)$.

The number of possible joint states is exponential in the number of random variables, so practical methods are provided to find x that minimizes $c(x)$ subject to $c(x) \geq u$ using a decomposable and smooth arithmetic circuit representing the factor graph.

Recall η is an arithmetic circuit representing the likelihood function of a factor graph. Define ζ as the cumulative function of η as follows, using the ordered domain D,

$$\zeta(x) \stackrel{\text{def}}{=} \sum_{y \leq x; y \in D} \eta(\Lambda[y]) \tag{3.5}$$

where $\Lambda[y]$ represents the indicator values consistent with joint states y. Abusing notation, a conditioned version can be defined,

$$\zeta(x|a) \stackrel{\text{def}}{=} \sum_{a < y \leq x; y \in D} \eta(\Lambda[y]) \tag{3.6}$$

Definition (3.6) means $\zeta(x) = \zeta(a) + \zeta(x|a)$. Let $x = (x_0, x_1, \ldots, x_n)$ be a joint state in D and consider the last random variable X_n. Note that x represents a vector of states, one for each random variable. Define $A[\alpha] \in D$ to be the joint state with all zeros except for random variables and states α, e.g., $A[X_n = x_n] = (0,0,0,\ldots,x_n)$. Now $\zeta(x) = \zeta(A[X_n = x_n]) + \zeta(x|A[X_n =$

$x_n]$). Thus $\zeta(A[X_n = x_n])$ is the component of $\zeta(x)$ from states $(0,0,0,\ldots,0)$ to $(0,0,0,\ldots,x_n)$ inclusive and $\zeta(x|A[X_n = x_n])$ is the component of $\zeta(x)$ beyond states $(0,0,0,\ldots,x_n)$, continuing up to $(x_0,x_1,x_2,\ldots,x_n)$.

However, $\zeta(A[X_n = x_n]) = \eta(\Lambda[X_n < x_n])$ because $\Lambda[X_n < x_n]$ represents X_n being in state less than x_n and all other random variables before X_n being fully summed out. Thus, taking $\eta(\Lambda[X_n < 0]) \stackrel{\text{def}}{=} 0$ yields

$$\zeta(x) = \eta(\Lambda[X_n < x_n]) + \zeta(x|A[X_n = x_n]) \tag{3.7}$$

This can be generalised to an arbitrary random variable X_i. Let $e_i = [X_j = x_j]_{j>i}$ be the known states in x of random variables ordered after X_i. As $\zeta(x|A[e_i])$ partitions D at $A[e_i]$ and states in D that agree with e_i are contiguous, it follows that,

$$\zeta(x|A[e_i]) = \eta(\Lambda[e_i, X_i < x_i]) + \zeta(x|A[e_i, X_i = x_i]) \tag{3.8}$$

Recursively applying (3.8) and (3.7) leads to

$$\zeta(x) = \sum_i \eta(\Lambda[e_i, X_i < x_i]) \tag{3.9}$$

Thus the cumulative distribution function $c(x)$ may be calculated using n calls to the arithmetic circuit (plus one call to determine z if needed). So the samples can be generated in time linear in the number of random variables and circuit size, even when conditioned on evidence. This is the key advantage of this method. Prior independent sampling methods are only linear in time for distributions without conditioning.

3.2.2 The Method

The method proceeds, as in Algorithm 1, one random variable at a time, $X_n, X_{n-1}, \ldots, X_i, \ldots, X_1$, accumulating the terms of equation (3.9).

First determine z by using the defined arithmetic circuit and then draw random variate u uniformly from $[0,z]$, the real interval from 0 to z inclusive. Rather than accumulating the sum, u is decremented according to the terms of (3.9) each time $X_i = x_i$ is determined. For each random variable X_i, determine the largest x_i subject to $\eta(\Lambda[x, X_i \leq x_i]) \leq u$ where x is the accumulated set of $X_i = x_i$ pairs.

Step 5 of Algorithm 1 is potentially expensive for large random variables, naively requiring $|\mathbb{D}(X_i)|$ calls to η. However, it is a classic one-dimensional search task and many efficient methods are available. Again it should be pointed out that this is true even for distributions with conditioning. Also note that the call to η in Step 6 can sometimes be avoided

by caching results from Step 5. For Boolean random variables, Steps 5 and 6 collectively require a single call to η, namely $\eta(\Lambda[x, X_i = 0])$.

It is trivial to extend Algorithm 1 to draw samples from a joint probability distribution conditioned on evidence. Simply replace Step 5 for a conditioned random variables X_i and instead set x_i from evidence.

Algorithm 1: Proposed method for circuit based sampling

Data: An arithmetic circuit computing likelihood function η, and an ordered sequence of n random variables
Result: A sampled joint state x

1 Let x be an empty set to receive $X_i = x_i$ pairs for the sample result
2 Let $z = \eta(1, 1, 1, \ldots, 1)$
3 Let u be a uniform random variate in range $[0, z]$
4 **for** *i from n down to 1* **do**
5 Find largest x_i where $\eta(\Lambda[x, X_i \leq x_i]) \leq u$
6 Subtract $\eta(\Lambda[x, X_i \leq x_i])$ from u
7 Include $X_i = x_i$ into x
8 **end**
9 **return** x

3.2.3 Marginals Method Variation

As mentioned, it is possible to determine conditioned marginal distributions of all random variables in two passes though the arithmetic circuit. An *up* pass computes η and a *down* pass computes $\nabla\eta$.

This fact can be used to replace Steps 5 and 6 with a single call to η and $\nabla\eta$. This may be further improved using auxiliary data structures to restrict computation to the required marginals for a *down* pass. In the experiments reported here, such restricted computations were not implemented.

3.2.4 Theoretical support for circuit based sampling

This sampling research was published in my earlier sampling paper [141]. My research makes certain mathematical and structural assumptions: (1) The arithmetic circuit η must represent a non-negative unnormalized likelihood function over a discrete, finite, and ordered domain. (2) The baseline likelihood polynomial circuit must be tractable for inference. (3) Evaluating cumulative probabilities over the ordered domain must remain computationally efficient. These specific conditions must be met to guarantee that evaluating

the cumulative partial likelihoods, such as $\eta(\Lambda[e_i, X_i < x_i])$ maintains the compact representation necessary for the proposed algorithm to run efficiently. Subsequent publications by Broadrick [11] [10] provide theoretical support and validate these assumptions.

The paper "Polynomial Semantics of Tractable Probabilistic Circuits" by Broadrick, Zhang, and Van den Broeck [11] proves that different polynomial encodings of distributions are equivalent in expressive efficiency. Arithmetic circuits and probabilistic circuits efficiently compute multilinear polynomials. These polynomials represent multivariate probability distributions. Examples of such polynomials include likelihood polynomials, network polynomials, Fourier transforms and generating functions. The authors [11] examine the relationships between circuit representations of these polynomials. The authors prove that for distributions over binary variables, each of these probabilistic circuit models is equivalent in the sense that any circuit for one of them can be transformed into a circuit for any of the others with only a polynomial increase in size.

The arithmetic circuit (described in section 3.1) computing the likelihood function η corresponds to the likelihood polynomial (described by [11]). The likelihood polynomial directly computes the probability mass function whereas the network polynomial uses additional input variables and a special structure to do marginal inference. Also for binary variables, likelihood polynomial circuits can be transformed to network polynomial circuits with only a polynomial increase in size. The authors also prove that marginal probabilities on a circuit representing a likelihood polynomial can be computed in linear time. This justifies the assumption that the arithmetic circuit η is a theoretically tractable representation.

Another paper "Probabilistic Circuits for Cumulative Distribution Functions" by Broaderick et al [10] proves that finite discrete variables can be mapped to binary indicator variables while preserving elementwise order relations. The authors prove this using a method called "Less-Than Encoding" denoted by $LT_k(x)$. Because a CDF requires an ordered summation, the binary representation must respect the elementwise "less than or equal to" relation. Less-Than Encoding maps a discrete state x out of k possible states to a binary vector consisting of x ones followed by $k-x$ zeros. This specific encoding preserves ordinality. It formally guarantees that for any two states x and y, $x \leq y$ if and only if $LT_k(x) \leq LT_k(y)$. Since this method preserves the elementwise relation, the cumulative probability over the finite discrete domain matches the cumulative probability over the encoded binary domain exactly.

My proposed sampling method evaluates the cumulative function $\zeta(x)$ by imposing an arbitrary ordering over the possible states of each random variable. Step 5 of algorithm 1 re-

quires finding the largest x_i subject to the cumulative partial likelihood $\eta(\Lambda[x, X_i \leq x_i]) \leq u$. The Less-Than Encoding method proves that evaluating a cumulative distribution function based on the imposed ordering remains computationally tractable within the circuit. It proves that finite discrete variables can be mapped into the circuit in a way that respects the imposed ordering. The authors [10] demonstrate that circuits computing a Probability Mass Function (PMF) can be transformed into circuits computing a CDF in polynomial time. This establishes the theoretical tractability of computing cumulative probabilities directly within the circuit architecture.

3.3 Evaluation

In this work, I am introducing two algorithms (main method and marginals variation) designed to efficiently generate samples from a factor graph using knowledge compilation. To achieve this, I am using a custom knowledge compilation framework called *compiled-knowledge* (CK), which is implemented in Python 3 [126]. CK compiles the factor graph into an arithmetic circuit over floating-point types, using Darwiche's approach [20].

Next, the arithmetic circuit is compiled into a program comprising of a static array of memory locations and a sequence of instructions. The memory is used for output values, input values, temporary values, constant values and a stack. By compiling into a program, the compiler can apply further optimizations, producing a compact representation that runs efficiently on a virtual CPU.

Table 3.1: Workload Bayesian Networks and their sizes.

Size	**Bayesian Network**	**Random Variables**	**Parameters**	**Support (log2)**
small	asia	8	18	8
small	cancer	5	10	5
small	earthquake	5	10	5
small	sachs	11	178	17
small	survey	6	21	7
medium	alarm	37	509	54
medium	child	20	230	30
medium	insurance	27	1008	45
medium	mildew	35	540150	108
large	hailfinder	56	2656	107
large	hepar2	70	1453	82
very large	pathfinder	109	72079	190

3.3.1 Workloads

The workloads used for this evaluation were all discrete Bayesian networks. They were taken from the BNLearn repository. The repository classifies the Bayesian Networks into small, medium, large or very large. The networks, along with their random variable counts and parameters, are given in 3.1.

3.3.2 Samplers

Six samplers were used as baselines to compare the two proposed methods. Three of those baseline samplers are from publicly available Python libraries. Another three samplers are custom implementations of common algorithms. Overall, the eight samplers were:

1. **POM Gibbs**: The first is a Gibbs sampler available from the Python library 'Pomegranate' (with burn-in),

2. **BUGS Gibbs**: Next is the Gibbs sampler available in the MultiBUGS R library (with burn-in),

3. **CK M-H**: Custom implementation of a Metropolis-Hastings sampler using CK (with burn-in),

4. **PGM Reject**: Next is the rejection sampler available in the publicly available Pgmpy Python library (without burn-in),

5. **CK Reject**: Custom implementation of a rejection sampler using CK (without burn-in),

6. **CK Forward**: Custom implementation of a forward sampler for Bayesian Networks using CK. Since forward samplers typically don't have conditioning, this sampler had to implemet rejection sampling for this purpose (without burn-in),

7. **CK WMC**: This is the main proposed algorithm using weighted model counting, developed using CK (without burn-in),

8. **CK MAR**: This is the marginal method variation proposed algorithm, developed using CK (without burn-in).

In a standard Metropolis-Hastings approach, the naive practice of altering only one random variable per iteration leads to very high inter-sample correlation. The custom imple-

mentation ameliorates this by skipping over n samples for each sample used. Here n is the total no. of random variables.

3.3.3 Measures

The eight sampling algorithms were evaluated on the following two metrics:

1. Sampling Time

2. Accuracy of the empirical distribution compared to the model distribution

The sampling time testing for all eight samplers were carried out on Dell laptop with 16GB of RAM and an Intel Core i7-1185G7 CPU. Accuracy measurements were carried out on separate high-performance systems.

All the sampling algorithms in CK work by converting each workload into an WMC or MAR arithmetic circuit and then the circuit is compiled into a program. The sampler startup times reported include these compilation stages. Furthermore, each indirect sampler was configured with a 1,000-iteration burn-in period and no random restarts. Burn-in represented ≤ 1% of samples so is not reported separately.

Typically the accuracy of an empirical distribution would be calculated by using a formal metric such as Kullback-Leibler (KL) divergence. But such measures are not practical when dealing with large discrete supports. Therefore I developed the *factorised histogram-intersection*, described next.

The standard histogram-intersection method measures how much two histograms overlap by summing the smallest values in each corresponding bin. Each histogram represents a probability distribution. Often, the result is normalized (e.g., divided by the sum of one histogram's bins) so the score lies between 0 and 1, where higher values indicate greater similarity. [143].

In a Bayesian Network, the joint probability distribution is factorised into the product of a number of conditional probability tables (CPTs). Each CPT is a collection of rows, each row being a conditioned probability distribution. This motivates the measure of empirical distribution accuracy as the probability-weighted sum of histogram intersections. Thus, the factorised histogram-intersection for distributions P and Q is,

$$\mathrm{FHI}(P,Q) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{X, y=Y} P(Y=y)\ \mathrm{CHI}(P,Q,X|Y=y) \tag{3.10}$$

where,

$$\text{CHI}(P,Q,X|Y=y) \stackrel{\text{def}}{=} \text{HI}(P(X|Y=y), Q(X|Y=y)) \tag{3.11}$$

HI is the usual histogram-intersection,

$$\text{HI}(P,Q) \stackrel{\text{def}}{=} \sum_{x=X} min(P(X=x), Q(X=x)) \tag{3.12}$$

The value of n in (3.10) is the sum of $P(Y=y)$ for all CPT rows. However, this always equals the number of CPTs.

3.4 Results

As mentioned previously, the eight sampling algorithms (including two proposed) were evaluated based on two metrics: sampling time and accuracy. The results are therefore seperated into two sections. Section 3.4.1 explores the advantages provided by knowledge compilation on sampling time. This section also compares baseline samplers against the proposed samplers on the basis of sampling time. Section 3.4.2 details the accuracy of each sampler. The time taken to convert a Bayesian network into an arithmetic circuit and then into a program is included in the sampler startup time metric. CK M-H and CK WMC use a WMC program, whereas CK MAR uses a MAR program. These results have also been published in my paper [141].

3.4.1 Sampling Time

The sampling time measured as wall-clock time is broken down into two components: sampler startup time and time per sample.

Sampler startup time is the duration before the first sample is generated. It includes burn-in times for all MCMC samplers, as well as the compilation periods for CK Reject, CK M-H, CK WMC, and CK MAR samplers. Tables 3.2 and 3.3 show the sampler startup time for all samplers.

Time per sample indicates the average time period to generate each sample. This is obtained via linear regression over five runs conducted at five different sample sizes: 100,000, 150,000, 200,000, 250,000, and 300,000.

In each run, a distinct random seed is used to select the conditioning evidence. For every random variable in the network, one of its states is chosen at random. Then, 10% of

Table 3.2: Sampler startup time (in milliseconds) for small Bayesian networks.

Bayesian Network	**asia**	**cancer**	**earthquake**	**sachs**	**survey**
POM Gibbs	87.6	*failed*	*failed*	160	*failed*
BUGS Gibbs	80.0	51.62	41.1	175	94.0
CK M-H	1.00	0.557	0.376	1.5	0.619
PGM Reject	24.7	26.5	26.0	85.4	32.6
CK Reject	2.22	2.78	2.57	9.84	2.24
CK Forward	0.303	0.17	0.63	0.264	0.117
CK WMC	0.986	0.532	0.365	1.49	0.609
CK MAR	1.09	0.634	0.462	2.35	0.737

Table 3.3: Sampler startup time (in milliseconds) for medium, large and very large networks.

Bayesian Network	**alarm**	**child**	**insurance**	**mildew**	**hailfinder**	**hepar2**	**pathfinder**
POM Gibbs	*failed*	*failed*	*failed*	*failed*	*failed*	*failed*	*failed*
BUGS Gibbs	725	246	367	1050	1330	212	2400
CK M-H	3.74	463000	65.2	955	10700	17.1	91400
PGM Reject	210	143	293	1298310	499	464	>3600000
CK Reject	270	78.4	277	147280	533	399	>3600000
CK Forward	0.909	5.12	7.55	1553.1	30.6	5.3	>3600000
CK WMC	4.06	46.3	70.9	1150	10700	18.3	91400
CK MAR	7.40	46.3	177	2430	10700	28.8	91500

those chosen states (ensuring at least one) are given to the sampler as conditions. Tables 3.4 and 3.5 show the time per sample for all samplers.

The two proposed samplers (CK-WMC and CK-MAR) had the lowest startup time values for small sized networks. In medium, large and very large networks, all the CK samplers (when compared to the non-CK samplers) had very low startup time values. However, all the startup times were so small that many applications would consider them insignificant.

Tables 3.4 and 3.5 illustrate that both proposed methods achieve the best time-per-sample performance on smaller networks. On larger networks, CK M-H outperforms the others, highlighting the advantages of a WMC-based Metropolis-Hastings approach. For large and very large networks, CK-WMC ranks second in performance.

Both CK-WMC and CK-MAR are independent samplers, while Metropolis-Hastings and GIbbs are dependent samplers. Among independent samplers, CK-WMC and CK-MAR provided the fastest per-sample performance.

Table 3.4: Time per sample (in milliseconds) for small Bayesian networks. The best two sampling times are underlined.

Bayesian Network	asia	cancer	earthquake	sachs	survey
POM Gibbs	24.8	*failed*	*failed*	11.1	*failed*
BUGS Gibbs	0.0994	0.0141	0.0789	0.076	0.0555
CK M-H	0.0228	0.0152	0.0154	0.0421	0.018
PGM Reject	3.33	0.915	0.704	406	2.70
CK Reject	0.674	0.0536	0.0865	105	0.0653
CK Forward	0.142	0.251	0.0972	0.293	0.209
CK WMC	0.00367	0.00298	0.00284	0.0232	0.0034
CK MAR	0.00652	0.00322	0.0028	0.0478	0.00527

Table 3.5: Time per sample (in milliseconds) for medium, large and very large Bayesian networks. The best two sampling times for each workload are underlined. A dash (-) indicates that time per sample was greater than 1 second.

Bayesian Network	alarm	child	insurance	mildew	hailfinder	hepar2	pathfinder
POM Gibbs	*failed*	*failed*	*failed*	*failed*	*failed*	*failed*	*failed*
BUGS Gibbs	0.0399	0.023	5.88	94.1	0.529	0.61	52.7
CK M-H	0.421	0.154	4.02	74.4	2.16	1.59	30.1
PGM Reject	-	672	-	-	-	-	-
CK Reject	255	109	-	-	-	-	-
CK Forward	7.09	0.514	9.44	-	-	22.1	-
CK WMC	0.449	0.109	5.82	220	3.57	1.54	39.3
CK MAR	0.736	0.251	10.1	248	4.47	3.15	60.6

3.4.2 Accuracy

Accuracy values for small and medium workloads were computed for all samplers except for POM Gibbs, which failed on most workloads. Figure 3.1 illustrates the factorized histogram intersections across eight samplers. For each sampler, a single experimental run, consisting of 10,000 samples following a burn-in period of 1,000 samples, was conducted on every small and medium network. Note that even the factorized histogram-intersection method proved too computationally intensive to apply to large and very large networks.

There is no statistically significant variation in accuracy between the baseline samplers and the proposed methods. Note that intersample correlation was not evaluated. It is well known that Gibbs and Metropolis-Hastings have high intersample correlation whereas the other methods do not.

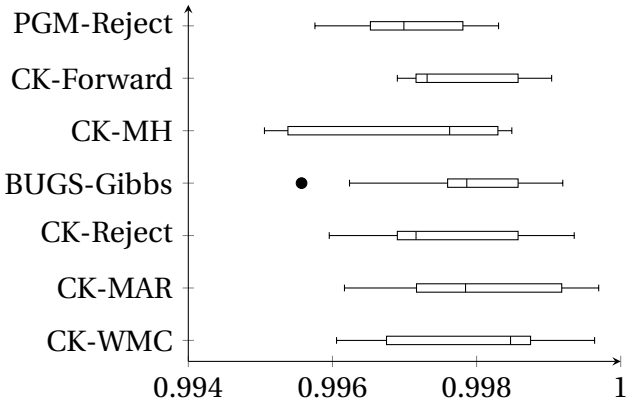


Figure 3.1: Accuracy for all 8 samplers

3.5 Conclusion

Two methods were developed and evaluated for drawing independent samples from a probabilistic graphical model, where the model is compiled into an arithmetic circuit. These inverse transform sampling techniques rely on decomposing a cumulative probability distribution into a sum of efficiently calculable terms using the arithmetic circuit. For small models, the proposed methods showed the best performance (both for sampler startup time and time per sample). For larger models, dependent samplers were faster in terms of time per sample. However, dependent samplers, suffer from intersample correlation and are unsuitable for synthetic data generation. Considering only independent sampling, the proposed CK-WMC method dominated.

CHAPTER 4

NOVEL DECOMPOSITION SCHEME FOR LAPLACE NOISE PERTURBATION TO ENHANCE DIFFERENTIAL PRIVACY

4.1 Overview

A synthetic data generation process involves designing a generative process that produces data in a way that captures the important statistics of the reference database. In my proposed approach, this generative process is a probabilistic graphical model (particularly a Bayesian Network). Privacy preservation is an important consideration while creating this probabilistic model. The differential privacy (DP) framework is a prominent privacy preservation method used in the creation of synthetic datasets. Bayesian Networks (BN) work well with DP applications, providing an accessible framework for querying statistical properties and setting parameters based on conditional probabilities. BNs make it easier to calculate probabilities straight from the data, avoiding the need to deal with events that never happen in the real data.

However, implementing differential privacy poses significant computational challenges. In order to understand these challenges consider a concept called cross-table. Cross-tables for a collection of random variables are tabular formats used to record the count of the number of occurrences of different combinations of values of the random variables. A cross-table holds all the necessary information for any possible statistical calculation. Adding random noise to cross-tables to mask individual data contributions can present significant computational challenges. The primary difficulty arises from zero-frequency elements

(i.e. rows for combinations of values that are never seen). Cross-tables never include zero-frequency elements, making them compact. But zero-frequency elements need to be considered during noise addition. This is because the number of zero-frequency elements can be much larger than the cross-table itself. While adding random noise to observed data is straightforward, data that never appears (i.e. zero-frequency elements) must be managed carefully to maintain privacy guarantees. This is not merely a theoretical concern but a practical challenge, as the cost of noise addition scales with the number of possible data combinations. In large datasets, there can be so many combinations that the time and resources needed to handle them become impractical. To reduce this computational cost, this chapter proposes a novel decomposition scheme for additive Laplacian noise.

Consider two well known approaches to generate synthetic data [152]. One is taking records from a reference database and perturbing them, e.g., changing values and randomizing record elements. The other approach (which is used here) is to construct a generating process that emits data, where the generating process is designed to somehow match the reference database. For both approaches, it is difficult to construct a perturbation process that guarantees both privacy and statistical similarity to the reference data. Choosing the right kinds of perturbation to protect privacy yet maintain similarity is a challenging trade-off that requires careful implementation.

In addition, no generating process can provide information beyond what is inherent in the model. This means that privacy assurances can focus on the generating process rather than the resulting data. That focus is particularly advantageous when the generating process is based on generative models as privacy assurance may focus on the information embedded in the models. Furthermore, a generation process is potentially unlimited in the data it may produce, without compromising previously assured privacy guarantees. This is ensured by the post-processing property of differential privacy. The post-processing property states that once a quantity is privatized, the privatization cannot be reversed without using the original data again.

Considering these things, the proposed synthetic data generation approach takes existing databases as references to create generative models that can emit potentially unlimited amounts of data. Bayesian Networks (BN) [72] were used as the generative model because a BN can be directly examined to determine the statistical properties of the data they generate. This proposed generation method takes the BN structure as input. The BN structure may be defined manually. Values for the parameters are set algorithmically. Further details of the synthetic data generation method including the mathematical framework behind it will be described in chapter 5 of this thesis.

There are several benefits to using a BN. First, BNs are computationally efficient and scale well with high-dimensional data. Second, their directed acyclic graph structure can be leveraged to explore causal relationships among variables. However, there is also a drawback: although factorizing the full joint distribution is general enough to capture any dependency structure [56], in practice, simplifying assumptions about the graphical structure are often made to ease model inference. These assumptions may overlook higher-order dependencies.

The proposed BN-based method maybe more computationally complex when compared to directly perturbing the datasets. However, this can be managed by compiling the network into an efficient arithmetic circuit. After the model is constructed, the CK-WMC sampler mentioned in the previous chapter was used for sampling the model.

4.2 Related Work

Privacy leakage occurs when available information can identify an individual by linking the data to that person. To effectively de-identify personal information, two key steps are required: (1) remove or modify all direct personal identifiers (as done for many publicly available datasets), and (2) remove or alter any other information that, on its own or combined with other data, could reasonably identify someone. There are a variety of techniques a bad actor can use to infer sensitive information.

Inference attacks are techniques that exploit indirect hints or correlations in the data to extract sensitive information even when all explicit identifiers have been removed by proper anonymisation [85]. For instance, a bad actor may use an inference attack to deduce sensitive medical information about individuals from an anonymised healthcare dataset. There are two types of inference attacks: attribute inference and membership inference. In attribute inference, the known attributes of a person are used to predict their missing attributes. Membership inference involves figuring out whether a specific individual's real data was used to generate the synthetic dataset.

A reconstruction attack is where the synthetic data and the model generating it are used to rebuild the original dataset. For example, a rare disease could be linked to a specific age group and a specific occupation. If this pattern is too specific, the attacker could guess back who had that disease and reconstruct sensitive personal details. Similarly, a linkage attack combines synthetic data with external datasets to re-identify individuals. In a correlation attack, hidden correlations in the synthetic data that mirror real-world correlations are exploited to expose individuals.

There are certain properties that data can have that makes it vulnerable to these attacks. Homogeneity is a data property that arises when data records are too uniform in their sensitive attributes [88] [85]. This property can make it easier to infer sensitive information. Large number of fields (i.e. columns) combined with small number of records (i.e. rows) in a dataset makes each combination of attributes more unique and therefore makes it easier to infer sensitive information. Similarly, highly imbalanced distributions, where some attributes are extremely rare, can be a privacy risk. Uneven distribution of sensitive attributes (e.g., large number of records share a single attribute) is yet another data property that can reduce privacy. These data properties and privacy risks demonstrate the need for a framework that provides strict privacy and accuracy guarantees.

Differential privacy is a mathematical framework that adds controlled noise to the data in order to reduce the probability of leaking private or sensitive information. It is a well-defined framework with strong mathematical guarantees. It limits the effect that the addition or removal of a single database item has on the overall outcome of any analysis done on the dataset [48] [46]. This privacy protection process faces the same dilemma as most other privacy protection methods which is the privacy utility tradeoff. The basic process is that a sensitive dataset is taken and a sanitised version of that data with privacy guarantees is generated. The sanitised version needs to retain the utility of the data for the given application while providing the aforementioned privacy guarantee[55] [78]. In differential privacy, this privacy utility tradeoff is due to the noise being added to the probability distribution of the original data.

Depending on the guarantees provided, differential privacy can be strict or approximate. Strict (or Pure) Differential Privacy requires that for any two neighboring datasets differing by a single entry, and for any possible output event of a algorithm T, the probability that T produces this event differs by at most a multiplicative factor of e^{ϵ} whether T is applied to one dataset or its neighbour [49]. Here ϵ is the "privacy loss" parameter: a non-negative real number that controls how much the output distributions on two neighbouring datasets can differ. The formal definition and details of ϵ-DP is provided in section 2.3.1. It should be noted that when ϵ is small it provides meaningless accuracy and when ϵ is large it delivers weak privacy. The concept of approximate (ϵ, δ) - differential privacy was introduced to address this issue. Approximate (ϵ, δ) - differential privacy allows a tiny slack δ to the strict e^{ϵ} bound, meaning that with probability up to δ the algorithm's behavior may exceed the e^{ϵ} bound [49]. The formal definition and details of (ϵ, δ) - DP can found in Section 2.3.3.

The Laplace algorithm for adding noise is used to release numeric query results under ϵ-DP. In order to understand the Laplace algorithm, it is necessary to understand an important parameter called l_1 sensitivity. The l_1-sensitivity of a function $f : \mathbb{N}^{|X|} \rightarrow \mathbb{R}^k$ can be defined as: $\Delta f = \max_{X,X'} ||f(X) - f(X')||_1$ Here X and X' are neighbouring datasets. The Laplace algorithm for differential privacy works by calculating f and perturbing every coordinate with noise drawn from the Laplace distribution. The noise should be proportional to the l_1 sensitivity. The exponential algorithm for noise addition handles settings where the output isn't a real number (e.g. selecting a feature subset, a label, or any categorical choice). It works by randomizing over a discrete or structured output space based on a utility score. Details for the exponential algorithm can be found in section 2.3.6.

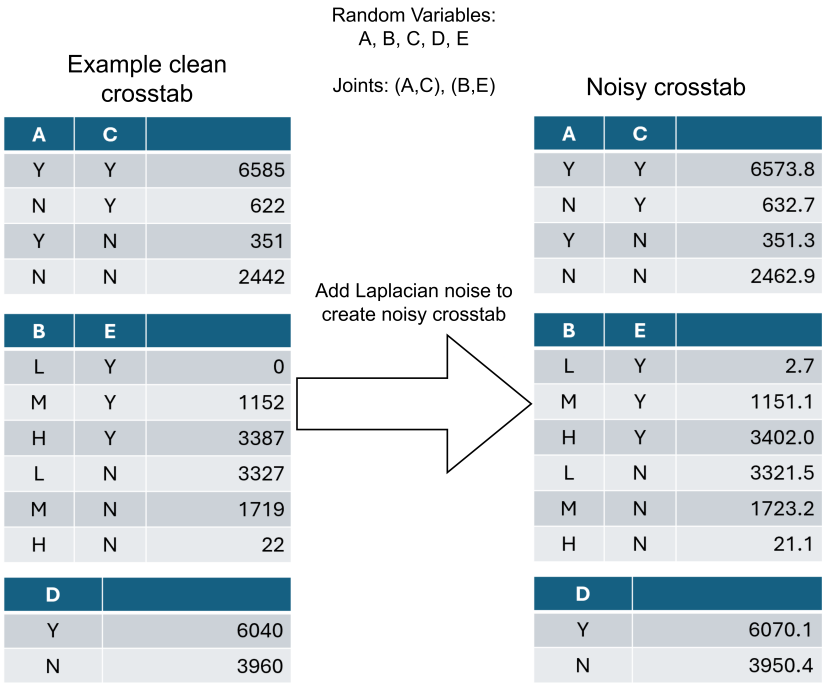
4.3 Proposal

Privacy preservation measures must be implemented during any synthetic data generation process to guarantee the necessary control over privacy leakage. Often, such measures are required to comply with some existing regulation

Consider a concept called cross-table. Cross-tables for a collection of random variables are tabular formats used to record the count of the number of occurrences of different combinations of values of the random variables. A cross-table holds all the necessary information for any possible statistical calculation. For example, given random variables A, B and C each with possible values 'yes' and 'no', there are eight different combinations of values. The definition of a crosstab can then be generalized to replace counts with weights that can take non-negative, non-integer values. Table 4.1 shows an example cross-tab.

Table 4.1: Example of cross-tab.

A	**B**	**C**	**weight**
yes	yes	yes	10
yes	yes	no	6
yes	no	yes	1
yes	no	no	14
no	yes	yes	2
no	yes	no	5
no	no	yes	1
no	no	no	3



A	C	
Y	Y	6585
N	Y	622
Y	N	351
N	N	2442

B	E	
L	Y	0
M	Y	1152
H	Y	3387
L	N	3327
M	N	1719
H	N	22

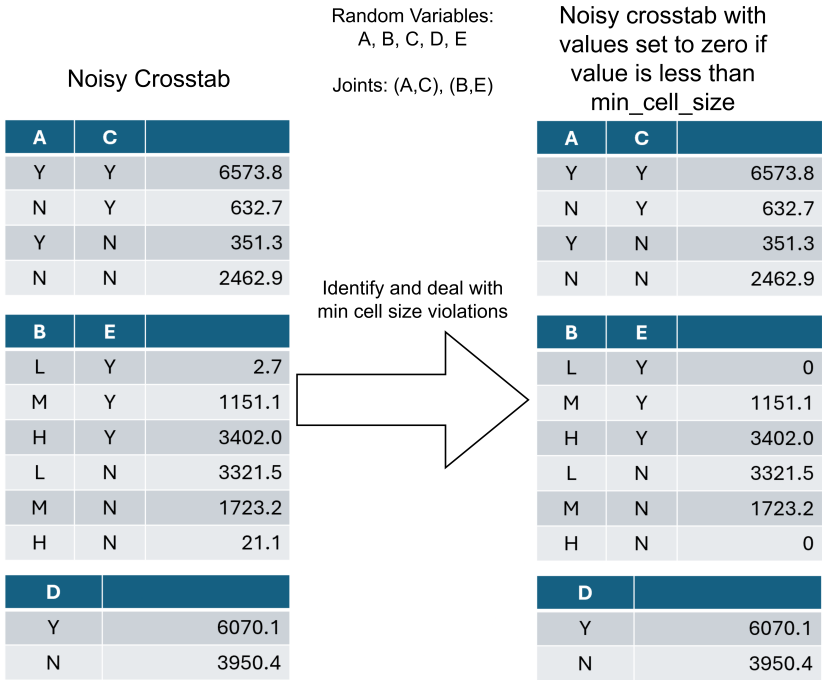
D	
Y	6040
N	3960

A	C	
Y	Y	6573.8
N	Y	632.7
Y	N	351.3
N	N	2462.9

B	E	
L	Y	2.7
M	Y	1151.1
H	Y	3402.0
L	N	3321.5
M	N	1723.2
H	N	21.1

D	
Y	6070.1
N	3950.4

Step 1: Laplace Noise Addition



A	C	
Y	Y	6573.8
N	Y	632.7
Y	N	351.3
N	N	2462.9

B	E	
L	Y	2.7
M	Y	1151.1
H	Y	3402.0
L	N	3321.5
M	N	1723.2
H	N	21.1

D	
Y	6070.1
N	3950.4

A	C	
Y	Y	6573.8
N	Y	632.7
Y	N	351.3
N	N	2462.9

B	E	
L	Y	0
M	Y	1151.1
H	Y	3402.0
L	N	3321.5
M	N	1723.2
H	N	0

D	
Y	6070.1
N	3950.4

Step 2: Suppressing cells below a minimum cell size of 50

Figure 4.1: A demonstration of the noise injection processes for privacy preservation.

Before using a crosstab to define a conditional probability table (CPT), two additional privacy-preserving steps are implemented: (1) adding Laplacian noise for differential privacy, and (2) suppressing cells below a minimum size. These are shown diagrammatically in Figure 4.1 using a simple fictitious example.

4.3.1 Laplace Noise Addition

In the first step, random Laplacian noise values are added to every cross-table count. The amount of noise is controlled by a parameter called noise scale. A noise scale value of x means that random noise with zero mean and σ variance is added or subtracted from each count. However, the actual value added or subtracted is a random floating-point number. If this step causes a count to go negative, the count is reset to zero.

In a differential privacy process, cross-tab weights are perturbed by adding noise to those weights and suppressing the weights below a threshold. Here the noise is drawn from a Laplacian distribution, $L(\mu, b)$, with a Probability Density Function (PDF):

$$p(x|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right), \tag{4.1}$$

and the corresponding Cumulative Distribution Function (CDF) is:

$$P(X < x|\mu, b) = c_{\mu,b}(x) = \frac{1}{2} + \frac{\text{sgn}(x-\mu)}{2} \exp\left(\frac{x-\mu}{b}\right). \tag{4.2}$$

In particular, the parameters are set to $\mu = 0$ and $b = \Delta f/\epsilon$, where ϵ and Δf are differential privacy parameters epsilon and sensitivity, respectively.

4.3.2 Suppression.

In the second step, the minimum cell size is enforced. When any count is less than the minimum cell size, it is reset to zero. Minimum cell size suppression can be performed as part of arbitrary post-processing, the likes of which are inherently protected, but that has no bearing on the Differential Privacy property.

The suppression of weights is controlled by a parameter known as min-cell-size which is represented by r. Cross-tab rows with weight $< r$ are not made available to subsequent processes. In this case, subsequent processes may assume a weight for unavailable combinations of values of the random variables, for example, 0, 1, or $r/2$, etc. Consideration is restricted to situations where r is positive and rounding of values is not used. (Any rounding system may be translated to a non-rounding system by subtracting 0.5 from r.)

Differential privacy has an important property called post-processing. It states that a given algorithm that is (ε, δ) - differentially private will remain so irrespective of any post-processing done on it. Formally this can be stated as: Let $M : X^n \rightarrow Y$ be (ε, δ) - differentially private, and let $F : Y \rightarrow Z$ be an arbitrary randomized mapping. Then $F \circ M$ is (ε, δ) - differentially private. This property ensures that actions taken during the suppression stage will not compromise privacy. It also guarantees that privacy cannot be compromised by drawing potentially infinite number of samples for synthetic data generation.

4.3.3 Reducing Computational Cost via Decomposition

In some practical situations, when a cross-tab is created, rows with zero values are not reported. For example, when using SQL to query a database, "select A, B, C, count(*) from τ group by A, B, C", zero weights are not returned. This provides a significant computational advantage when the space of possible values is large and many of the weights are zero. It provides efficiency for the creation and storage of cross-tabs. It may also lead to efficient models that use such cross-tabs.

However, achieving strict ϵ-Differential Privacy, requires constructing the full state space of possible values (i.e. including zeros). This poses a dilemma when adding noise to a cross-tab. Noise must be added to both the zero and non-zero weights of a cross-tab which forces the noise process to consider all possible combinations of values, even if they are not explicitly represented in the cross-tab. This can lead to unacceptable computational costs.

It should be noted that when a min-cell-size is enforced subsequent to adding noise, then the combination of adding noise, then suppressing low weights may lead to no overall effect on a cross-table row. This inspires an approach that decomposes the perturbation process in a more efficient manner.

As shown in Table 4.2, the perturbation of a weight conforms to one of four cases:

Table 4.2: Four different cases of the combination of initial and subsequent weight.

Case Name	**Initial Weight**	**Subsequent Weight After Suppression**
ZS	Z: weight = zero (not represented in the cross-tab)	S: addition of noise leaves weight < r (suppressed)
ZN	Z: weight = zero (not represented in the cross-tab)	N: addition of noise leaves weight ≥ r (not suppressed)
WS	W: weight > zero (represented in the cross-tab)	S: addition of noise leaves weight < r (suppressed)
WN	W: weight > zero (represented in the cross-tab)	N: addition of noise leaves weight ≥ r (not suppressed)

The proposal here is to restructure the perturbation process so that case ZS involves no processing to provide a computational advantage. Observe that the W cases (WS and WN) do not need special treatment, only the Z cases. The Z cases can then be treated collectively as a repeated application of noise, followed by a suppression decision.

For an individual row, let the probability of case ZN given Z be α, thus:

$$\alpha = P(ZN|Z) \tag{4.3}$$

$$= \int_r^\infty \frac{1}{2b} \exp\left(-\frac{x}{b}\right) dx \tag{4.4}$$

$$= \frac{1}{2} \exp\left(-\frac{r}{b}\right). \tag{4.5}$$

Consider a cross-tab of m rows with zero weight. The number of ZN cases is a binomial random variable with distribution $B(m, \alpha)$.

The noise distribution in the case ZN is a 'truncated' Laplacian. Given that $\mu = 0$ and $r > 0$, the PDF is defined as

$$p(x|r, b, ZN) = \begin{cases} \frac{1}{2b} \exp\left(-\frac{x}{b}\right) & \text{if } x > r \\ 0 & \text{if } x \leq r \end{cases}, \tag{4.6}$$

where z is the required normalisation constant. The corresponding CDF is:

$$P(X < x|r, b, ZN) = \frac{1}{z} \begin{cases} \int_r^\infty \exp\left(-\frac{x}{b}\right) & \text{if } x > r \\ 0 & \text{if } x \leq r \end{cases} \tag{4.7}$$

$$= \frac{1}{2z} \begin{cases} \exp\left(-\frac{x}{b}\right) & \text{if } x > r \\ 0 & \text{if } x \leq r \end{cases}, \tag{4.8}$$

with

$$\frac{1}{z} = \int_r^\infty \exp\left(-\frac{x}{b}\right) = \frac{1}{2} \exp\left(-\frac{x}{b}\right). \tag{4.9}$$

Therefore,

$$P(X < x|r, b, ZN) = \begin{cases} \exp\left(-\frac{r-x}{b}\right) & \text{if } x > r \\ 0 & \text{if } x \leq r \end{cases}. \tag{4.10}$$

The inverse CDF is $r - b \ln x$. CK-WMC sampler may used to draw samples from this distribution.

Putting this all together, Algorithm 2 is a process to add noise and suppression for a cross-tab, with m rows of zero weight.

Algorithm 2: Privacy-preserving Cross-tab Generation for Case ZN.

Data: Parameters r and b, and no. of zero weight rows m.
Result: Privacy preserving Cross-tab variables w.

1 Set $\alpha = \frac{1}{2}\exp\left(-\frac{r}{b}\right)$;
2 Draw a random variate $n \sim B(m, \alpha)$;
3 Randomly choose n different combinations of values for the cross-tab random variables, where each chosen combination was not a combination with row weight > 0;
4 **for** *each chosen combination* c **do**
5 Draw a random variate $x \sim U(0,1)$;
6 Let $w = r - b\ln(x)$;
7 Include a row in the cross-tab with combination c and weight w;
8 **for** *each row with original weight* $w > 0$ **do**
9 Draw a random variate $x \sim Lap(0, b)$;
10 **if** $w + x < r$ **then**
11 Suppress the row: $w = 0$;
12 **else**
13 Set the row weight to $w = w + x$;

Line 3 of the algorithm is the challenging step in this process. If $n \ll m$ then this step may be achieved using rejection sampling, i.e., by uniformly sampling possible combinations and keeping a sample only if it is not already in the cross-tab. To be computationally efficient, m should be large, and α should be small, i.e., a large value for r relative to b.

The paper "Differentially Private Summaries for Sparse Data" by Cormode et al [26] provides the primary theoretical support for this decomposition scheme. The authors [26] identify the same computational bottleneck addressed here: adding noise to every zero-frequency combination transforms a compact sparse dataset into a massive dense output. To resolve this, they propose generating differentially private summaries directly from the original data without materializing the full noisy contingency table. Using an explicit sampler and a binomial sampling argument, they show that selectively processing only the zero-entries that pass a suppression threshold yields an output distribution identical to the naive approach of materializing all noisy entries then suppressing low counts [26]. This equivalence holds provided the sampler matches the implicit distribution induced by the full-noisy model [26]. The sampling rules in Algorithm 2 must follow the same construction. Under those conditions, Cormode et al [26] formally guarantee the differential privacy of the zero-weight processing.

4.4 Evaluation

4.4.1 Experimental Settings

The new decomposition scheme was evaluated using the Texas dataset [145]. The Texas Hospital Discharge dataset is a large public-use data file provided by the Texas Department of State Health Services. The dataset used here consists of 50,000 records uniformly sampled from a pre-processed data file that contains patient records from the year 2013. Eighteen data attributes are retained, of which 11 are categorical and 7 are continuous.

Performance evaluation can be conducted in three aspects: utility, privacy and computational cost (scalability).

1. **Utility:** Three key approaches to evaluating the utility of synthetic data are used here, as suggested in [50].

    - **Structural Similarity**: confirm variable names, types, formats and 'edit check' rules are satisfied.
    - **Bias and Stability**: measure the variation across each time synthetic data is generated or the generating model is calibrated.
    - **Statistical Similarity**: measure the statistical differences between real and synthetic datasets. Histogram Intersection (HI) and Kullback–Leibler Divergence (KL) are two different methods used to measure the similarity between two probability distributions or histograms. A higher HI value indicates a greater similarity. On the other hand, KL with a value of 0 indicates that the two distributions are identical.

2. **Privacy:** Apart from the theoretical DP guarantee, another privacy metric from Stadler et al [139] is also used.

    - **Privacy gain (PG) [139]**: the privacy gain of publishing a synthetic dataset S in place of the raw data R for target record r_t as the reduction in the adversary's advantage when given access to S instead of R, i.e., $PG \triangleq Adv(R, r_t) - AdV(S, r_t)$. The privacy gain can assess whether synthetic data is, as promised, an effective anonymisation mechanism.

3. **Computational cost:** Compare the computation costs (CPU time or the number of calculations) of adding noise to all cross-tab values and to sparse cross-tabs by decomposing the noise distribution.

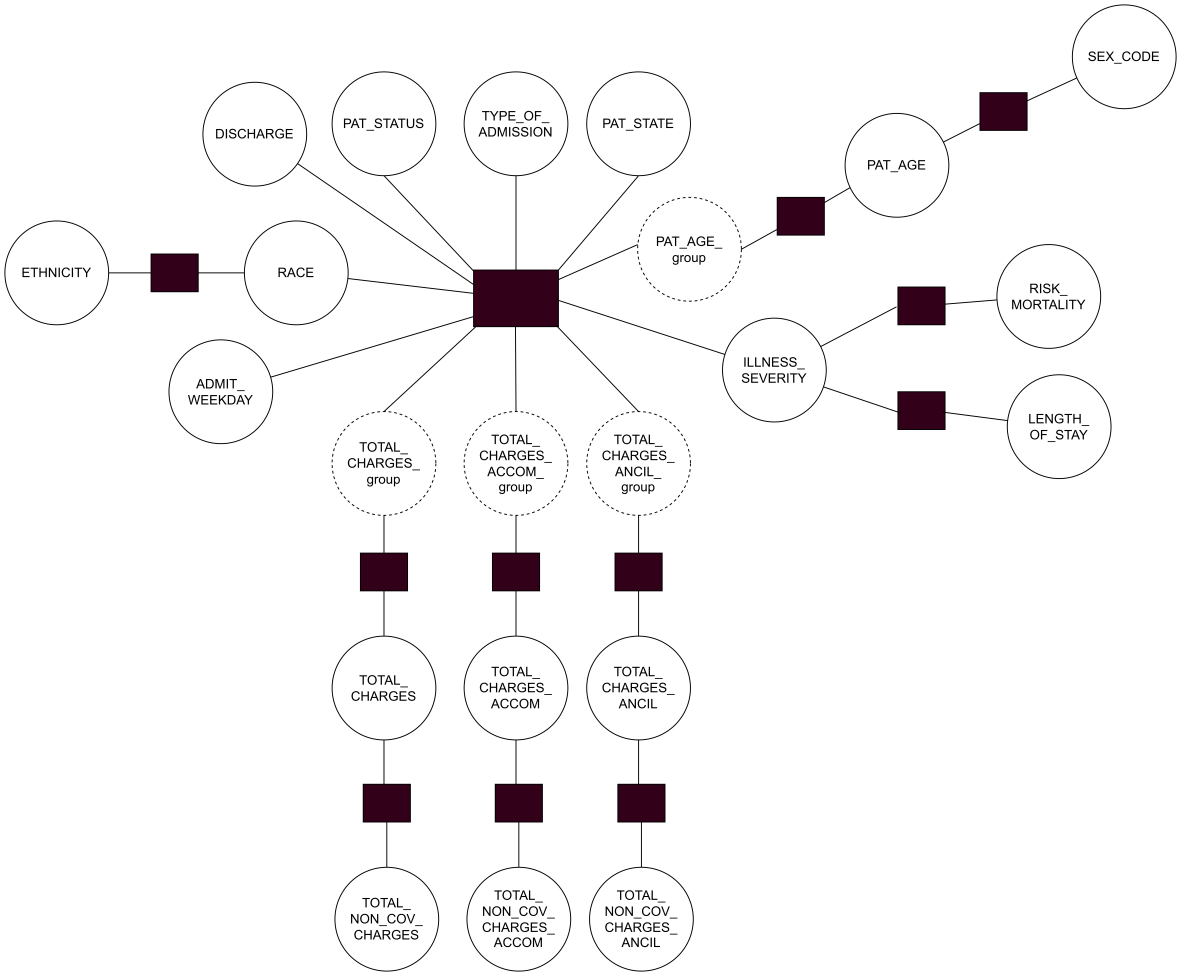


Figure 4.2: The probabilistic graphical model used for the Texas dataset. Random variables are shown as circles. Crosstabs are shown as squares.

4.5 Results

The probabilistic model representing the Texas dataset is illustrated in Figure 4.2. Each dashed random variable was functionally derived through a process of histogram normalization, involving the categorization of corresponding original random variable values into five distinct buckets. A noteworthy aspect of this model is that the crosstabs generated between these paired variables are not directly extracted from the dataset; hence, they are not inherently subject to noise. Consequently, I introduced noise to a total of 12 such crosstabs. This noise induction employed two distinct approaches: the proposed method (decomposed Laplacian noise) and the standard method (conventional full state space representation).

4.5.1 Utility

Structural Similarity: Correct data types, variable names and formats are guaranteed, as the synthetic values are derived directly from the reference dataset. However, edit check rules cannot be enforced since these rules are unavailable for the Texas dataset. Nevertheless, invalid combinations of variable values can be effectively controlled by ensuring their corresponding frequency is set to zero within cross-tables.

Bias and Stability: Once constructed, the model fully specifies a stationary joint distribution. Consequently, multiple synthetic datasets generated from the same model are unbiased and perfectly stable.

Statistical Similarity: The histogram-intersection (HI) values were determined for all cross-tables in Figure 4.2 for both the standard method and the proposed method. Figure 4.3 shows the highest, lowest and median HI values out of all the HI values from the 12 crosstabs. It is apparent that there is no statistically significant difference between standard and proposed methods.

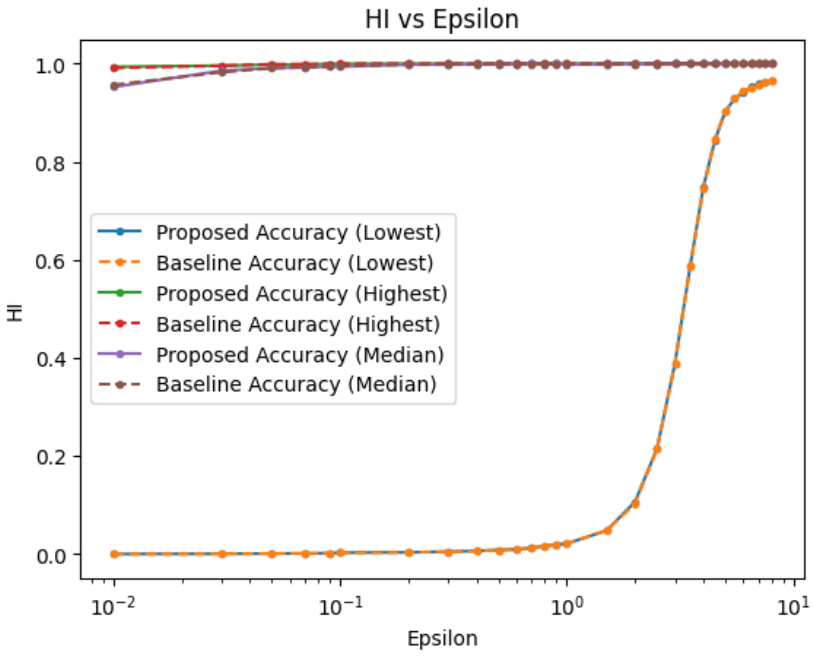
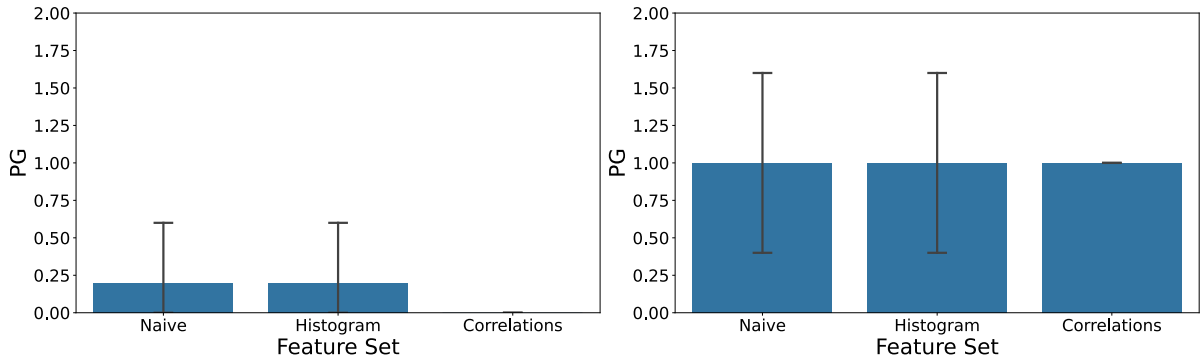
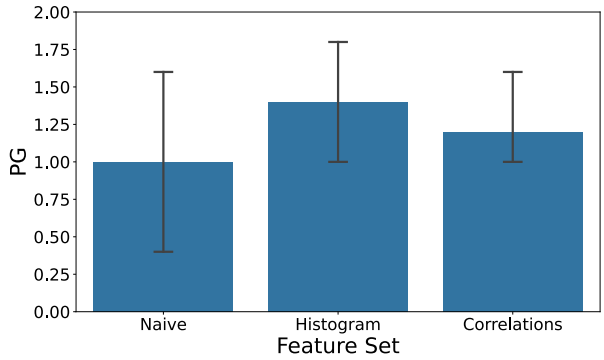


Figure 4.3: Graph showing the accuracy of the proposed and standard method. The highest, lowest and median histogram-intersection (HI) values are shown.

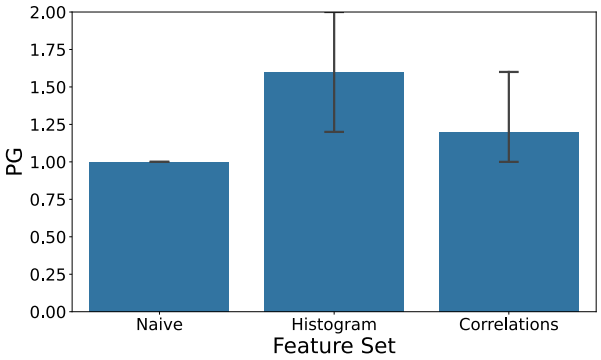


(a) SanitiserNHSk10

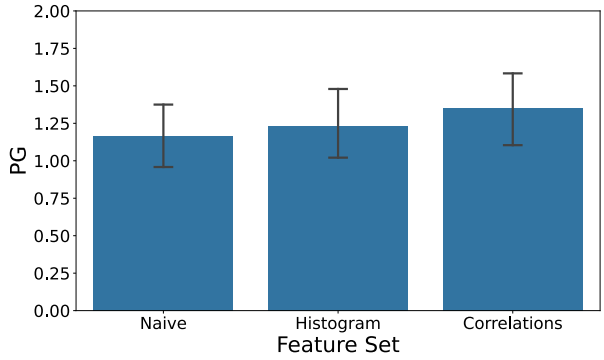
(b) BayesianNet



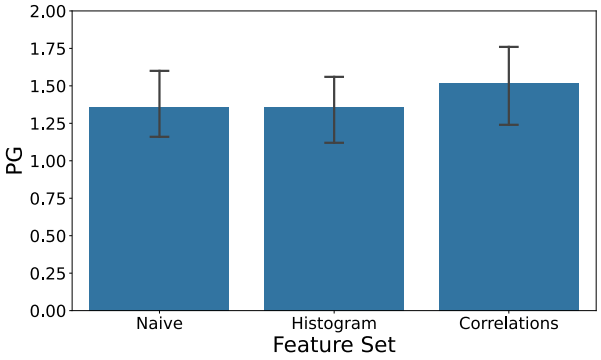
(c) PrivBayes $\epsilon = 1$



(d) PrivBayes $\epsilon = 0.1$



(e) Proposed $\epsilon = 1$



(f) Proposed $\epsilon = 0.1$

Figure 4.4: The comparison of the privacy gain (PG) results of linkability game between several generation models and the proposed method.
These privacy gain plots were created with help from a collaborator for my paper [142].

4.5.2 Privacy

In addition to the privacy guarantees provided by DP, the privacy performance is also evaluated using the PG metric via conducting linkability games following [139].

A linkability game involves a privacy challenge between an adversary and a challenger, where the challenger releases a synthetic dataset accessible to the adversary. The adver-

sary's objective is to infer whether a target record originates from the original dataset, leveraging the synthetic dataset and prior knowledge. In addition, the adversary can use different feature extraction techniques to distinguish the feature vectors extracted from synthetic datasets with and without a target record. The results for the generation model using three different feature sets are provided:

- Naive: this feature set is designed to capture the basic statistical properties of the dataset. The median, mean and variance is calculated for every numerical attribute for this feature set. As for categorical attributes, the number of unique categories as well as the least and most common categories are recorded. This approach provides a straightforward summary of the data's central tendency, variability, and category frequencies.
- Histograms: this set focuses on understanding the distribution of data attributes. For numerical attributes, it segments data into bins of a configurable size, creating a histogram that reflects the data's distribution. Categorical attributes are treated by counting the frequency of each category. The number of bins for numerical attributes can be adjusted for each dataset, allowing for tailored analysis of data distribution.
- Correlations: this feature set aims to uncover relationships between pairs of attributes within the dataset. It calculates the correlations between numerical attributes directly. For categorical attributes, it first converts them into a numerical format through dummy encoding, which transforms categorical data into a series of binary variables. This enables the computation of a pairwise correlation matrix, offering insights into how attributes relate to each other.

Baseline methods. Three other synthetic data generators are used as the baseline methods, namely SanitiserNHSk10, BayesianNet, and PrivBayes.

- SanitiserNHSk10 [100]. This sanitization procedure, described by NHS England, employs a deterministic function to apply a series of predefined row-level transformations to input data (R), producing a sanitized dataset (S) that meets a heuristic privacy definition. Typical transformations include generalization, perturbation, or the deletion of individual rows to enhance data privacy.
- BayesianNet [173]. This method leverages Bayesian Networks (BN) to construct models, yet it lacks integration of privacy preserving mechanisms. It focuses on model accuracy and inference capabilities without addressing the privacy concerns associated with the data it processes.

- PrivBayes [173]. This is the PrivBayes algorithm explained in section 2.4. The primary distinction from our proposed method lies in its approach to differential privacy: during the model training phase, differential privacy noise is introduced to the probabilities of conditional distributions. However, this method does not offer formal Differential Privacy (DP) guarantees, marking a significant difference.

Figure 4.4 provides the privacy gain (PG) results across different generation models in a linkability game framework. The figure presents various approaches, including SanitiserNHSk10, BayesianNet, and PrivBayes with ϵ = 1 and 0.1, and the proposed method with ϵ = 1 and 0.1. The proposed method shows consistently robust privacy gains across all feature sets. SanitiserNHSk10 exhibits significantly lower privacy gains across all feature sets, while BayesianNet and PrivBayes show higher privacy gains similar to the proposed method. The privacy parameter ϵ appears to influence the stability of results, with $\epsilon = 0.1$ generally showing better performance than the results for $\epsilon = 1$.

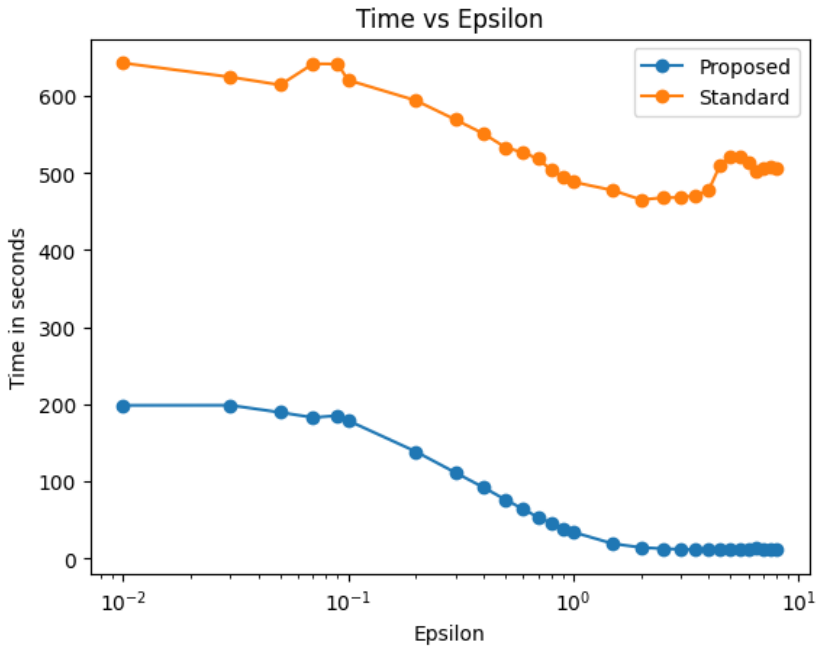


Figure 4.5: Total time taken for all crosstabs to generate noise for a range of epsilon values.

4.5.3 Computational Cost

The time taken to generate noise for all 12 crosstabs were calculated for both the proposed and standard method. The wall clock time was measured for epsilon values ranging from

0.01 to 8. Figure 4.5 shows the time taken to generate noise for the whole range of epsilon values.

The experiments here focused on the Laplace mechanism because it provides a closed-form, pure epsilon differential privacy guarantee. It is also possible to calibrate for queries with known sensitivity. This choice enables theoretical guarantees and the implementation of the synthesis pipeline, and it is also used extensively in the literature. However, Laplace has known limitations under composition. Basic sequential composition accumulates privacy loss additively, so repeated releases quickly exhaust the privacy budget. However, here a model is released, so subsequent generation of synthetic data does not exhaust the privacy budget.

4.6 Conclusions

The time taken to generate noise for the proposed method and the standard method was calculated over a range of epsilon values. The time taken by the standard method was approximately three times that of the proposed method. This demonstrates the significant computational cost advantage that can be gained by enforcing the min-cell-size and by restructuring the perturbation process for the zero weight cases.

The accuracy of the proposed method was compared against the standard method for generating noise. The histogram-intersection (HI) was calculated for both methods over a range of epsilon values for all crosstabs. There is no statistically significant difference in accuracy between the proposed method and the standard method. The privacy performance via the PG metric was evaluated by conducting linkability games with three baseline data generators. The proposed method not only meets Differential Privacy (DP) standards but also enhances overall privacy by reducing potential leakage, providing strong protection for sensitive information in complex, high-dimensional datasets.

CHAPTER 5

COST-EFFICIENT AND DIFFERENTIALLY PRIVATE SYNTHETIC DATA GENERATION WITH REFERENTIAL INTEGRITY USING PROBABILISTIC GRAPHICAL MODELS

5.1 Overview

In this thesis, synthetic data generation is done by constructing a generating process that produces data in a way that captures the important statistics of the reference database. The generating process being used here relies on a probabilistic graphical model (PGM). Such a generating process based on a PGM can produce unlimited synthetic data by sampling the model. Privacy preservation for the model is done using the modified differential privacy framework proposed in Chapter 4. The sampler used to emit data is the CK-WMC sampler proposed in Chapter 3.

The proposed synthetic data generator needs to work well with multi-table relational datasets. A relational dataset stores information about related objects in a tabular structure. In order to understand relational datasets, I will introduce some terminology here. A table in a relational dataset can be represented as an *entity*. Each entity defines a set of *fields* that correspond to the columns of that table. A *record* refers to an actual instance of an entity stored in a table (i.e. a single row of the table). The proposed generator would need to represent a relational dataset using PGMs.

Referential integrity is an important concept that defines relationships between tables in such a relational dataset. It can be understood by examining the relationship between

primary keys and foreign keys in a relational dataset. A primary key is an unique identifier field in a table that individually identifies a record of that table. When a field in one table references a primary key in another table, it is called a foreign key. Referential integrity is a property about relationships between tables which asserts that every foreign key value refers to an existing primary key. Referential integrity maybe compromised due to any of the following reasons.

1. **Joining tables**: If inner join is used, only rows with matching keys are kept. This may drop rows that should be part of the relationship. If outer join is used, missing values (NaN) will be introduced where no match exists. This breaks the expected foreign key linkage.

2. **One-to-Many Relationships**: Merging can duplicate data from the "one" side of a one-to-many relationship, potentially complicating how you interpret the integrity of the original relationships.

3. **Loss of Explicit Constraints**: When data is in separate tables, relational databases enforce primary and foreign key constraints. Once merged, that structure is flattened, and you lose the automatic checks that maintain referential integrity.

The proposed generator should be able to represent a relational dataset using PGMs. There are difficulties in representing multi-table relational datasets using PGMs. The PGM model may need to represent complex entity relationships. Multiple independent tables in the relational database may refer to the same entity and the model would need to take that into account. The PGM model must ensure that referential integrity is maintained in the generated synthetic dataset.

The main contribution for this research is a novel cost-efficient synthetic data generator that can natively handle multi-table relational datasets while maintaining referential integrity and providing privacy guarantees via differential privacy. The proposed generator constructs a complex probabilistic model to ensure referential integrity. It does this by creating a hierarchy of simpler probabilistic models with each model representing entities with relationships of definable fixed or random cardinalities.

In this hierarchical tree structure, every arc represents a specific record generation step. At each step, multiple records are generated within a given context set for the entity until one or more stopping conditions are met. The context set for an entity e contains all the parent information that will condition the generation of e's records for one record generation step.

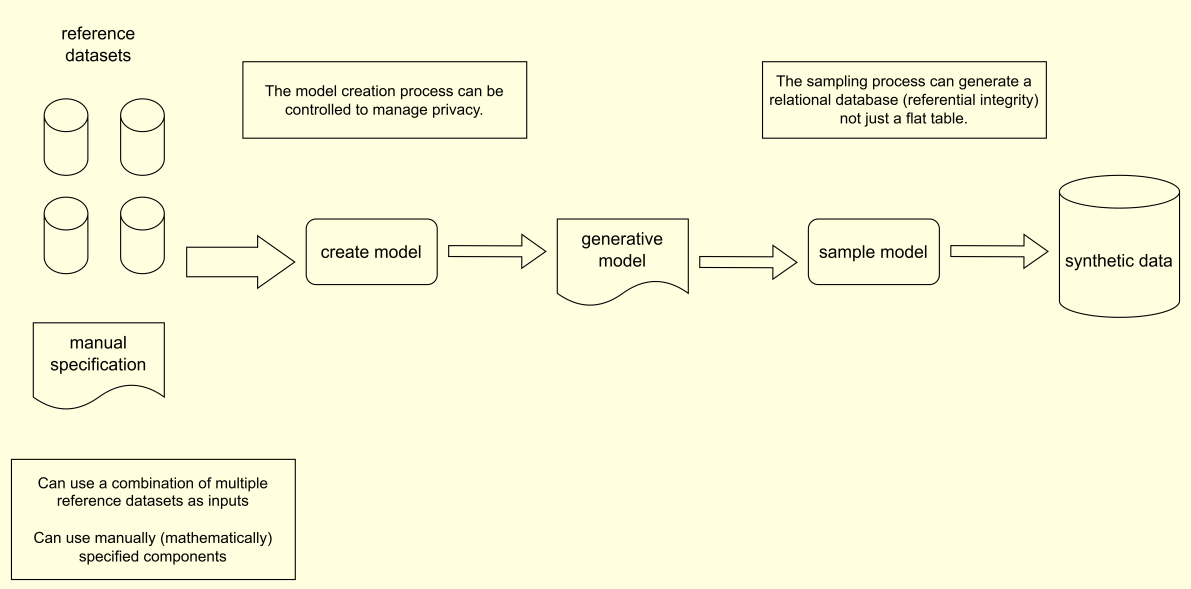


Figure 5.1: Diagram of the synthetic data generation process

At each step in the tree, the PGM is used to sample a child record, conditioned on the data from its context set as needed by the model. The context set can include the parent entity and the previous record. This approach captures the conditional dependencies between records, ensuring that the synthetic data respects important statistics of the probabilistic model. Parent-child entity relationships represents 1-to-many relationship between their records. Referential integrity is reinforced by assigning a unique identifier to every record; these identifiers act as primary keys in their respective tables and serve as foreign keys in related child tables, thereby maintaining consistent and accurate links between records. Every entity also has a count field which is a counter, for the record number counting the children of a single parent record.

This proposed generator requires creating a generative model (i.e. PGM) using multiple sources of real data and hand crafted components. Bayesian Networks (BN) were used as the generative model because they can be directly examined to determine the statistical properties of the data they generate. They are also computationally efficient and scale well with high-dimensional data. The structure of the BN is taken as an input and the parameter values are set algorithmically. Figure 5.1 represents the entire synthetic data generation process in the proposed generator. Figure 5.2 is example of entity relationships consisting of a single tree.

The record generation process must have a mechanism to determine how many child records to generate per parent record. For the proposed generator this is determined by

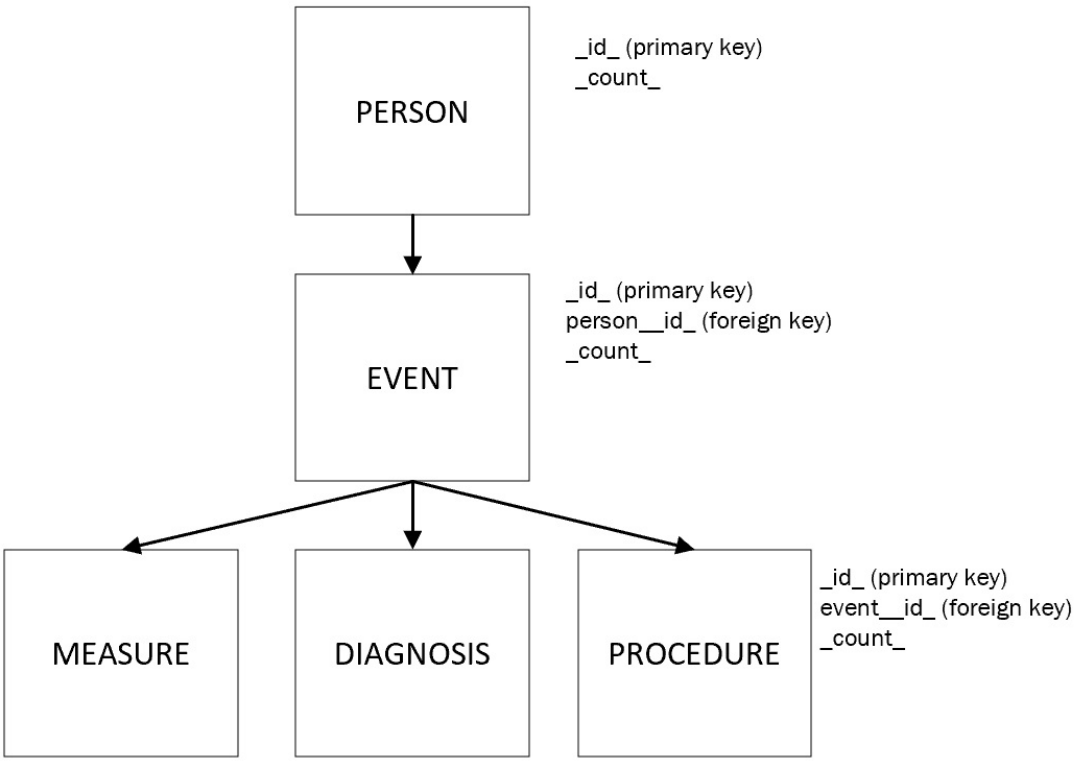


Figure 5.2: Example entity relationships consisting of a single tree

testing one or more stopping conditions. The following five stopping conditions cover all possible scenarios that determine when to stop the generation process.

1. **Fixed Limit**: Stop generating if the record count value reaches a fixed value. (i.e., fixed number of children per parent).

   Consider the example entity relationships diagram for a patient journey model shown in Figure 5.2. `PERSON`, `EVENT`, `MEASURE`, `DIAGNOSIS` and `PROCEDURE` are all entities. `PERSON` has a one-to-one relationship with `EVENT` and `EVENT` has one-to-many relationships with `MEASURE`, `DIAGNOSIS` and `PROCEDURE`. Assume that there are 3 `MEASURE`s per `EVENT`. This means that for every record in the `EVENT` entity, 3 records must be generated by the `MEASURE` entity. This is an example of fixed limit stopping condition.

2. **Variable Limit**: Stop generating if the record count value reaches the value of another field of the entity or parent.

   Assume that the number of diagnoses for this patient journey model is a random variable with a distribution ranging from 3 to 10. This means that `EVENT` needs to have a random variable *num_of_diagnoses* that represents a conditional probability

distribution with values ranging from 3 to 10. Therefore the cardinality of DIAGNOSIS is the variable limit *num_of_diagnoses*.

3. **Fixed Comparison**: Stop generating if the value of a nominated field reaches a fixed value.

   For every individual person, their events form a sequence. A pair of adjacent events will have a random number of days between them. Assume that all events corresponding to a person can only cover 5000 days in total. Here the entity EVENT needs to have a random variable *num_of_days_since_prev_event* which is a conditional probability distribution. The EVENT entity must also have fields *day_number_of_prev_event* and *day_number*. The value for *day_number_of_prev_event* will be copied from the previous event's *day_number* field. Therefore the field *day_number* is equal to the sum of *day_number_of_prev_event* and *num_of_days_since_prev_event*. The cardinality of EVENT is a fixed comparison to ensure that *day_number* is greater than 5000.

4. **Variable Comparison**: Stop generating if the value of a nominated field reaches the value of another field of the entity or parent.

   Consider the previous example where a pair of adjacent events will have a random number of days between them. If the total number of days is denoted by a variable *limit_days* instead of a fixed number like 5000, the stopping condition type becomes variable comparison. The cardinality of EVENT is a variable comparison to ensure that *day_number* is greater than *limit_days*.

5. **Field State**: Stop generating if a nominated field in the entity matches a value in a set of stopping values.

   For an example of the field state stopping condition, consider the case where event generation needs to stop because the corresponding person died. This means stop generating events after event type is Death. Here EVENT needs to have a random variable *death_type*. The cardinality of EVENT is a field state where *death_type* is Death.

5.2 Related Work

This section reviews current state-of-the-art synthetic data generators. There are synthetic data generation mechanisms that rely on graphical models, copulas, marginals and generative adversarial networks. Consider two state-of-the-art synthetic data generators that are publicly available. The first one under consideration here called DataSynthesizer [112] was

explored in section 2.4.1 of this thesis. The second data generator being considered here is the Synthetic Data Vault (SDV) [107] which was explored in section 2.4.2.

DataSynthesizer is an example of a synthetic data generator that works using graphical models. It relies on a differentially private Bayesian Network (BN) to model the joint distribution [112] [173]. DataSynthesizer does not natively support multi-table relational datasets. Therefore the reference datasets must be merged into a single table before providing it to the generator. The algorithm learns the Bayesian Network using the input dataset. In the first step, all the joint probability distributions $P(v,u)$ must be determined. Here v and u represent two nodes where the edge from u to v represents the probability of v being caused by u (i.e $P(v|u)$). If the node v is dependent on more than one node (i.e. $P[v|U]$ where $U = u_1, \ldots, u_k$), then $P[v|U]$ is needed to sample v.

After all the joint probability distributions $P(v,u)$ are determined, the next step for the DataSynthesizer is to calculate the conditional distributions $P(v|u)$ from the joint distributions [112] [173]. The algorithm selects the minimum required number of these marginals by building the network. The first node of the network is randomly assigned an attribute. The Exponential algorithm (or mechanism) is then used to select the remaining attributes one by one. Typically, the mutual information metric would be used to find the most correlated marginals. However, in the setting of differential privacy, mutual information has very high sensitivity. So the authors [173] introduced a function similar to mutual information. After determining the marginals, the second step is to add Laplacian noise to satisfy differential privacy.

Synthetic Data Vault (SDV) is a synthetic data generator that works using Gaussian Copulas [107]. It can natively support multi-table relational datasets. SDV works by first modeling the statistical properties of each table using Gaussian copulas to capture the marginal distributions and inter-column dependencies. This is done by creating the joint probability distribution of all the random variables. First the marginal distributions of all the random variables are converted into standard normal distributions with the help of their cumulative distribution functions (CDF). The dependence between these random variables are then modelled using a covariance matrix. The Gaussian copula combines these new distributions with the covariance matrix to construct the joint probability distributions of the original random variables [107].

The SDV framework then extends this modeling to multiple related tables using a technique called Conditional Parameter Aggregation (CPA) [107]. CPA works by aggregating statistical information from child tables for each row of a parent table. It computes distribution parameters and covariance matrices of related child-table data, adding these as

extra columns to the parent table. These columns are referred to as derived columns and they explicitly capture the conditional relationships between parent and child tables. The CPA approach works when there is a parent-child relationship. But the children can have more children as well. Hence, the authors [107] proposed a Recursive Conditional Parameter Aggregation (RCPA) approach where the CPA is done recursively through all the parent's descendants.

Both the SDV generator and my proposed generator can natively handle multi-table relational datasets. SDV relies on the RCPA technique to maintain referential integrity in the database. The proposed generator relies on the previously mentioned hierarchical tree structure to maintain referential integrity. Parent-child entity relationships in the proposed generator are 1-to-many relationships. SDV can handle multi-parent relationships, while the proposed generator cannot. SDV does not provide strong privacy guarantees via differential privacy. It can only obfuscate the data by introducing additional noise or perturbation in the model parameters [107]. The main advantage of the proposed generator over SDV is that it provides strong privacy guarantees via differential privacy. The cost-efficient novel decomposition scheme from Chapter 4 is used for achieving differential privacy. The proposed generator also reduces computational cost by relying on the CK-WMC sampler explained in Chapter 3.

DataSynthesizer has its own implementation of differential privacy. Therefore, both DataSynthesizer and the proposed generator can provide strong privacy guarantees. However DataSynthesizer is limited to single table datasets. In order to evaluate DataSynthesizer for this research, I used the inner join from the 'pandas' Python library to merge all the tables of the given multi-table dataset. The ability to natively handle multi-table relational datasets is a key advantage the proposed generator has over DataSynthesizer. The proposed generator is expected to have a significant advantage in terms of computational cost because it will not have to work with one large high-dimensional table instead it can work with multiple related tables. Both the novel sampling method and the novel decomposition scheme for differential privacy are expected to provide a reduction in computational cost as well.

5.3 Proposal

The main contribution here is a cost-efficient and differentially private synthetic data generator that uses a hierarchy of probabilistic models to maintain referential integrity. In this section, I will provide a mathematical description of the proposed generator.

Let E be the set of all entities in the relational schema. For each entity $e \in E$, a set of parent entities U_e can be defined. Since the design of the synthetic data generator is hierarchical, each entity has one or zero immediate parent, but U_e can also include all ancestors up the chain. For a root entity r, $U_r = \{\}$.

When generating a new record for entity e, there is one context record x_u available for each ancestor entity u in U_e. The values of the fields for those context records are collected into a context set, defined as follows.

$$C_e = \{x_u = e_u | e_u \in U_e\}, \tag{5.1}$$

When generating records for entity e, the number of records n_e needs to be determined. As mentioned previously, it is possible to have practical mechanisms (i.e. stopping conditions) for doing that. The details regarding n_e will be closely examined later.

It should be noted that the count of records generated for e potentially depends on C_e. Given the value n_e, each record $x_{e,i}$ for $i = 1, 2, \ldots, n_e$ is drawn from a distribution conditioned on:

1. The parent context C_e
2. The previously generated record $x_{e,i-1}$ from the same entity e

This is the set of records for entity e with context C_e. This is formally written as:

$$x_{e,i} \sim X_e \mid C_e, x_{e,i-1}, \quad \text{for } i = 1, 2, \ldots, n_e \tag{5.2}$$

The base case or initial prior, $x_{e,0}$ is defined as part of the synthetic data generation system, as is X_e for each entity e.

Now n_e needs to be more closely examined. The parameter n_e is not treated as a random variable. Instead there is a set of stopping functions for a particular generation step with a given context set. If any of the stopping functions in the set return *True*, then the generation step stops, and the algorithm proceeds to the next entity. A generation step may have any number of stopping functions associated with it. However, all stopping functions fall under one of five stopping conditions. The stopping conditions determine n_e indirectly.

Recall that for an entity e, the child records are generated sequentially. After generating each record $x_{e,i}$, all the stopping functions are checked to decide whether to stop (no more child records) or continue (generate $x_{e,i+1}$).

The five types of stopping conditions are denoted as Z_{fix_lim}, Z_{var_lim}, Z_{fix_comp}, Z_{var_comp} and Z_{field_state}. All possible stopping functions are covered by these 5 types of stopping conditions. Using Z to denote the overall stopping function yields,

$$Z(i, x_{e,i}, x_{e,i-1}, C_e) \in \{True, False\} \tag{5.3}$$

where,

- i is the index of the record just generated in entity e.
- $x_{e,i}$ is the i-th record's data.
- $x_{e,i-1}$ is the previously generated record for $i > 1$. If $i = 1$, $x_{e,0}$ is the initial prior.
- C_e is the context set for entity e.

If Z evaluates to *True*, the generator stops creating more records in e for that parent context. Otherwise i is incremented and $x_{e,i+1}$ is generated.

Each of the stopping functions can be represented as a predicate that feeds into Z. If any of those conditions are met, $Z(.)$ = True.

1. **Fixed Limit**

   For this stopping condition, the generating process stops if the record count value reaches a fixed value L.

   $$Z_{fix_lim,L}(i) : i \geq L$$

2. **Variable Limit**

   For this condition, the generating process stops if the record count value reaches the value of another field (either in the same entity or the parent). Let $l(C_e)$ be the limiting value provided by a specific field in the context C_e (or it could come from the same entity's attributes if needed). Then

   $$Z_{var_lim,l}(i, C_e) : i \geq l(C_e)$$

3. **Fixed Comparison**

   For this condition, the generating process stops if the value of a nominated field f in the current record reaches a fixed threshold T. Let $x_{e,i}[f]$ denote the value of field f in the record $X_{e,i}$. Then

   $$Z_{fix_comp,f,T}(x_{e,i}) : x_{e,i}[f] \geq T$$

4. **Variable Comparison**

   For this condition, the generating process stops if the value of a nominated field reaches the value of another field of the entity or parent. Let $g(C_e)$ represent the relevant comparison value drawn from a field in the context (or it could also be from another field in $x_{e,i}$ itself). Formally,

$$Z_{var_comp,f,g}(x_{e,i}, C_e) : x_{e,i}[f] \geq g(C_e)$$

5. **Field State**

   For this condition, the generating process stops if a nominated field f in the entity e matches a value in a predetermined set S.

$$Z_{field_state,f,S}(x_{e,i}) : x_{e,i} \in S$$

Since a single stopping event should terminate record generation for the current parent, all stopping functions are combined with logical "OR" notations. Thus, the overall stopping function is:

$$Z(i, x_{e,i}, x_{e,i-1}, C_e) = Z_1 \vee Z_2 \vee Z_3 \vee \ldots$$

The individual stopping functions are denoted as $Z_1, Z_2, Z_3, \ldots$. Each of the stopping functions will fall under one of the five stopping stopping conditions. For instance, consider a stopping function Z_j that falls under the fixed limit stopping condition. This results in $Z_{fix_lim,L_j}(i) : i \geq L_j$.

5.4 Implementation Details

This section provides the details about developing and implementing my proposed generator in Python. A custom implementation of knowledge complication known as *compiled-knowledge* (CK) was written in Python 3 [126]. CK compiles a factor graph to an arithmetic circuit over floating point types using the method proposed by Chavira & Darwiche [20]. Next, the arithmetic circuit is compiled into a program comprising of a static array of memory locations and a sequence of instructions. The memory is used for output values, input values, temporary values, constant values and a stack. By compiling into a program, the compiler can apply further optimizations, producing a compact representation that runs efficiently. The *compiled-knowledge* library has been released publicly [126].

The proposed synthetic data generator called Synthorus was written in Python using CK library. It was also released publicly [127]. This Python custom synthetic data generator library is used to specify and build generative models and record simulations. It also includes privacy and utility reporting features. Since it is built on top of CK, the CK-WMC sampler introduced in chapter 3 of this thesis is available for use. The differential privacy framework mentioned in chapter 4 is also part of this synthetic data generator library.

The proposed synthetic data generator as implemented here requires the user to define the model structure and certain key parameters. After the structure is specified, parameter learning can be done algorithmically. In this implementation, this is achieved through a model specification file, which typically includes the following elements.

1. **Data Sources**: The file location in the system of data sources for all entities need to be specified.

2. **Random variables and their possible values**: The random variables need to be defined for each data source. These are fields (i.e. columns) in the synthetic data tables. Typically, the values for these random variables are configured to be inferred from their respective data sources.

3. **Cross tables and DP parameters**: The novel decomposition scheme for Differential Privacy (DP) mentioned in the last chapter was used throughout the evaluation process of this synthetic data generator. Therefore, the cross tables need to be specified from the random variables. They dictate which reference statistics can be preserved for synthetic data records. The differential privacy parameter values that controls noise injection into the cross tables should also be specified.

4. **Entities and their relationships**: The random variables, primary keys and foreign keys must be specified for every entity. This controls the relationship and the sampling order among the entities.

5. **Simulation parameters**: This refers to the sampling values and the cardinality that need to specified for the model and data generation process.

After the specification file is created the next step is to extract the clean and noisy cross tables from the given model spec. The cross tables are created in accordance with the DP parameters specified. The DP framework used here is the novel decomposition scheme explained in chapter 4. A simulator object is created based on the model spec and the desired sampler. The sampler used here is the CK-WMC sampler explained in chapter 3

5.5 Privacy composition across linked tables

The proposed synthetic data generator uses the novel decomposition scheme for differential privacy described in Chapter 4. However, when generating synthetic relational databases, privacy loss accumulates across multiple linked tables. This is because, in a relational database, the same individual often appears in multiple tables. Each differentially private release that touches their record contributes to that person's total privacy risk. Total epsilon is at most the sum of the epsilons for each mechanism that used the person's data.

Privacy composition was described in section 2.3.5. Consider k generation processes each of which satisfies ϵ_i differential privacy. Then their sequential execution on the same dataset satisfies $\sum_{i=1}^{k} \epsilon_i$. Consequently, the total privacy loss for generating a relational database of k linked tables is the sum of the privacy budgets allocated to each individual table's generation process. Let ϵ_{total} be the global privacy budget for the entire relational dataset. This budget must be shared across the related entities such that $\sum_{i=1}^{k} \epsilon_i \leq \epsilon_{\text{total}}$. There are two different strategies to allocate the privacy budget.

1. **Uniform Allocation**: Here the total privacy budget is distributed equally across all components or tables in a relational dataset. If there are k tables and a global privacy budget of ϵ_{total}, each table receives an identical budget allocation defined as $\epsilon_i = \frac{\epsilon_{\text{total}}}{k}$. This method simplifies budget tracking and guarantees the privacy composition bound is not violated. The main limitation of this method is that it ignores the structural heterogeneity of relational databases. A table with two categorical variables receives the same budget as a complex table with fifty continuous variables. Consequently, complex tables suffer severe utility degradation due to excessive noise, while simple tables receive more budget than necessary to maintain their statistical properties.

2. **Weighted Allocation**: Weighted allocation assigns a specific fraction of the global privacy budget to each table based on its characteristics or analytical importance. Each table i is assigned a weight w_i, such that the sum of all weights equals 1. The budget for table i is calculated as $\epsilon_i = w_i \epsilon_{\text{total}}$. This approach optimizes the overall statistical utility of the synthetic database. It directs the privacy budget to where it is most needed to preserve data fidelity. However, determining the optimal weights adds complexity to the generation pipeline. The optimal weights can be determined on the basis of dimensionality, sensitivity, analytical priority etc.

5.6 Evaluation

The proposed generator was evaluated to examine its computational efficiency and referential integrity. A significant reduction in computational cost is expected for the proposed generator. This comes from three different parts of the data generation process. First, the proposed generator can natively handle multiple related tables. When compared to a typical synthetic data generator that is forced to work with one large high-dimensional table, the proposed generator can take advantage of the reduction in dimensionality to reduce the cost of data generation. Second, the proposed generator uses the novel Laplacian decomposition scheme for differential privacy introduced in chapter 4 to provide privacy guarantees. As demonstrated in chapter 4, the proposed decomposition method is more efficient than the standard method. Finally, the proposed generator emits data using CK-WMC sampler proposed in chapter 3. Compared to other independent samplers, the CK-WMC sampler was shown to be the most efficient in terms of computational cost.

The proposed generator is expected to generate synthetic data with referential integrity across all related tables. This is due to the hierarchical tree structure mentioned earlier. The hierarchy of simple probabilistic models represent all entities with relationships of definable fixed or random cardinalities. All relationships between tables are represented as conditional dependencies in the Bayesian Networks that represent the entities. Both the computational cost and referential integrity of the proposed generator was evaluated by comparing it against other state-of-the-art generators. In the following sub-sections, I will provide the details of the workloads, baselines and measures used during the evaluation process.

5.6.1 Workloads

The evaluation workloads are multi-table relational datasets that I categorized into small, medium and large.

Table 5.1: Relational datasets and their sizes.

Size	Relational Dataset	No. of Tables	File Size
small	Restbase	3	3MB
medium	Walmart	3	13.73MB
medium	Countries	4	9.8MB
large	Financial	8	78.8MB

The 'Restbase', 'Countries' and 'Financial' datasets were taken from the CTU relational

dataset repository [30]. 'Walmart' was one of the available example datasets that come with the 'sdv' Python library. There is a version of 'Walmart' available in the CTU relational dataset repository. But that was not the version used here for evaluation.

5.6.2 Baselines

Two synthetic data generators were used as baselines for evaluation: (1) DataSynthesizer [112] and (2) Synthetic Data Vault [107].

DataSynthesizer uses a greedy algorithm (PrivBayes [173]) to construct its Bayesian network. This method begins by randomly selecting a root attribute and then iteratively adds remaining attributes as nodes by choosing, for each, the set of parent attributes that maximizes (differentially private) mutual information, subject to a limit on the maximum number of parents. DataSynthesizer does not natively support multi-table relational datasets. It only supports a single table dataset. Therefore, in this evaluation, I used inner join from the 'pandas.merge' Python library to merge all the tables of the given dataset. However, the implementation of differential privacy in DataSynthesizer [112] provides privacy guarantees.

On the other hand, Synthetic Data Vault natively supports multi-table relational datasets [107]. It works by first modeling the statistical properties of each table using techniques such as Gaussian copulas to capture the marginal distributions and inter-column dependencies. The framework then extends this modeling to multiple related tables through methods like Conditional Parameter Aggregation, enabling it to preserve the relational structure of the original data. SDV does not have differential privacy. It can only obfuscate the data by introducing additional noise or perturbation in the model parameters [107].

5.6.3 Measures

The synthetic data generator is evaluated on each workload using two measures: (1) data generation time and (2) referential integrity.

The synthetic data generation time includes the time taken to create PGMs, the time taken to create the clean/noisy crosstabs and the sampling time to emit the required data. Since I am using the CK-WMC sampler, the sampling time includes the time taken to transform each PGM into an arithmetic circuit and then into a WMC program. All measurements were from processes executed on a Lenovo laptop with 32GB of LPDDR5X-6400MHz RAM and an AMD Ryzen 7 7840HS processor. The generation time is presented here as training time and sampling time. The evaluation process for DataSynthesizer and Synthetic Data Vault (SDV) baseline generators required different considerations.

As mentioned earlier, DataSynthesizer cannot natively handle multi-table relational datasets. The synthetic data generation was done after combining all the relevant tables into a single table using the inner join function in 'pandas.merge' Python library. Since the proposed generator can natively handle multi-table relational datasets, I manually created entity diagrams for all four workloads. The synthetic data generation time for the proposed generator was evaluated by using these diagrams to create the specification files (mentioned in section 5.4). DataSynthesizer has implemented differential privacy. Therefore, generation time was measured for selected epsilon values ranging from 0.1 to 1 for both DataSynthesizer and the proposed synthetic data generator. The measurements were carried out on all four workloads. The training time and sampling time for both DataSynthesizer and the proposed generator for the first 3 workloads are given as graphs. For the last workload, the data is given in a table.

Both SDV and the proposed generator can handle multi-table relational datasets. However, unlike the proposed generator, SDV has multi-parent support. Therefore separate representations were created for SDV. These representations correspond to the actual relational schema in the CTU relational repository [30]. The representation of the 'Financial' dataset for both proposed generator and SDV is shown in the next sub-section. Representations of the other workloads are given in Appendix B of this thesis. The equivalent of specification files were created for SDV. They are called metadata files in SDV. As previously mentioned, differential privacy (DP) is not implemented in SDV. So for evaluating the proposed generator against SDV, I am comparing the generation time of the proposed generator with no DP against the generation time of SDV for all four workloads. The training time and sampling time for all four workloads for both SDV and the proposed generator is represented in a single table.

Referential integrity was measured using a score that ranges from 0 (worst) to 1 (best). This referential integrity score is from the 'sdvmetrics' Python library. Let s be the synthetic data. Then s_f represents the foreign key values in s and s_p represents the matched primary key value for those foreign key values. This score refers to the proportion of foreign key values that can be observed in the primary key column.

$$\text{score} = \frac{|s_f, s_f \in s_p|}{|s_f|} \tag{5.4}$$

If any foreign key value is missing, this score counts it as valid. Therefore it will still be considered in the numerator.

5.6.4 Relational Schema for 'Financial' workload

The 'Financial' relational dataset has 8 tables. Figure 5.3 shows the original relational schema from the CTU relational repository [30]. I used the same schema for SDV generator. All tables except 'disp' has zero or one foreign key. The 'disp' table contains two foreign keys: one referencing the 'account' table and another referencing the 'client' table. Note that SDV does not have differential privacy implemented.

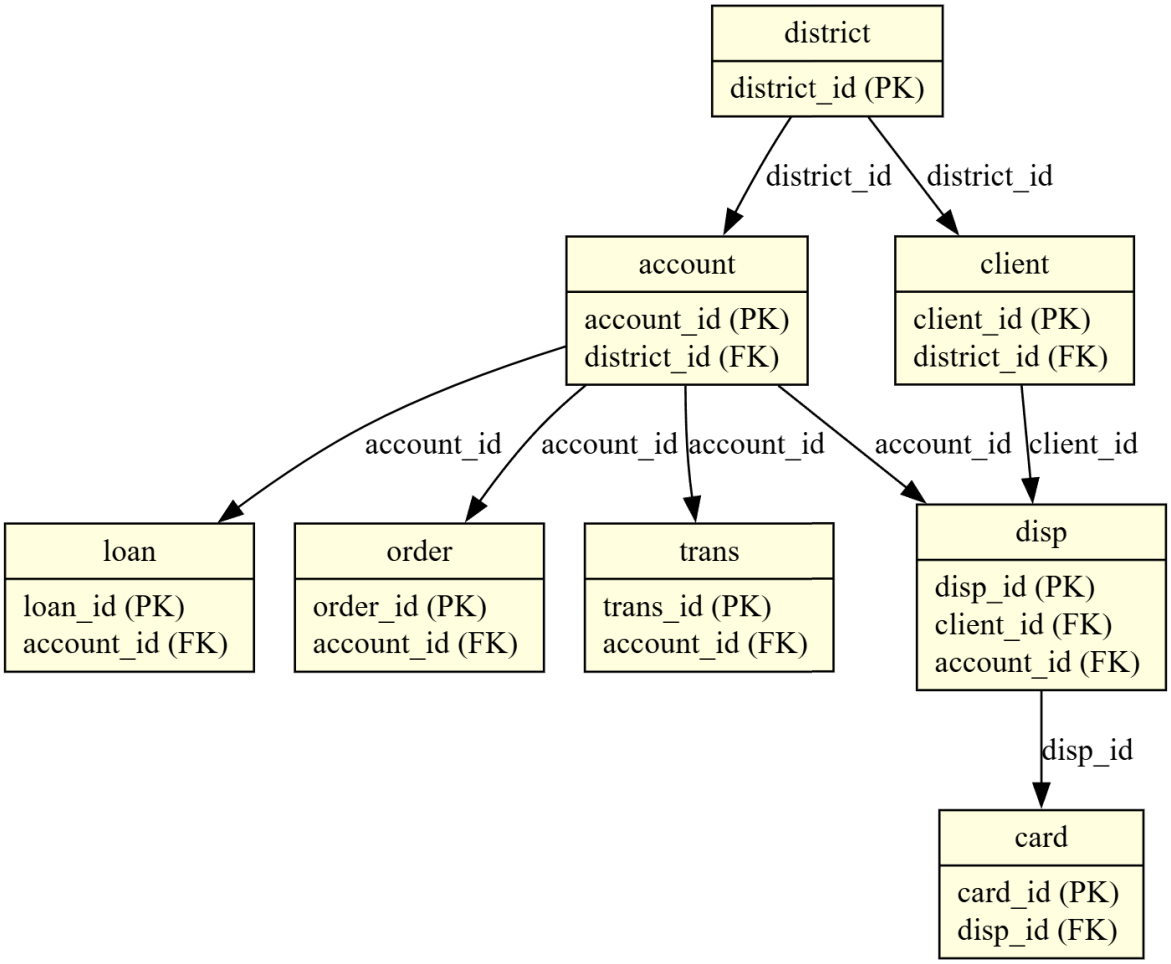


Figure 5.3: **Financial Dataset**: This structure represents the original schema from the CTU relational repository. Only the Primary Key (PK) and Foreign Key (FK) are shown.

The proposed generator was designed to handle one-to-many relationships. Every table of the relational dataset is represented by an entity. The hierarchical structure of the proposed generator requires every entity to have only one foreign key. Therefore I modified the schema to make the generator compatible with the 'Financial' dataset. The 'district' table was treated as a reference table, so no corresponding entity was created. A relationship between 'client' and 'account' entities was added. The relationship between the 'client' entity and 'disp' entity was deleted with the assumption that 'disp' would still be linked to 'client' through 'account'. Figure 5.4 shows the representation used for the proposed generator.

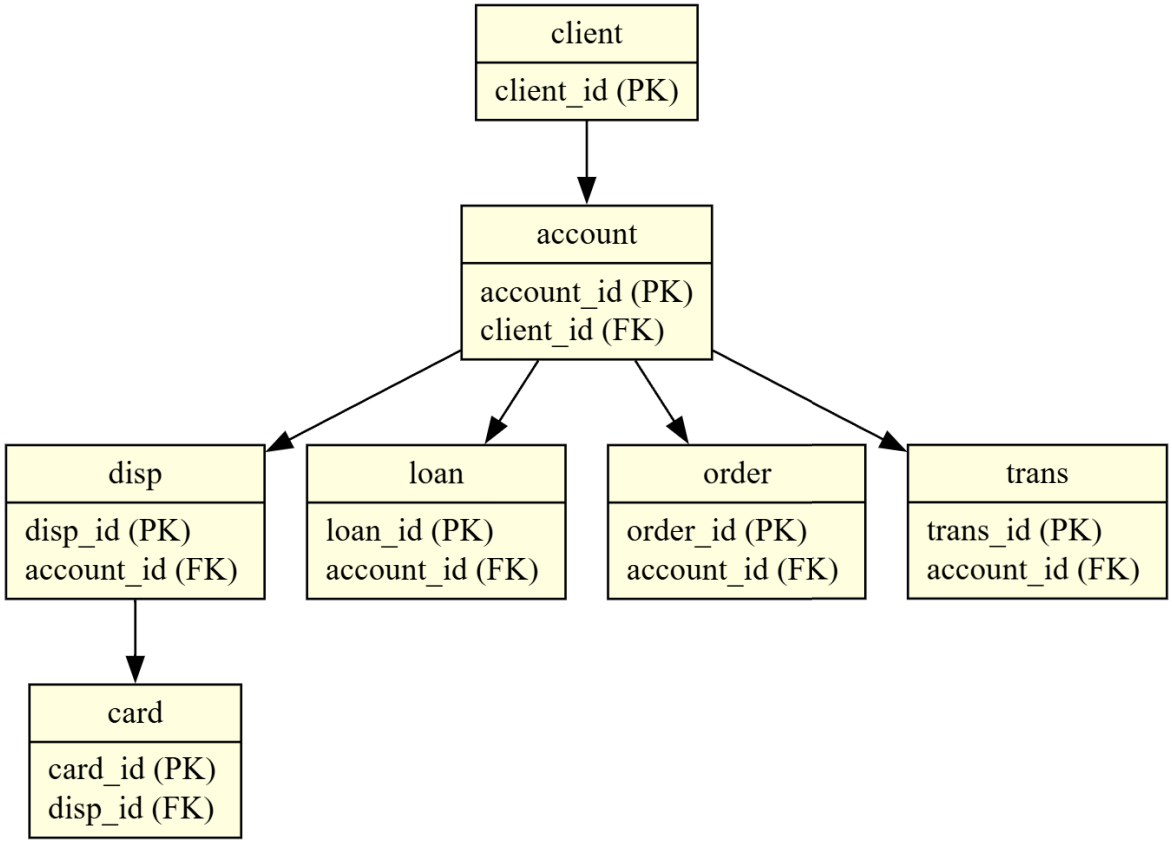


Figure 5.4: **Financial Dataset**: This structure was used for proposed generator. Not all columns in the tables of the dataset are shown. Only the Primary Key (PK) and Foreign Key (FK) are shown.

The other workloads ('Restbase', 'Walmart' and 'Countries') have much simpler schema. Their representations are given in Appendix B. Note that the representations in figure 5.3 and 5.4 was used for evaluating SDV and the proposed generator. DataSynthesizer was evaluated by merging all the tables in the dataset using the inner join function in 'pandas.merge' Python library.

5.7 Results

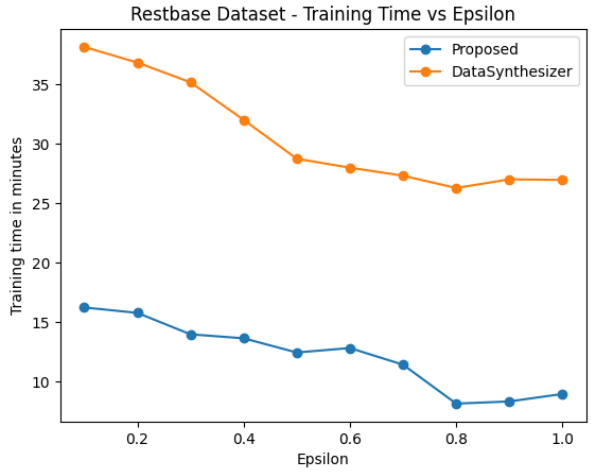
This section consists of two parts. Section 5.6.1 compares the synthetic data generation time of the proposed generator against the two baselines for all workloads. Section 5.6.2 reports on the referential integrity of the proposed generator and the baseline generators. The two baselines are DataSynthesizer [112] and Synthetic Data Vault [107].

5.7.1 Synthetic Data Generation Time

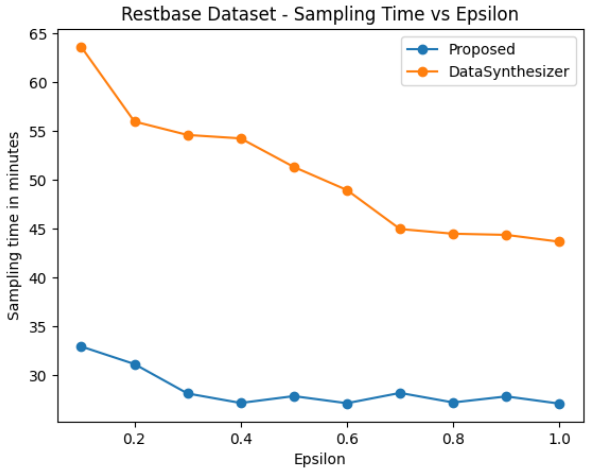
Synthetic data generation time measured as wall-clock time includes the time taken to train the model and then sample from the model. Figure 5.5 compares the training time and sampling time of the proposed generator against the DataSynthesizer [112] generator for the first three workloads. The proposed generator achieved the best generation times across all three workloads. For the smallest workload, 'Restbase', it was 2 to 3 times faster than DataSynthesizer in both training and sampling. On medium-sized workloads ('Walmart' and 'Countries'), the proposed generator demonstrated an even more substantial improvement, performing 5 to 7 times faster. This indicates that the performance advantages of the proposed generator become increasingly pronounced with larger workloads. Table 5.2 compares the training time and sampling time of the proposed generator with the DataSynthesizer for the largest workload 'Financial'. These results are presented in a table rather than graphically, as DataSynthesizer's training time exceeded the preset calculation limit of 5000 minutes. The proposed generator demonstrated superior performance here as well.

Table 5.2: Proposed generator and DataSynthesizer synthetic data generation time for 'Financial' dataset.

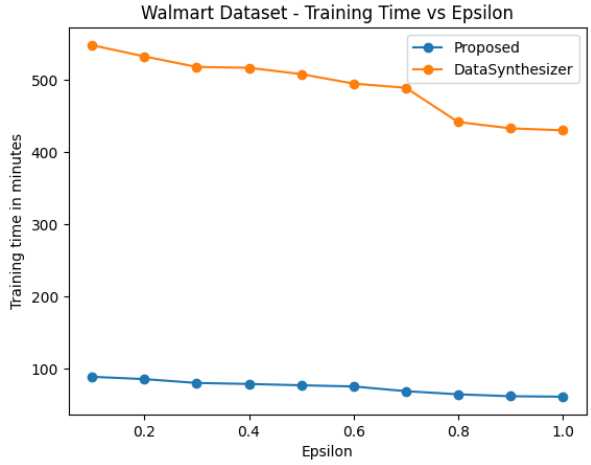
Epsilon	**Proposed generator training time (in min)**	**Proposed generator sampling time (in min)**	**DataSynthesizer training time (in min)**	**DataSynthesizer sampling time (in min)**
No DP	462.6977	536.2672	>5000	-
0.1	740.2538	599.3396	>5000	-
0.2	728.4735	575.4528	>5000	-
0.3	680.8628	572.7132	>5000	-
0.4	667.8766	566.4502	>5000	-
0.5	651.5558	561.4209	>5000	-
0.6	654.7318	562.3601	>5000	-
0.7	625.9512	561.2579	>5000	-
0.8	598.9884	558.5894	>5000	-
0.9	574.2364	557.6823	>5000	-
1	555.8741	553.7044	>5000	-



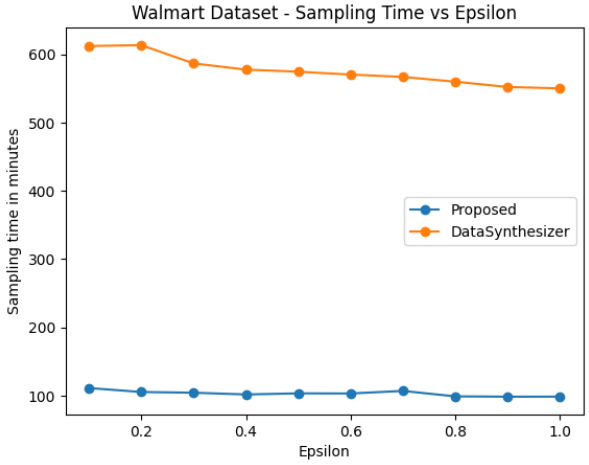
(a) Restbase - Training Time vs Epsilon



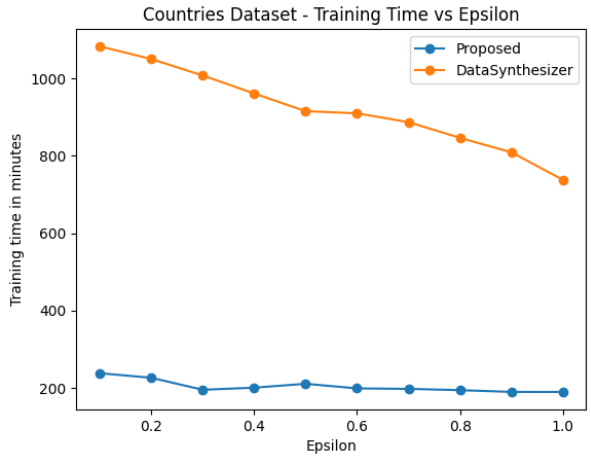
(b) Restbase - Sampling Time vs Epsilon



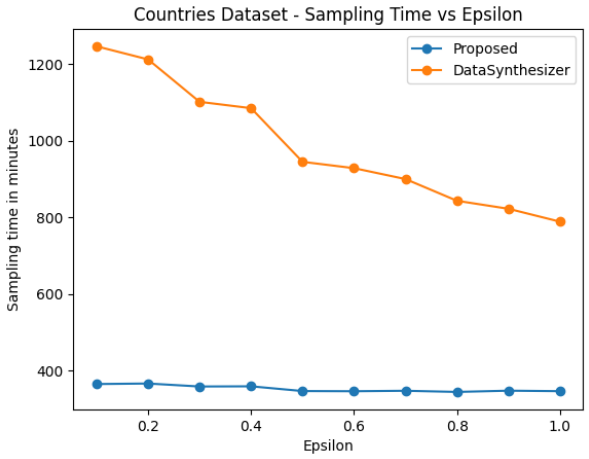
(c) Walmart - Training Time vs Epsilon



(d) Walmart - Sampling Time vs Epsilon



(e) Countries - Training Time vs Epsilon



(f) Countries - Sampling Time vs Epsilon

Figure 5.5: The synthetic data generation time results of the proposed generator vs DataSynthesizer for 'Restbase', 'Walmart' and 'Countries' datasets.

Overall the proposed generator had the best synthetic data generation time for all four workloads. The hierarchical tree structure of proposed generator provided a significant advantage over DataSynthesizer for selected epsilon values ranging from 0.1 to 1. The novel decomposition scheme for differential privacy and the CK-WMC sampling algorithm contributed towards this as well. The value of this contribution can be observed by considering synthetic data generation time results of 'Synthetic Data Vault'. When comparing results of the proposed generator with no DP against the results of 'Synthetic Data Vault', the proposed generator still performs better.

Table 5.3 shows the training time and sampling time of the proposed generator and SDV. The proposed generator performs better overall. SDV results were calculated using the 'HMASynthesizer' algorithm in the 'sdv' Python library. It is only algorithm available for relational datasets in the community edition of SDV. The paid enterprise edition of the product may have had other algorithms, but that was unavailable for use. 'HMASynthesizer' is optimized for around 5 tables. Hence it failed for the 'Financial' workload.

Table 5.3: Proposed generator and Synthetic Data Vault (SDV) synthetic data generation time for all datasets.

Dataset	**Proposed generator training time (in min)**	**Proposed generator sampling time (in min)**	**SDV training time (in min)**	**SDV sampling time (in min)**
Restbase	5.9311	25.5714	27.8347	22.6376
Walmart	30.5381	94.3795	222.5118	198.993
Countries	127.5801	334.6735	719.4604	470.066
Financial	462.6977	536.2672	*failed*	*failed*

5.7.2 Referential Integrity

Referential integrity score was calculated for DataSynthesizer with no DP using all four workloads. The values were as follows:

1. **Restbase**: 0.97
2. **Walmart**: 0.969
3. **Countries**: 0.73
4. **Financial**: 0.52

Note that referential integrity is less than 1 for all four workloads. This means that some relationships between tables are broken when the tables are merged. The proposed generator and Synthetic Data Vault had referential integrity values of 1 for all datasets.

5.8 Conclusions

A novel cost-efficient and differentially private synthetic data generator that maintains referential integrity using PGMs was developed and evaluated. It utilized the decomposition scheme for Laplacian differential privacy proposed in Chapter 4 and the CK-WMC sampler proposed in Chapter 3 to reduce computational cost. The main contribution here is the proposed algorithm used to construct a complex probabilistic model to ensure referential integrity. It did this by creating a hierarchy of simpler probabilistic models with each model representing entities with relationships of definable fixed or random cardinalities. The proposed generator also provides multiple mechanisms for managing the cardinality between entities. The proposed generator was evaluated using four multi-table relational datasets as workloads. Two metrics were considered for this evaluation: (1) data generation time and (2) referential integrity. The generation time is measured for DP values ranging from 0.1 to 1. The referential integrity score that was used ranged from 0 (worst) to 1 (best). The proposed generator had the best data generation time when compared against DataSynthesizer and Synthetic Data Vault. Both the proposed generator and the Synthetic Data Vault had referential integrity scores of 1 for all datasets. DataSynthesizer had lower referential integrity scores. This was because DataSynthesizer did not natively support multi-table relational datasets.

CHAPTER 6

DISCUSSION & CONCLUSION

The first contribution of this thesis was two methods for drawing independent samples from a PGM, where the PGM is compiled into an arithmetic circuit. The two novel samplers (CK-WMC and CK-MAR) which were based on inverse transform sampling showed a statistically significant reduction in computational cost. Although dependent samplers were faster for larger models, the proposed samplers dominated when compared against other independent samplers. As part of the evaluation for this work, the factorised histogram-intersection method was introduced. Typical metrics like KL divergence for calculating the accuracy of empirical distributions are not practical when dealing with large discrete supports. Therefore the computationally efficient factorised histogram-intersection method was introduced.

Independent sampling methods are important for synthetic data generators because they do not introduce any additional statistical relationships, beyond that which is in the model. But the proposed methods have applications beyond synthetic data generation. The reduction in computational cost achieved by the proposed sampling algorithm will make the proposed methods valuable for any application that requires independent sampling from a PGM. For instance, various applications require inferencing to determine probability values for required propositions (within the domain of the model). Many healthcare applications routinely use PGMs, especially Bayesian networks (BNs), for decision support in diagnosing diseases and planning treatments [123]. Inferencing via sampling (i.e. approximate inference) is more efficient in handling such PGMs with thousands of random variables than exact inference. By sampling from these models, it is possible to simulate

patient states or progression of disease to help medical reasoning. Similar applications can be found in finance [138], robotics [44] etc.

The second contribution of this thesis was the novel decomposition scheme for Differential Privacy (DP) that reduced the computational cost of Laplacian noise perturbation process. The time taken to generate noise for the proposed method was compared against the standard method (i.e. without the new decomposition scheme) over a range of epsilon values. The time taken by the standard method was approximately three times that of the proposed method. The proposed method had similar accuracy to the standard method. Although, the proposed method was demonstrated here using Bayesian Networks (BN), the method can work for any statistical model that works on counts. This is because the noise addition process does not rely on the structural assumptions of Bayesian Networks. On the other hand, state-of-the-art synthetic data generators like PrivBayes [173] rely more on Bayesian Networks. They rely on the ability of Bayesian Networks to decompose a high-dimensional data distribution into a collection of low-dimensional conditional distributions.

In PrivBayes [173], the BN is constructed using a DP algorithm that utlizes the exponential mechanism for adding noise. The PrivBayes [173] algorithm selects attribute-parent pairs based on a modified mutual information score. All relevant DP distributions of the data are then computed in the sub-spaces of the Bayesian Network via the Laplace mechanism. The contribution reported for PrivBayes [173] is the application of noise partial joint probabilities rather than the full joint probability distribution. The proposed method builds on this to demonstrate that decomposing the noise process yields additional efficiency gains. So PrivBayes needs Bayesian networks to achieve differential privacy, and it isn't a generic framework that can be directly applied with any statistical model. However, as demonstrated, the proposed method can work for any statistical model (that relies on counts) and at the same time provide privacy guarantees in datasets with complex dependencies. In the proposed method, the min-cell-size parameter practically induces further noise in the model, beyond that added by the Laplacian process. In particular, when a cross table has low counts then the noise from enforcing min-cell-size will be more noticeable. However, low count combinations are exactly the states DP should protect. Low counts may be caused by rare combinations of states or a cross table that has many random variables.

The final contribution of this thesis is creating a novel cost-efficient and differentially private synthetic data generator that maintains referential integrity using PGMs. The proposed generator constructs a complex probabilistic model that ensures referential integrity. It does this by creating a hierarchy of simpler probabilistic models with each model repre-

senting entities with relationships of definable fixed or random cardinalities. The proposed generator was evaluated in terms of data generation time and referential integrity. DataSynthesizer [112] and Synthetic Data Vault [107] were the two state-of-the-art publicly available synthetic data generators used as baselines. The workloads were four multi-table relational datasets taken from the CTU Relational Dataset Repository [30]. The proposed generator had the best data generation time when compared against DataSynthesizer and Synthetic Data Vault. Both the proposed generator and the Synthetic Data Vault had referential integrity scores of 1 for all datasets. DataSynthesizer had lower referential integrity scores. This was probably because DataSynthesizer did not natively support multi-table relational datasets.

An important advantage that the proposed synthetic data generator had against DataSynthesizer and Synthetic Data Vault was the reduction in computational cost. Both the proposed generator and DataSynthesizer have strong privacy guarantees via differential privacy. Note that the proposed generator used the new decomposition scheme for DP in the evaluation. Synthetic Data Vault does not have DP based privacy guarantees. In Synthetic Data Vault, users can only perturb the model parameters to adjust privacy. This does not provide well defined guarantees like DP. Both the proposed generator and Synthetic Data Vault can natively handle multi-table relational datasets. The user needs to merge the tables manually before doing any work with DataSynthesizer.

Another advantage of the proposed generator is the ability to manage cardinalities. This is important in many applications. For instance, consider patient journey modeling. Patient journey modeling is a technique used to map and analyze the full experience a patient has within a healthcare system. The Lumos database is a statewide linked database representing the health system in New South Wales, Australia. It connects general practice (GP) electronic records with records from emergency, hospital and other health services. All the health records for an individual patient are linked with a central key. The proposed generator is suitable for generating synthetic data from Lumos and other similar datasets. Managing cardinalities is important here because of the variability and complexity of patient experiences.

In the proposed generator, the number of records is determined by using a set of stopping functions (all of which fall under one of five stopping conditions). Patient journeys consist of a series of medical visits, procedures, and/or consultations. The exact number of such events a patient has can be hard to predict. This results in a random one-to-many relationship. The stopping functions allow the model to handle this variability by managing cardinality and creating an accurate representation of events. Examples of stopping func-

tions that fall under each of the stopping conditions is given in subsection 5.1. Managing cardinality also helps to ensure referential integrity across datasets by enforcing clear, consistent rules for how records in related tables can link to each other. This includes enforcing relationship constraints, preventing orphaned records and maintaining consistency.

Limitations and Future Work

The proposed generator has two limitations: (1) it is designed to work with discrete distributions having finite number of states and (2) each entity can have only one foreign key.

The first limitation of the proposed generator is that the two proposed novel sampling methods were designed for discrete distributions having a finite number of states. The feasibility of the methods for continuous distributions were not studied. It would be challenging to extend these methods to continuous distributions. A reason for this is that inverse transform sampling for continuous distributions involves calculating and inverting the CDF. CDFs of continuous distributions might not have closed-form inverses or could require numerical approximation methods, significantly increasing computational cost. Another reason is that arithmetic circuits efficiently represent and compute probabilities through sums and products, which naturally fit discrete cases. In continuous distributions, integrals would need to represented using arithmetic circuits which is more complex.

The proposed sampling methods were also not investigated for random variables with countable and infinite states. Similar to continuous distributions, they relate to very large domains (e.g. floats or large integers). New techniques may need to be developed to handle these kinds of distributions since large domains need special attention to be encoded as efficient arithmetic circuits. This is considered as potential future work.

The second limitation of the proposed synthetic data generator is that it can only handle one-to-many relationships between entities. In other words each entity can have only one foreign key. The hierarchical tree structure of the proposed generator requires every entity to have only one foreign key. Consider the relational schema shown in Figure 5.3. Since 'disp' had two foreign keys, I could not use this structure directly. As explained in sub-section 5.5.4, it had to be modified (entity diagram shown in figure 5.4). Developing a method to directly handle multiple foreign keys is considered as potential future work for improving the proposed generator.

Apart from these two limitations, there is also the difficulty in handling high-cardinality data. When an attribute has very high cardinality, many categories have low or zero empirical counts. The Laplace noise may dominate the signal. For high-cardinality attributes,

many bins are dominated by noise rather than signal. Negative noisy counts need to be handled by truchation – this can introduce bias. Probability mass gets spread thinly across many categories, degrading utility for rare but meaningful values. In graphical models, this can show up as poor parameter estimation for conditional probability tables with large domains. This can result in synthetic data that fails to preserve realistic frequency structure. This is not a problem with the proposed method, per se but a fundamental tension between DP noise and sparse domains.

The efficiency of algorithm 2 in chapter 4 depends on the relationship between the number of zero-weight rows (m), the number of cells that cross the threshold (n), and the probability (α) of that crossing. Line 3 of the algorithm involves randomly selecting n different combinations from the space of zero-weight cells. This step relies on rejection sampling, which works efficiently only if $n \ll m$. The value n is drawn from a binomial distribution $B(m, \alpha)$. Therefore, for n to remain much smaller than m, the probability α must be very small. The probability α is defined as $\frac{1}{2}\exp\left(-\frac{r}{b}\right)$. For α to be small, the minimum cell size parameter (r) must be large relative to the noise scale (b). In situations where a very strict privacy budget is required combined with a low suppression threshold r, the probability α increases. Consequently, n grows larger. In this scenario, the rejection sampling step becomes inefficient because it would be repeatedly guessing combinations that are already taken or are non-zero. This destroys the computational advantage and can make the decomposition scheme slower than the standard full-state-space method. However this performance degradation can be easily prevented. The meta-algorithm simply needs to revert back to the standard method when n is no longer significantly smaller than m.

Finally, it should be noted that the methods presented in this thesis apply only to graphical models that are tractable. Publications by Darwiche [39] [41] provide the criteria to determine which models are tractable. The first paper [39] demonstrates how compiling a Bayesian network into a multivariate polynomial, represented as an arithmetic circuit, enables exact inference and partial derivative computation in linear time relative to the circuit size. In a recent 2022 paper, Darwiche [41] reviews the core properties, techniques, and historical milestones of compiling models into tractable circuits for linear-time probabilistic and logical reasoning.



APPENDIX A - RESULTS RAW DATA

Table A.1: Raw accuracy values for all samplers (runs 1–5) used for evaluation in chapter 3. Data from figure 3.1

Sampler	Value 1	Value 2	Value 3	Value 4	Value 5
CK-WMC	0.998555357	0.996750315	0.998919632	0.996507399	0.998471167
CK-MAR	0.997314887	0.997847002	0.999181747	0.999694690	0.996166914
CK-Reject	0.997070612	0.998566563	0.998975666	0.996655483	0.995955684
BUGS-Gibbs	0.998338854	0.998786441	0.998576207	0.995571345	0.997773736
CK-MH	0.998296017	0.998141745	0.998399097	0.995211628	0.995056582
CK-Forward	0.998846598	0.997313509	0.999041781	0.997159878	0.998575963
PGM-Reject	0.997809996	0.998305018	0.996166914	0.995756908	0.996747871

Table A.2: Raw accuracy values for all samplers (runs 6–9) used for evaluation in chapter 3. Data from figure 3.1

Sampler	Value 6	Value 7	Value 8	Value 9
CK-WMC	0.996056968	0.998004471	0.998749003	0.999637669
CK-MAR	0.996953891	0.997166393	0.998288461	0.999528489
CK-Reject	0.997159878	0.998575963	0.996900412	0.999354762
BUGS-Gibbs	0.996236667	0.997864158	0.997598071	0.999196273
CK-MH	0.995382127	0.997626156	0.996563798	0.998489197
CK-Forward	0.996900412	0.997159888	0.998575973	0.996900422
PGM-Reject	0.996991257	0.996526533	0.997809996	0.998305018

Table A.3: Accuracy of the proposed method (introduced in chapter 4) for laplace noise decomposition in differential privacy. The highest, lowest, mean and median histogram-intersection (HI) values are shown. Data used in figure 4.3

Epsilon	**Lowest**	**Highest**	**Mean**	**Median**
0.01	0.000176	0.993284	0.952125	0.624836
0.03	0.000585	0.995626	0.985489	0.653549
0.05	0.001043	0.997713	0.990812	0.663945
0.07	0.001457	0.997771	0.992153	0.669596
0.09	0.001877	0.999396	0.993782	0.674510
0.1	0.002846	0.999244	0.993627	0.624300
0.2	0.003550	0.999702	0.997849	0.640780
0.3	0.004893	0.999636	0.997886	0.653973
0.4	0.006460	0.999810	0.998691	0.666665
0.5	0.008278	0.999777	0.998908	0.678286
0.6	0.009579	0.999853	0.999050	0.689586
0.7	0.012689	0.999862	0.999470	0.701174
0.8	0.016377	0.999959	0.999338	0.712052
0.9	0.019542	0.999924	0.999678	0.723490
1	0.021218	0.999917	0.997850	0.683553
1.5	0.048116	0.999914	0.998770	0.734770
2	0.106754	0.999966	0.999181	0.782791
2.5	0.215412	0.999980	0.999356	0.829263
3	0.389201	0.999978	0.999593	0.874933
3.5	0.587321	0.999982	0.999583	0.916530
4	0.748425	0.999986	0.999644	0.946443
4.5	0.843436	0.999991	0.999664	0.964312
5	0.902540	0.999993	0.999695	0.975429
5.5	0.928470	0.999995	0.999722	0.980492
6	0.940066	0.999986	0.999781	0.983704
6.5	0.952136	0.999991	0.999790	0.986324
7	0.957964	0.999991	0.999819	0.987691
7.5	0.961300	0.999995	0.999824	0.988675
8	0.964788	0.999994	0.999837	0.989736

Table A.4: Accuracy of the standard method for differential privacy (used for comparing against proposed method in chapter 4). The highest, lowest, mean and median histogram-intersection (HI) values are shown. Data used in figure 4.3

Epsilon	Lowest	Highest	Mean	Median
0.01	0.000175	0.991057	0.956447	0.626613
0.03	0.000593	0.995605	0.982311	0.653319
0.05	0.001041	0.998584	0.990673	0.663973
0.07	0.001473	0.998876	0.992232	0.670038
0.09	0.001867	0.999051	0.993847	0.674843
0.1	0.002731	0.999573	0.994754	0.625042
0.2	0.003558	0.999900	0.997808	0.640924
0.3	0.004830	0.999726	0.998088	0.654143
0.4	0.006510	0.999864	0.998423	0.666480
0.5	0.008273	0.999861	0.998871	0.678310
0.6	0.010012	0.999821	0.999127	0.689872
0.7	0.012413	0.999888	0.999310	0.700942
0.8	0.016988	0.999911	0.999355	0.712282
0.9	0.018872	0.999910	0.999446	0.723017
1	0.021548	0.999948	0.997814	0.683326
1.5	0.048163	0.999961	0.998719	0.734993
2	0.102663	0.999975	0.999038	0.782818
2.5	0.214426	0.999977	0.999293	0.829342
3	0.389194	0.999977	0.999544	0.875243
3.5	0.586415	0.999982	0.999590	0.916769
4	0.744306	0.999984	0.999645	0.945848
4.5	0.847361	0.999982	0.999705	0.964894
5	0.903473	0.999989	0.999737	0.975032
5.5	0.928415	0.999987	0.999776	0.980562
6	0.943294	0.999988	0.999778	0.983856
6.5	0.951779	0.999992	0.999801	0.986100
7	0.957762	0.999989	0.999820	0.987784
7.5	0.961551	0.999993	0.999840	0.988786
8	0.964094	0.999994	0.999843	0.989694

Table A.5: Total time taken for all crosstabs to generate noise for a selected range of epsilon values (both the standard method and proposed method from chapter 4). Data used in figure 4.5

Epsilon	**Time Baseline**	**Time Proposed**
0.01	642.661515	198.665607
0.03	624.498561	198.765966
0.05	614.260881	189.498279
0.07	641.664195	182.861565
0.09	641.409274	185.324754
0.1	620.523657	178.976827
0.2	594.080235	138.679255
0.3	569.166713	111.371757
0.4	550.976605	92.136088
0.5	533.233756	76.322573
0.6	526.430633	64.050377
0.7	518.399050	53.285170
0.8	503.440114	45.880629
0.9	495.753281	39.118244
1	488.358737	34.262286
1.5	477.515739	19.217064
2	465.442338	14.315880
2.5	467.905475	12.795072
3	468.236976	11.908838
3.5	470.942411	11.417040
4	477.935338	11.271289
4.5	510.559095	11.711972
5	520.267089	12.282902
5.5	521.881700	12.772415
6	513.891789	12.773545
6.5	502.694301	13.293750
7	505.460278	11.797377
7.5	507.511421	11.742761
8	506.088968	11.979658

Table A.6: 'Restbase' dataset: Synthetic data generation time for this workload for both proposed generator and DataSynthesizer (from chapter 5). Data used in figure 5.5

Epsilon	**Proposed generator training time (in min)**	**Proposed generator sampling time (in min)**	**DataSynthesizer training time (in min)**	**DataSynthesizer sampling time (in min)**
No DP	5.9311	25.5714	19.3442	36.9209
0.1	16.2293	32.9701	38.1176	63.6069
0.2	15.7719	31.1792	36.8038	55.9787
0.3	13.9721	28.1677	35.1452	54.6048
0.4	13.6362	27.1976	31.9899	54.2532
0.5	12.4443	27.8971	28.7217	51.3351
0.6	12.8184	27.1721	27.9726	48.9743
0.7	11.4199	28.2237	27.2969	44.9829
0.8	8.1496	27.2484	26.2625	44.5046
0.9	8.3315	27.8671	26.9932	44.3857
1	8.961	27.1309	26.9457	43.6860

Table A.7: 'Walmart' dataset: Synthetic data generation time for this workload for both proposed generator and DataSynthesizer (from chapter 5). Data used in figure 5.5

Epsilon	**Proposed generator training time (in min)**	**Proposed generator sampling time (in min)**	**DataSynthesizer training time (in min)**	**DataSynthesizer sampling time (in min)**
No DP	30.5381	94.3795	383. 9140	511. 9428
0.1	88.7303	111.1425	547. 7736	612. 2108
0.2	85.5639	105.3531	532. 2399	613. 4177
0.3	80.2453	104.3055	517. 6525	586. 7113
0.4	78.84	101.766	516. 4909	577. 6351
0.5	76.9846	103.3539	507. 7229	574. 5685
0.6	75.4043	103.2053	494. 5731	570. 3770
0.7	68.8581	106.9213	488. 8197	566. 9049
0.8	64.5285	98.9968	441. 4055	559. 9767
0.9	61.867	98.4838	432. 6663	552. 2257
1	61.2203	98.5478	429. 9204	550. 1309

Table A.8: 'Countries' dataset: Synthetic data generation time for this workload for both proposed generator and DataSynthesizer (from chapter 5). Data used in figure 5.5

Epsilon	**Proposed generator training time (in min)**	**Proposed generator sampling time (in min)**	**DataSynthesizer training time (in min)**	**DataSynthesizer sampling time (in min)**
No DP	127.5801	334.6735	680.5167	665.3908
0.1	237.9453	364.2846	1082.7883	1246.7205
0.2	226.268	365.6552	1050.1019	1212.4709
0.3	195.1622	357.8707	1007.4995	1101.677
0.4	200.5383	358.3219	960.9798	1085.1037
0.5	210.5921	345.9597	915.2187	944.8969
0.6	198.7205	345.5036	909.6238	928.3304
0.7	197.4579	346.6911	886.7958	899.8554
0.8	194.1652	343.7019	846.2275	843.112
0.9	189.5966	346.9279	808.6729	822.222
1	186.6321	345.6638	737.7596	788.563

APPENDIX B

APPENDIX B - ADDITIONAL FIGURES

The following are the relational data schema I used in Chapter 5 for the proposed generator and Synthetic Data Vault (SDV) generator. The four workloads (i.e. multi-table relational datasets) used were 'Restbase', 'Walmart', 'Countries' and 'Financial'. They were taken from CTU relational dataset repository [30]. Some workloads required different schema for the proposed generator and SDV. This was because SDV could handle multi-parent relationships while the proposed generator could not. 'Restbase', 'Countries' and 'Financial' had to be restructured to make them compatible with the proposed generator. The entity diagram for 'Financial' is shown in Figure 5.3 and 5.4 on Chapter 5. Entity diagrams for the other three workloads are given below.

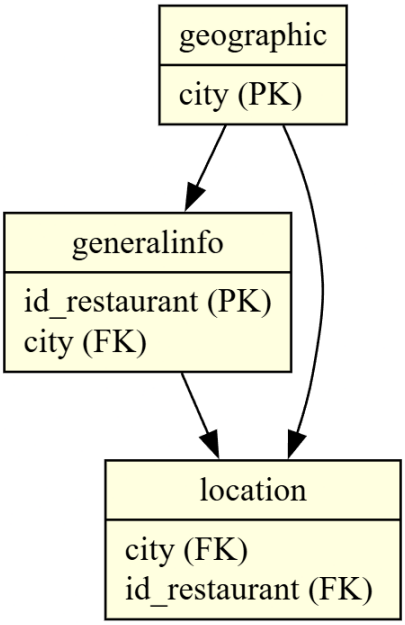


Figure B.1: **Restbase Dataset**: This structure was used for *Synthetic Data Vault* generator. Not all columns in the tables of the dataset are shown. Only the Primary Key (PK) and Foreign Key (FK) are shown.

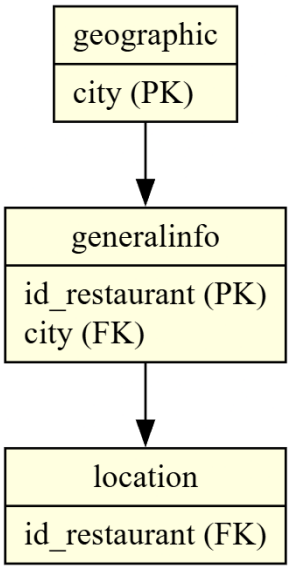


Figure B.2: **Restbase Dataset**: This structure was used for proposed generator. Not all columns in the tables of the dataset are shown. Only the Primary Key (PK) and Foreign Key (FK) are shown.

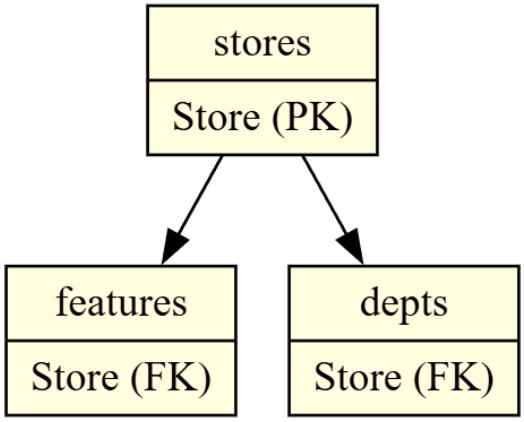


Figure B.3: **Walmart Dataset**: This structure was used for both proposed and *Synthetic Data Vault* generator. Not all columns in the tables of the dataset are shown. Only the Primary Key (PK) and Foreign Key (FK) are shown.

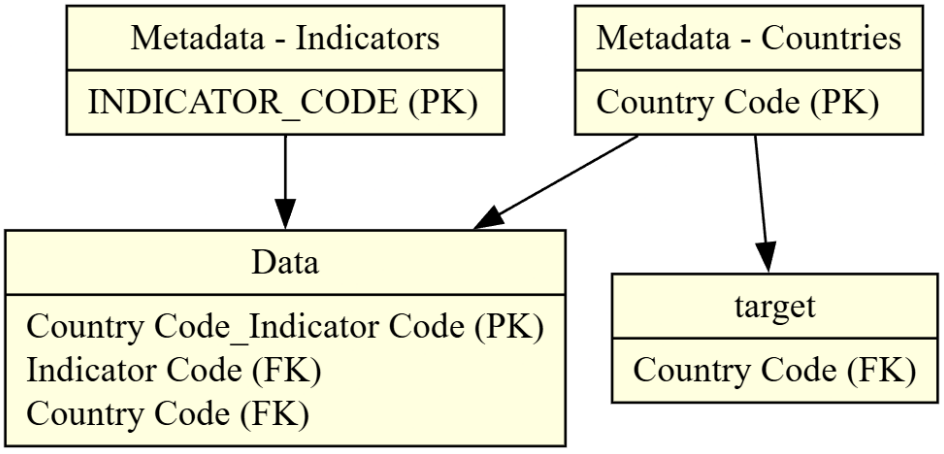


Figure B.4: **Countries Dataset**: This structure was used for *Synthetic Data Vault* generator. Not all columns in the tables of the dataset are shown. Only the Primary Key (PK) and Foreign Key (FK) are shown.

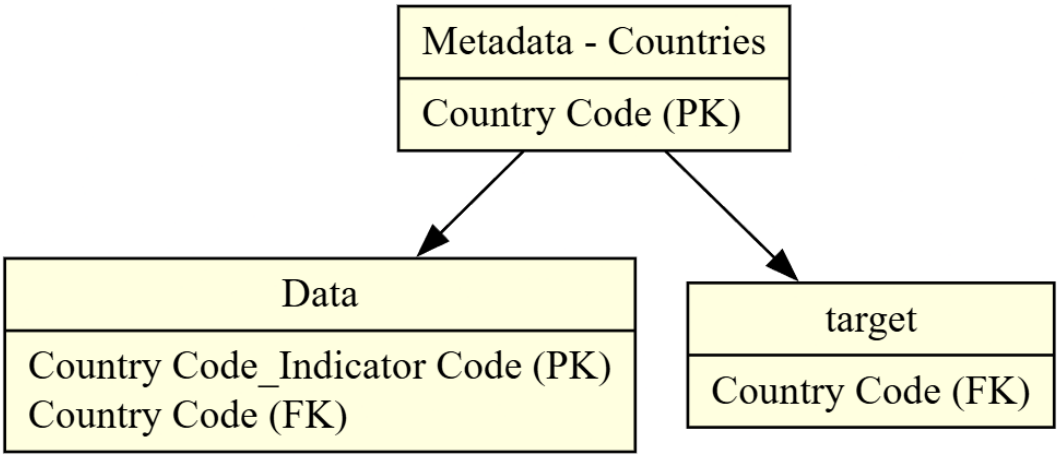


Figure B.5: **Countries Dataset**: This structure was used for proposed generator. Not all columns in the tables of the dataset are shown. Only the Primary Key (PK) and Foreign Key (FK) are shown.

BIBLIOGRAPHY

[1] N. C. ABAY, Y. ZHOU, M. KANTARCIOGLU, B. THURAISINGHAM, AND L. SWEENEY, *Privacy preserving synthetic data release using deep learning*, in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2018, pp. 510–526.

[2] W. ÅGREN AND V. Ú. SOSA, *Hierarchical conditional tabular gan for multi-tabular synthetic data generation*, arXiv preprint arXiv:2411.07009, (2024).

[3] K. AHMED, S. TESO, K.-W. CHANG, G. VAN DEN BROECK, AND A. VERGARI, *Semantic probabilistic layers for neuro-symbolic learning*, Advances in Neural Information Processing Systems, 35 (2022), pp. 29944–29959.

[4] K. AHMED, E. WANG, K.-W. CHANG, AND G. VAN DEN BROECK, *Neuro-symbolic entropy regularization*, in Uncertainty in Artificial Intelligence, PMLR, 2022, pp. 43–53.

[5] A. ALLAM, S. FEUERRIEGEL, M. REBHAN, AND M. KRAUTHAMMER, *Analyzing patient trajectories with artificial intelligence*, Journal of medical internet research, 23 (2021), p. e29812.

[6] H. ASGHAR, M. DING, T. RAKOTOARIVELO, S. MRABET, AND D. KAAFAR, *Differentially private release of high-dimensional datasets using the gaussian copula*, Journal of Privacy and Confidentiality, 10 (2020), pp. 1–38.

[7] B. BALLE, G. BARTHE, M. GABOARDI, J. HSU, AND T. SATO, *Hypothesis testing interpretations and renyi differential privacy*, in International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2496–2506.

[8] A. R. BENAIM, R. ALMOG, Y. GORELIK, I. HOCHBERG, L. NASSAR, T. MASHIACH, M. KHAMAISI, Y. LURIE, Z. S. AZZAM, J. KHOURY, ET AL., *Analyzing medical research results based on synthetic data and their relation to real data results: systematic comparison from five observational studies*, JMIR medical informatics, 8 (2020), p. e16492.

[9] Y. BOUFKHAD, E. GREGOIRE, P. MARQUIS, B. MAZURE, AND L. SAIS, *Tractable cover compilations.*, in Proceedings of the 15th International Joint Conference on Artifical Intelligence - Volume 1, vol. 1, 1997, pp. 122–127.

[10] O. BROADRICK, W. CAO, B. WANG, M. TRAPP, AND G. VAN DEN BROECK, *Probabilistic circuits for cumulative distribution functions*, in Proceedings of the UAI Workshop on Tractable Probabilistic Modeling (TPM), 8 2024.

[11] O. BROADRICK, H. ZHANG, AND G. VAN DEN BROECK, *Polynomial semantics of tractable probabilistic circuits*, in Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence, N. Kiyavash and J. M. Mooij, eds., vol. 244 of Proceedings of Machine Learning Research, PMLR, 15–19 Jul 2024, pp. 418–429.

[12] F. P. BUSCH, M. WILLIG, J. SENG, K. KERSTING, AND D. S. DHAMI, *ψ net: Efficient causal modeling at scale*, in International Conference on Probabilistic Graphical Models, PMLR, 2024, pp. 452–469.

[13] K. CAI, X. LEI, J. WEI, AND X. XIAO, *Data synthesis via differentially private markov random fields*, Proceedings of the VLDB Endowment, 14 (2021), pp. 2190–2202.

[14] O. CAPPÉ, E. MOULINES, AND T. RYDÉN, *Inference in hidden Markov models*, Springer series in statistics, Springer, 2005.

[15] F. CARCILLO, A. DAL POZZOLO, Y.-A. LE BORGNE, O. CAELEN, Y. MAZZER, AND G. BONTEMPI, *Scarff: a scalable framework for streaming credit card fraud detection with spark*, Information fusion, 41 (2018), pp. 182–194.

[16] F. CARCILLO, Y.-A. LE BORGNE, O. CAELEN, AND G. BONTEMPI, *Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization*, International Journal of Data Science and Analytics, 5 (2018), pp. 285–300.

[17] F. CARCILLO, Y.-A. LE BORGNE, O. CAELEN, Y. KESSACI, F. OBLÉ, AND G. BONTEMPI, *Combining unsupervised and supervised learning in credit card fraud detection*, Information Sciences, 557 (2021), pp. 317–331.

[18] G. CASELLA AND E. I. GEORGE, *Explaining the gibbs sampler*, The American Statistician, 46 (1992), pp. 167–174.

[19] R. CASTELLON, A. GOPAL, B. BLONIARZ, AND D. ROSENBERG, *Dp-tbart: A transformer-based autoregressive model for differentially private tabular data generation*, arXiv preprint arXiv:2307.10430, (2023).

[20] M. CHAVIRA AND A. DARWICHE, *Compiling bayesian networks using variable elimination.*, in International Joint Conference on Artificial Intelligence, vol. 2443, Citeseer, 2007.

[21] A. CHOI AND A. DARWICHE, *On relaxing determinism in arithmetic circuits*, in Proceedings of the 34th International Conference on Machine Learning, D. Precup and Y. W. Teh, eds., vol. 70 of Proceedings of Machine Learning Research, PMLR, 06–11 Aug 2017, pp. 825–833.

[22] E. CHOI, S. BISWAL, B. MALIN, J. DUKE, W. F. STEWART, AND J. SUN, *Generating multi-label discrete patient records using generative adversarial networks*, in Machine learning for healthcare conference, PMLR, 2017, pp. 286–305.

[23] Y. CHOI, G. FARNADI, B. BABAKI, AND G. VAN DEN BROECK, *Learning fair naive bayes classifiers by discovering and eliminating discrimination patterns*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 10077–10084.

[24] Y. CHOI, A. VERGARI, AND G. VAN DEN BROECK, *Probabilistic circuits: A unifying framework for tractable probabilistic models*, UCLA. URL: http://starai. cs. ucla. edu/papers/ProbCirc20. pdf, (2020), p. 6.

[25] C. CHOW AND C. LIU, *Approximating discrete probability distributions with dependence trees*, IEEE Transactions on Information Theory, 14 (1968), pp. 462–467.

[26] G. CORMODE, C. PROCOPIUC, D. SRIVASTAVA, AND T. T. TRAN, *Differentially private summaries for sparse data*, in Proceedings of the 15th International Conference on Database Theory, 2012, pp. 299–311.

[27] P. CORRELL, A.-M. FEYER, P.-T. PHAN, B. DRAKE, W. JAMMAL, K. IRVINE, A. POWER, S. MUIR, S. FERDOUSI, S. MOUBARAK, ET AL., *Lumos: a statewide linkage programme in australia integrating general practice data to guide system redesign*, Integrated Healthcare Journal, 3 (2021), p. e000074.

[28] C. CULNANE, B. I. RUBINSTEIN, AND V. TEAGUE, *Health data in an open world*, arXiv preprint arXiv:1712.05627, (2017).

[29] J. CUSSENS, *Bayesian network learning with cutting planes*, in Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Uncertainty in AI 2011, Arlington, Virginia, USA, 2011, AUAI Press, p. 153–160.

[30] CZECH-TECHNICAL-UNIVERSITY(CTU), *Ctu-relational-dataset-repository*. `https://relational.fel.cvut.cz/`. Accessed: 2025-03-20.

[31] A. DAL POZZOLO, G. BORACCHI, O. CAELEN, C. ALIPPI, AND G. BONTEMPI, *Credit card fraud detection: A realistic modeling and a novel learning strategy*, IEEE Transactions on Neural Networks and Learning Systems, 29 (2018), pp. 3784–3797.

[32] A. DAL POZZOLO, O. CAELEN, Y.-A. LE BORGNE, S. WATERSCHOOT, AND G. BONTEMPI, *Learned lessons in credit card fraud detection from a practitioner perspective*, Expert Systems with Applications, 41 (2014), pp. 4915–4928.

[33] A. DANDEKAR, R. A. M. ZEN, AND S. BRESSAN, *Comparative evaluation of synthetic data generation methods*, in Proceedings of ACM Conference (Deep Learning Security Workshop), 2017.

[34] M. DANG, P. KHOSRAVI, Y. LIANG, A. VERGARI, AND G. VAN DEN BROECK, *Juice: A julia package for logic and probabilistic circuits*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 16020–16023.

[35] F. K. DANKAR AND M. IBRAHIM, *Fake it till you make it: Guidelines for effective synthetic data generation*, Applied Sciences, 11 (2021), p. 2158.

[36] A. DARWICHE, *Compiling knowledge into decomposable negation normal form*, in IJCAI, vol. 99, 1999, pp. 284–289.

[37] A. DARWICHE, *Decomposable negation normal form*, Journal of the ACM (JACM), 48 (2001), pp. 608–647.

[38] A. DARWICHE, *On the tractable counting of theory models and its application to truth maintenance and belief revision*, Journal of Applied Non-Classical Logics, 11 (2001), pp. 11–34.

[39] A. DARWICHE, *A differential approach to inference in bayesian networks*, Journal of the ACM, 50 (2003), p. 280–305.

[40] A. DARWICHE, *Modeling and reasoning with bayesian networks*, Cambridge University Press, 2009.

[41] A. DARWICHE, *Tractable boolean and arithmetic circuits*, arXiv preprint arXiv:2202.02942, (2022).

[42] A. DARWICHE AND P. MARQUIS, *A knowledge compilation map*, Journal of Artificial Intelligence Research, 17 (2002), pp. 229–264.

[43] R. DECHTER AND I. RISH, *Directional resolution: The davis-putnam procedure, revisited*, in Principles of Knowledge Representation and Reasoning, J. Doyle,

E. Sandewall, and P. Torasso, eds., The Morgan Kaufmann Series in Representation and Reasoning, Morgan Kaufmann, 1994, pp. 134–145.

[44] F. DELLAERT, M. KAESS, ET AL., *Factor graphs for robot perception*, Foundations and Trends® in Robotics, 6 (2017), pp. 1–139.

[45] R. DENG, D. CHEN, C. YAO, D. HU, Q. XIAN, AND S. ZHANG, *A gradient-penalized conditional timegan combined with multi-scale importance-aware network for fault diagnosis under imbalanced data*, Sensors, 25 (2025), p. 6825.

[46] C. DWORK, *Differential privacy: A survey of results*, in International conference on theory and applications of models of computation, Springer, 2008, pp. 1–19.

[47] C. DWORK, K. KENTHAPADI, F. MCSHERRY, I. MIRONOV, AND M. NAOR, *Our data, ourselves: Privacy via distributed noise generation*, in Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25, Springer, 2006, pp. 486–503.

[48] C. DWORK, F. MCSHERRY, K. NISSIM, AND A. SMITH, *Calibrating noise to sensitivity in private data analysis*, in Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3, Springer, 2006, pp. 265–284.

[49] C. DWORK, A. ROTH, ET AL., *The algorithmic foundations of differential privacy*, Foundations and Trends® in Theoretical Computer Science, 9 (2014), pp. 211–407.

[50] K. EMAM, *Seven ways to evaluate the utility of synthetic data*, IEEE Security & Privacy, 18 (2020), pp. 56–59.

[51] B. J. FREY, *Extending factor graphs so as to unify directed and undirected graphical models*, in Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence, UAI'03, San Francisco, CA, USA, 2002, Morgan Kaufmann Publishers Inc., p. 257–264.

[52] S. GAMBS, F. LADOUCEUR, A. LAURENT, AND A. ROY-GAUMOND, *Growing synthetic data through differentially-private vine copulas*, Proceedings on Privacy Enhancing Technologies, (2021).

[53] G. GARCIA-DONATO AND M. A. MARTINEZ-BENEITO, *On sampling strategies in bayesian variable selection problems with large model spaces*, Journal of the American Statistical Association, 108 (2013), pp. 340–352.

[54] A. GELMAN AND D. B. RUBIN, *Inference from iterative simulation using multiple sequences*, Statistical science, 7 (1992), pp. 457–472.

[55] Q. GENG, W. DING, R. GUO, AND S. KUMAR, *Privacy and utility tradeoff in approximate differential privacy*, arXiv preprint arXiv:1810.00877, (2018).

[56] A. GONCALVES, P. RAY, B. SOPER, J. STEVENS, L. COYLE, AND A. P. SALES, *Generation and evaluation of synthetic patient data*, BMC Medical Research Methodology, 20 (2020), p. 1–40.

[57] S. GOPI, Y. T. LEE, AND L. WUTSCHITZ, *Numerical composition of differential privacy*, Advances in Neural Information Processing Systems, 34 (2021), pp. 11631–11642.

[58] M. GUEYE, Y. ATTABI, AND M. DUMAS, *Row conditional-tgan for generating synthetic relational databases*, in ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2023, pp. 1–5.

[59] S. GUPTA AND A. SETHI, *Fedhypevae: Federated learning with hypernetwork generated conditional vaes for differentially private embedding sharing*, arXiv preprint arXiv:2601.00785, (2026).

[60] D. J. HAND, *Synthetic datasets for statistical disclosure control: Theory and implementation by jörg drechsler*, International Statistical Review, 80 (2012), pp. 483–483.

[61] W. HASTINGS, *Monte carlo sampling methods using markov chains and their applications*, Biometrika, 57 (1970), pp. 97–109.
Publisher: Oxford University Press.

[62] M. HITTMEIR, A. EKELHART, AND R. MAYER, *On the utility of synthetic data: An empirical evaluation on machine learning tasks*, in Proceedings of the 14th International Conference on Availability, Reliability and Security, 2019, pp. 1–6.

[63] HITTMEIR, A. EKELHART AND R. MAYER, *Utility and privacy assessments of synthetic data for regression tasks*, in 2019 IEEE International Conference on Big Data (Big Data), IEEE, 2019, pp. 5763–5772.

[64] Z. JI, Q. XIA, AND G. MENG, *A review of parameter learning methods in bayesian network*, in Advanced intelligent computing theories and applications, D.-S. Huang and K. Han, eds., Springer International Publishing, 2015, pp. 3–12.

[65] K. JIA, Y. MA, Y. LI, AND F. WANG, *Prada-gan: A private adaptive generative adversarial network with bayes network structure*, arXiv preprint arXiv:2511.07997, (2025).

[66] G. L. JONES AND J. P. HOBERT, *Sufficient burn-in for gibbs samplers for a hierarchical random effects model*, The Annals of Statistics, (2004).

[67] J. JORDON, J. YOON, AND M. VAN DER SCHAAR, *Pate-gan: Generating synthetic data with differential privacy guarantees*, in International conference on learning representations, 2018.

[68] KAGGLE, *Credit card fraud detection*.
`https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud`.
Accessed: 2025-03-09.

[69] P. KAIROUZ, S. OH, AND P. VISWANATH, *The composition theorem for differential privacy*, in International conference on machine learning, PMLR, 2015, pp. 1376–1385.

[70] A. KARANAM, S. MATHUR, S. SIDHEEKH, AND S. NATARAJAN, *A unified framework for human-allied learning of probabilistic circuits*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, 2025, pp. 17779–17787.

[71] R. KHARDON AND D. ROTH, *Learning to reason*, Journal of The Acm, 44 (1997), pp. 697–725.

[72] D. KOLLER AND N. FRIEDMAN, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*, The MIT Press, 2009.

[73] A. KOTAL AND A. JOSHI, *Differentially private synthetic data generation using context-aware gans*, in 2024 IEEE International Conference on Big Data (BigData), IEEE, 2024, pp. 6289–6297.

[74] F. KSCHISCHANG, B. FREY, AND H.-A. LOELIGER, *Factor graphs and the sum-product algorithm*, IEEE Transactions on Information Theory, 47 (2001), pp. 498–519.

[75] B. LEBICHOT, Y.-A. LE BORGNE, L. HE-GUELTON, F. OBLÉ, AND G. BONTEMPI, *Deep-learning domain adaptation techniques for credit cards fraud detection*, in Recent advances in big data and deep learning: proceedings of the INNS big data and deep learning conference INNSBDdL2019, held at sestri levante, genova, Italy 16-18 April 2019, Springer, 2020, pp. 78–88.

[76] J. LELAND AND Y. CHOI, *On the hardness of approximating distributions with probabilistic circuits*, in The 8th Workshop on Tractable Probabilistic Modeling (TPM), July 2025.

[77] H. LI, L. XIONG, AND X. JIANG, *Differentially private synthesization of multi-dimensional data using copula functions*, in Advances in database technology: proceedings. International conference on extending database technology, vol. 2014, NIH Public Access, 2014, p. 475.

[78] M. LI, Y. TIAN, J. ZHANG, D. FAN, AND D. ZHAO, *The trade-off between privacy and utility in local differential privacy*, in 2021 International Conference on Networking and Network Applications (NaNA), IEEE, 2021, pp. 373–378.

[79] N. LI, Z. ZHANG, AND T. WANG, *Dpsyn: Experiences in the nist differential privacy data synthesis challenges*, arXiv preprint arXiv:2106.12949, (2021).

[80] Z. LIN, A. JAIN, C. WANG, G. FANTI, AND V. SEKAR, *Using gans for sharing networked time series data: Challenges, initial promise, and open questions*, in Proceedings of the ACM internet measurement conference, 2020, pp. 464–483.

[81] Y. LINDELL, *Tutorials on the foundations of cryptography: Dedicated to oded goldreich*, Springer, 2017.

[82] A. LIU, K. AHMED, AND G. VAN DEN BROECK, *Scaling tractable probabilistic circuits: A systems perspective*, in Proceedings of the 41th International Conference on Machine Learning (ICML), 7 2024.

[83] A. LIU, S. MANDT, AND G. V. D. BROECK, *Lossless compression with probabilistic circuits*, arXiv preprint arXiv:2111.11632, (2021).

[84] A. LIU, H. ZHANG, AND G. VAN DEN BROECK, *Scaling up probabilistic circuits by latent variable distillation*, in Proceedings of the International Conference on Learning Representations (ICLR), 5 2023.

[85] F. LIU AND X. ZHAO, *Disclosure risk from homogeneity attack in differentially privately sanitized frequency distribution*, IEEE Transactions on Dependable and Secure Computing, 20 (2023), pp. 3927–3939.

[86] H. LOELIGER, *An introduction to factor graphs*, IEEE Signal Processing Magazine, 21 (2004), pp. 28–41.

[87] Y. LONG, B. WANG, Z. YANG, B. KAILKHURA, A. ZHANG, C. GUNTER, AND B. LI, *G-pate: Scalable differentially private data generator via private aggregation of teacher discriminators*, in Advances in Neural Information Processing Systems, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds., vol. 34, Curran Associates, Inc., 2021, pp. 2965–2977.

[88] A. MACHANAVAJJHALA, D. KIFER, J. GEHRKE, AND M. VENKITASUBRAMANIAM, *l-diversity: Privacy beyond k-anonymity*, Acm transactions on knowledge discovery from data (tkdd), 1 (2007), pp. 3–es.

[89] S. MAHIOU, K. XU, AND G. GANEV, *dpart: Differentially private autoregressive tabular, a general framework for synthetic data generation*, arXiv preprint arXiv:2207.05810, (2022).

[90] V. MANGINAS, N. MANGINAS, E. STEVINSON, S. VARGHESE, N. KATZOURIS, G. PALIOURAS, AND A. LOMUSCIO, *A scalable approach to probabilistic neuro-symbolic robustness verification*, in 19th International Conference on Neurosymbolic Learning and Reasoning, 2025.

[91] J. MANOJKUMAR PATIL, N. LEE, A. M. SAADAT CHOWDHURY, Y. CHOI, AND P. SHAKARIAN, *Probabilistic circuits for knowledge graph completion with reduced rule sets*, arXiv e-prints, (2025), pp. arXiv–2508.

[92] P. MARQUIS, *Knowledge compilation using theory prime implicates*, in Proceedings of the 14th international joint conference on artificial intelligence - volume 1, IJCAI'95, Morgan Kaufmann Publishers Inc., 1995, pp. 837–843.
Number of pages: 7 Place: Montreal, Quebec, Canada.

[93] R. MCKENNA, D. SHELDON, AND G. MIKLAU, *Graphical-model based estimation and inference for differential privacy*, in International Conference on Machine Learning, PMLR, 2019, pp. 4435–4444.

[94] F. MCSHERRY AND K. TALWAR, *Mechanism design via differential privacy*, in 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), IEEE, 2007, pp. 94–103.

[95] W. W. S. P. MINISTRY OF HEALTH NSW, UNIVERSITY OF TECHNOLOGY SYDNEY, *Patient journey modelling – from data to insights into patient pathways*.
`https://digitalhealthcrc.com/projects/patient-journey-modelling/`.
Accessed: 2025-03-04.

[96] I. MIRONOV, *Rényi differential privacy*, in 2017 IEEE 30th computer security foundations symposium (CSF), IEEE, 2017, pp. 263–275.

[97] S. MUKHERJEE, Y. XU, A. TRIVEDI, N. PATOWARY, AND J. L. FERRES, *privgan: Protecting gans from membership inference attacks at low cost to utility*, Proceedings on Privacy Enhancing Technologies, 2021 (2021), pp. 142–163.

[98] R. B. NELSEN, *An introduction to copulas*, Springer, 2006.

[99] T. A. H. NGUYEN, D. NGUYEN, T.-N. VO, T. D. LE, AND S. GUPTA, *Causal-aware generative adversarial networks with reinforcement learning*, arXiv preprint arXiv:2510.24046, (2025).

[100] NHS ENGLAND, *A&E Synthetic Data*.
`https://data.england.nhs.uk/dataset/a-e-synthetic-data`, 2020.
Accessed: 2020-11-12.

[101] B. NOWOK, *Utility of synthetic microdata generated using tree-based methods*, UNECE Statistical Data Confidentiality Work Session, (2015), pp. 1–11.

[102] NSWHEALTH, *Lumos dataset*.
`https://www.health.nsw.gov.au/lumos`.
Accessed: 2024-11-15.

[103] A. G. D. OF HEALTH AND A. CARE, *The australian health system*.
`https://www.health.gov.au/about-us/the-australian-health-system`.
Accessed: 2025-03-04.

[104] N. PAPERNOT, M. ABADI, Ú. ERLINGSSON, I. J. GOODFELLOW, AND K. TALWAR, *Semi-supervised knowledge transfer for deep learning from private training data*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017.

[105] N. PAPERNOT, S. SONG, I. MIRONOV, A. RAGHUNATHAN, K. TALWAR, AND U. ERLINGSSON, *Scalable private learning with PATE*, in International Conference on Learning Representations, 2018.

[106] N. PARK, M. MOHAMMADI, K. GORDE, S. JAJODIA, H. PARK, AND Y. KIM, *Data synthesis based on generative adversarial networks*, Proc. VLDB Endow., 11 (2018), p. 1071–1083.

[107] N. PATKI, R. WEDGE, AND K. VEERAMACHANENI, *The synthetic data vault*, in 2016 IEEE international conference on data science and advanced analytics (DSAA), IEEE, 2016, pp. 399–410.

[108] J. PEARL, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, The Morgan Kaufmann series in representation and reasoning, Morgan Kaufmann, rev. 2. ed., transferred to digital printing ed., 1988.

[109] D. PEEL AND G. MACLAHLAN, *Finite mixture models*, John & Sons, (2000).

[110] R. PEHARZ, S. LANG, A. VERGARI, K. STELZNER, A. MOLINA, M. TRAPP, G. VAN DEN BROECK, K. KERSTING, AND Z. GHAHRAMANI, *Einsum networks: Fast and scalable learning of tractable probabilistic circuits*, in International Conference on Machine Learning, PMLR, 2020, pp. 7563–7574.

[111] F. PETITJEAN, G. I. WEBB, AND A. E. NICHOLSON, *Scaling log-linear analysis to high-dimensional data*, in 2013 IEEE 13th International Conference on Data Mining, IEEE, 2013, pp. 597–606.

[112] H. PING, J. STOYANOVICH, AND B. HOWE, *Datasynthesizer: Privacy-preserving synthetic datasets*, in Proceedings of the 29th International Conference on Scientific and Statistical Database Management, 2017, pp. 1–5.

[113] H. POON AND P. DOMINGOS, *Sum-product networks: A new deep architecture*, in 2011 IEEE international conference on computer vision workshops (ICCV workshops), 2011, pp. 689–690.

[114] H. POONIA, M. WILLIG, Z. YU, M. ZECEVIC, K. KERSTING, AND D. S. DHAMI, *X-spn: Characteristic interventional sum-product networks for causal inference in hybrid domains*, in Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence, N. Kiyavash and J. M. Mooij, eds., vol. 244 of Proceedings of Machine Learning Research, PMLR, 15–19 Jul 2024, pp. 3004–3020.

[115] M. POWN, S. SURESH, AND B. LAKSHMI, *Investigation of homo and hetero-junction double-gate tunnel-fet-based adiabatic inverter circuits*, IETE journal of research, 69 (2023), pp. 1124–1132.

[116] A. D. POZZOLO, O. CAELEN, R. A. JOHNSON, AND G. BONTEMPI, *Calibrating probability with undersampling for unbalanced classification*, in 2015 IEEE Symposium Series on Computational Intelligence, 2015, pp. 159–166.

[117] G. M. RAAB, B. NOWOK, AND C. DIBBEN, *Guidelines for producing useful synthetic data*, arXiv preprint arXiv:1712.04078, (2017).

[118] L. RABINER, *A tutorial on hidden markov models and selected applications in speech recognition*, Proceedings of the IEEE, 77 (1989), pp. 257–286.

[119] A. RAJABI AND O. O. GARIBAY, *Tabfairgan: Fair tabular data generation with generative adversarial networks*, Machine Learning and Knowledge Extraction, 4 (2022), pp. 488–501.

[120] D. RANKIN, M. BLACK, R. BOND, J. WALLACE, M. MULVENNA, G. EPELDE, ET AL., *Reliability of supervised machine learning using synthetic data in health care: Model to preserve privacy for data sharing*, JMIR medical informatics, 8 (2020), p. e18910.

[121] G. REBANE AND J. PEARL, *The recovery of causal poly-trees from statistical data*, in Proceedings of the third conference on uncertainty in artificial intelligence, UAI'87, AUAI Press, 1987, pp. 222–228.

Number of pages: 7 Place: Seattle, WA.

[122] R. REITER AND J. KLEER, *Foundations of assumption-based truth maintenance systems: Preliminary report*, in AAAI Conference on Artificial Intelligence, 01 1987, pp. 183–189.

[123] J. G. RICHENS, C. M. LEE, AND S. JOHRI, *Improving the accuracy of medical diagnosis with causal machine learning*, Nature communications, 11 (2020), p. 3923.

[124] F. RISMANCHIAN, S. H. KASSANI, S. M. SHAVARANI, AND Y. H. LEE, *A data-driven approach to support the understanding and improvement of patients' journeys: a case study using electronic health records of an emergency department*, Value in health, 26 (2023), pp. 18–27.

[125] R. M. ROGERS, A. ROTH, J. ULLMAN, AND S. VADHAN, *Privacy odometers and filters: Pay-as-you-go composition*, Advances in Neural Information Processing Systems, 29 (2016).

[126] ROPELESS, *Compiledknowledge*.
`https://github.com/ropeless/compiled_knowledge`, 2025.

[127] ROPELESS, *Synthorus: Synthetic data entities*.
`https://github.com/ropeless/synthorus`, 2026.

[128] D. B. RUBIN, *Statistical disclosure limitation*, Journal of official Statistics, 9 (1993), pp. 461–468.

[129] F. J. SARMIN, A. R. RAHMAN, C. J. HENRY, AND N. MOHAMMED, *Privacy-preserving fair synthetic tabular data*, arXiv preprint arXiv:2503.02968, (2025).

[130] T. SATTAROV, M. SCHREYER, AND D. BORTH, *Federated diffusion modeling with differential privacy for tabular data synthesis*, in 2025 3rd International Conference on Federated Learning Technologies and Applications (FLTA), IEEE, 2025, pp. 286–293.

[131] M. SCANAGATTA, G. CORANI, C. P. DE CAMPOS, AND M. ZAFFALON, *Learning treewidth-bounded bayesian networks with thousands of variables*, in Advances in Neural Information Processing Systems, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds., vol. 29, Curran Associates, Inc., 2016.

[132] M. SCANAGATTA, C. P. DE CAMPOS, G. CORANI, AND M. ZAFFALON, *Learning bayesian networks with thousands of variables*, in Advances in Neural Information Processing Systems, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds., vol. 28, Curran Associates, Inc., 2015.

[133] R. SCHRAG, *Compilation for critically constrained knowledge bases.*, in Proceedings of the National Conference on Artificial Intelligence, vol. 1, 1996, pp. 510–515.

[134] N. R. SELVAM, G. VAN DEN BROECK, AND Y. CHOI, *Certifying fairness of probabilistic circuits*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 12278–12286.

[135] J. SENG, F. P. BUSCH, P. PRASAD, D. S. DHAMI, M. MUNDT, AND K. KERSTING, *Scaling probabilistic circuits via data partitioning*, in Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence, S. Chiappa and S. Magliacane, eds., vol. 286 of Proceedings of Machine Learning Research, PMLR, 21–25 Jul 2025, pp. 3701–3717.

[136] A. SHIH, D. SADIGH, AND S. ERMON, *Hyperspns: Compact and expressive probabilistic circuits*, Advances in neural information processing systems, 34 (2021), pp. 8571–8582.

[137] S. SIDHEEKH, K. KERSTING, AND S. NATARAJAN, *Probabilistic flow circuits: Towards unified deep models for tractable probabilistic inference*, in Uncertainty in Artificial Intelligence, PMLR, 2023, pp. 1964–1973.

[138] A. SRIVASTAVA, A. KUNDU, S. SURAL, AND A. MAJUMDAR, *Credit card fraud detection using hidden markov model*, IEEE Transactions on Dependable and Secure Computing, 5 (2008), pp. 37–48.

[139] T. STADLER, B. OPRISANU, AND C. TRONCOSO, *Synthetic data–anonymisation groundhog day*, in 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 1451–1468.

[140] L. E. SUCAR, *Probabilistic Graphical Models: Principles and Applications*, Advances in Computer Vision and Pattern Recognition, Springer International Publishing, 2021.

[141] S. SURESH AND B. DRAKE, *Sampling of large probabilistic graphical models using arithmetic circuits*, in AI 2024: Advances in Artificial Intelligence, Singapore, 2025, Springer Nature Singapore, pp. 174–187.

[142] S. SURESH, G. ZHANG, B. LIU, AND B. DRAKE, *Cost-efficient and privacy-preserving synthesis of complex sensitive data*, Proceedings of the 2025 Australasian Computer Science Week, (2025), pp. 30–39.

[143] M. J. SWAIN AND D. H. BALLARD, *Color indexing*, International journal of computer vision, 7 (1991), pp. 11–32.

[144] THE NUMPY DEVELOPERS, *NumPy v2.2 manual: numpy.einsum*, 2024. Accessed: 2026-02-02.

[145] THE OPEN DATA INSTITUTE, *Diagnosing the NHS: Synae*. `https://www.odileeds.org/events/synae/`, 6 2023. Accessed 2023-10-01.

[146] THE PYTORCH CONTRIBUTORS, *PyTorch 2.6 documentation: torch.einsum*, 2024. Accessed: 2026-02-02.

[147] THE TENSORFLOW DEVELOPERS, *TensorFlow v2.16.1 api: tf.einsum*, 2024. Accessed: 2026-02-02.

[148] P. TIWALD, I. KRCHOVA, A. SIDORENKO, M. V. VIEYRA, M. SCRIMINACI, AND M. PLATZER, *Tabularargn: A flexible and efficient auto-regressive framework for generating high-fidelity synthetic data*, arXiv preprint arXiv:2501.12012, (2025).

[149] R. TORKZADEHMAHANI, P. KAIROUZ, AND B. PATEN, *Dp-cgan: Differentially private synthetic data and label generation*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, 2019, pp. 0–0.

[150] P. K. TRIVEDI, D. M. ZIMMER, ET AL., *Copula modeling: an introduction for practitioners*, Foundations and Trends® in Econometrics, 1 (2007), pp. 1–111.

[151] G. TRUDA, *Generating tabular datasets under differential privacy*, arXiv preprint arXiv:2308.14784, (2023).

[152] A. TUCKER, Z. WANG, Y. ROTALINTI, AND P. MYLES, *Generating high-fidelity synthetic patient data for assessing machine learning healthcare software*, NPJ Digital Medicine, 3 (2020), p. 147.

[153] A. D. VAL, *Tractable databases: How to make propositional unit resolution complete through compilation*, in Principles of Knowledge Representation and Reasoning, J. Doyle, E. Sandewall, and P. Torasso, eds., The Morgan Kaufmann Series in Representation and Reasoning, Morgan Kaufmann, 1994, pp. 551–561.

[154] L. C. VAN DER GAAG, *Symbolic and Quantiative Approaches to Resoning with Uncertainty 12th European Conference, ECSQARU 2013, Utrecht, The Netherlands, July 8-10, 2013, Proceedings*, Lecture Notes in Artificial Intelligence ; 7958, Springer Berlin Heidelberg, 1st ed. 2013. ed., 2013.

[155] A. VASWANI, N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, Ł. KAISER, AND I. POLOSUKHIN, *Attention is all you need*, Advances in neural information processing systems, 30 (2017).

[156] F. VENTOLA, S. BRAUN, Z. YU, M. MUNDT, AND K. KERSTING, *Probabilistic circuits that know what they don't know*, in Uncertainty in Artificial Intelligence, PMLR, 2023, pp. 2157–2167.

[157] A. VERGARI, N. DI MAURO, AND F. ESPOSITO, *Visualizing and understanding sum-product networks*, Machine Learning, 108 (2019), pp. 551–573.

[158] B. WANG AND G. VAN DEN BROECK, *On the relationship between monotone and squared probabilistic circuits*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, 2025, pp. 21026–21034.

[159] H. WANG, Q. LIANG, J. T. HANCOCK, AND T. M. KHOSHGOFTAAR, *Enhancing credit card fraud detection through a novel ensemble feature selection technique*, in 2023 IEEE 24th International Conference on Information Reuse and Integration for Data Science (IRI), 2023, pp. 121–126.

[160] L. WASSERMAN AND S. ZHOU, *A statistical framework for differential privacy*, Journal of the American Statistical Association, 105 (2010), pp. 375–389.

[161] B. WEN, Y. CAO, F. YANG, K. SUBBALAKSHMI, AND R. CHANDRAMOULI, *Causal-TGAN: Modeling tabular data using causally-aware GAN*, in ICLR Workshop on Deep Generative Models for Highly Structured Data, 2022.

[162] A. WILKINS-CARUANA, M. BANDARA, K. MUSIAL, D. CATCHPOOLE, AND P. J. KENNEDY, *Inferring actual treatment pathways from patient records*, Journal of Biomedical Informatics, 148 (2023), p. 104554.

[163] L. XIE, K. LIN, S. WANG, F. WANG, AND J. ZHOU, *Differentially private generative adversarial network*, 2018.

[164] L. XU, M. SKOULARIDOU, A. CUESTA-INFANTE, AND K. VEERAMACHANENI, *Modeling tabular data using conditional gan*, Advances in neural information processing systems, 32 (2019).

[165] L. XU AND K. VEERAMACHANENI, *Synthesizing tabular data using generative adversarial networks*, arXiv preprint arXiv:1811.11264, (2018).

[166] J. YOON, L. N. DRUMRIGHT, AND M. VAN DER SCHAAR, *Anonymization through data synthesis using generative adversarial networks (ads-gan)*, IEEE journal of biomedical and health informatics, 24 (2020), pp. 2378–2388.

[167] J. YOON, D. JARRETT, AND M. VAN DER SCHAAR, *Time-series generative adversarial networks*, Advances in neural information processing systems, 32 (2019).

[168] Z. YU, D. S. DHAMI, AND K. KERSTING, *Sum-product-attention networks: Leveraging self-attention in probabilistic circuits*, arXiv preprint arXiv:2109.06587, (2021).

[169] Z. YU, M. TRAPP, AND K. KERSTING, *Characteristic circuits*, Advances in Neural Information Processing Systems, 36 (2023), pp. 34074–34086.

[170] H. ZHANG, M. DANG, B. WANG, S. ERMON, N. PENG, AND G. VAN DEN BROECK, *Scaling probabilistic circuits via monarch matrices*, in Proceedings of the 42th International Conference on Machine Learning (ICML), 7 2025.

[171] H. ZHANG, B. JUBA, AND G. VAN DEN BROECK, *Probabilistic generating circuits*, in International conference on machine learning, PMLR, 2021, pp. 12447–12457.

[172] H. ZHANG, B. WANG, M. ARENAS, AND G. VAN DEN BROECK, *Restructuring tractable probabilistic circuits*, in Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS), 5 2025.

[173] J. ZHANG, G. CORMODE, C. M. PROCOPIUC, D. SRIVASTAVA, AND X. XIAO, *Privbayes: Private data release via bayesian networks*, ACM Transactions on Database Systems (TODS), 42 (2017), pp. 1–41.

[174] W. ZHANG, J. ZHAO, F. WEI, AND Y. CHEN, *Differentially private high-dimensional data publication via markov network*, EAI Endorsed Transactions on Security and Safety, 6 (2019).

[175] Z. ZHANG, T. WANG, N. LI, J. HONORIO, M. BACKES, S. HE, J. CHEN, AND Y. ZHANG, {*PrivSyn*}*: Differentially private data synthesis*, in 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 929–946.

[176] Z. ZHAO, A. KUNAR, R. BIRKE, H. VAN DER SCHEER, AND L. Y. CHEN, *Ctab-gan+: Enhancing tabular data synthesis*, Frontiers in big Data, 6 (2024), p. 1296508.