UNIVERSITÄT ZU LÜBECK
INSTITUTE OF INFORMATION SYSTEMS

From the Institute of Information Systems
of the University of Lübeck
Director: Prof. Dr. rer. nat. Ralf Möller

New Methods for Efficient Query Answering in Gaussian Probabilistic Graphical Models

Dissertation
for Fulfilment of
Requirements
for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences

Submitted by

Mattis Hartwig
from Braunschweig

Lübeck 2021

First referee	Prof. Dr. rer. nat. Ralf Möller
Second referee	Prof. Dr. rer. nat. Maciej Liśkiewicz
Date of oral examination	September 6, 2022
Approved for printing.	

Abstract

The normal (or Gaussian) distribution might be the most famous and most used statistical distribution. Reasons for the wide applicability of the Gaussian distribution are twofold. First, we find natural phenomena that follow exactly or approximately the normal distribution. Second, representing the normal distribution can be done by using only two statistics - the mean and the variance - and calculations are relatively easy and fast, making it a suitable distribution for machine learning with good time performance when modeling continuous behavior. Probabilistic graphical models (PGMs) have their origin in discrete environments and are used to describe the dependence (or conditional independence) of random variables (randvars) in a graphical structure. Since many real-world examples require modelling with continuous randvars, we work with Gaussian distributed PGMs in this dissertation. The core functionality for PGMs is query answering, where we focus on answering conditional probability queries. In this dissertation, we contribute to solving the problem of *efficient query answering* algorithms in three different Gaussian PGM setups.

First, we work on query answering in Gaussian Bayesian networks that contain indistinguishable randvars. The aim is to work with representatives of the indistinguishable randvars instead of using all individual randvars (ground level) themselves which is known as lifting. The crucial point is allowing for individual observations of randvars. Randvars are only indistinguishable as long as there is no further evidence. Indistinguishable randvars are treated and modeled in the same way and have identical influence on other randvars in the model. As soon as the randvars for specific individuals are known, this additional knowledge should be usable in the graphical model. We contribute new algorithms for the construction of a lifted joint distribution and for lifted query answering. The runtime improvements of lifted algorithms are realized when the models contain very many randvars. Second, we look at dynamic versions of Gaussian Bayesian networks. The development along a time dimension introduces structure that can be exploited for more efficient query answering. We investigate the relationship between dynamic Gaussian Bayesian networks and Gaussian Processes. Third, we look into a more general structure where randvars can be Gaussian or discrete. These graphical models are called hybrid PGMs and can be converted into GMMs. We develop a new algorithm for approximate query answering that can be used when the number of mixture components gets very high.

Kurzfassung

Die Normalverteilung (oder Gaußverteilung) ist wohl die berühmteste und meistgenutzte statistische Verteilung. Es gibt zwei konkrete Gründe für die breite Nutzung der Gaußverteilung. Erstens gibt es natürliche Phänomene, die entweder exakt oder zumindest annährend gaußverteilt sind. Zweitens bedarf es nur zwei Parametern – den Mittelwert und die Varianz – um die Verteilung zu spezifizieren. Berechnungen mit der Gaußverteilung sind relativ einfach und schnell, was vor allem für Anwendungen im maschinellen Lernen wichtig ist – besonders, wenn kontinuierliche Variablen beteiligt sind. Probabilistische graphische Modelle (PGMs) wurden ursprünglich mit diskreten Zufallsvariable entwickelt und werden genutzt um die Beziehungen (oder die bedingten Unabhängigkeiten) zwischen Zufallsvariablen zu modellieren. Da aber viele Anwendungen aus der realen Welt die Modellierung von kontinuierliche Zufallsvariablen benötigen, arbeiten wir in dieser Dissertationsschrift mit normalverteilten PGMs. Die Kernfunktionalität eines PGMs ist es Anfragen an das Modell zu beantworten; hier haben wir den Fokus auf konditional Verteilungen, die sich durch die Nutzung von Beobachtungen ergeben. Die Lösung des Problems der effizienten Anfragebeantwortung ist nicht trivial aber in vielen Anwendungsfällen (gerade mit einer großen Anzahl an Zufallsvariablen) relevant. In dieser Dissertationsschrift tragen wir zur effizienten Anfragebeantwortung in drei verschiedenen Szenarien von normalverteilten PGMs mit neuen Algorithmen bei.

Als erstes arbeiten wir an der Beantwortung von Anfragen in normalverteilten Bayesschen Netzen, die ununterscheidbare Zufallsvariablen enthalten. Das Ziel ist es mit Repräsentanten der ununterscheidbare Zufallsvariablen zu rechnen, anstatt alle individuellen Zufallsvariablen zu nutzen. Die Nutzung von Repräsentanten, anstatt von Individuen ist auch unter dem Term *Lifting* bekannt. Der Kern ist, dass man immer noch erlaubt auf individueller Ebene Beobachtungen anzustellen. Die ununterscheidbaren Zufallsvariablen, die gleiches Verhalten und gleichen Einfluss auf andere Zufallsvariablen haben, sind eben nur ununterscheidbar solange keine Evidenz vorliegt. Wir präsentieren neue Algorithmen zur Konstruktion von einer gelifteten multivariaten Verteilung und zu gelifteten Beantwortung von Modellanfragen. Als zweites schauen wir auf dynamische Varianten von normalverteilten Bayesschen Netzen. Die gleichbleibende Entwicklung von Zufallsvariablen über die Zeit wird in der Beantwortung von Modellanfragen genutzt. Wir untersuchen in diesem Kontext den Zusammenhang zwischen dynamischen normalverteilten Bayesschen Netzen und Gaußprozessen. Als drittes arbeiten wir mit einer etwas generelleren Struktur, sogenannte hybride Bayesschen Netzen, die sowohl normalverteilte als auch diskrete Zufallsvariablen enthalten. Diese Modelle können auch in

Gaußsche Mischverteilungen, die aus mehreren Komponente bestehen umgewandelt werden. Wir präsentieren einen neuen Algorithmus für approximative Anfragebeantwortung, der bei Mischverteilungen mit einer hohen Anzahl an Komponenten und Dimensionen zu einer Zeitersparnis führt.

Acknowledgements

I would not have thought that writing the acknowledgements would be a difficult task for me. The acknowledgments section is about thanking other people for their contribution and I feel thankful a lot for many different things in my life that had a huge impact on me writing this dissertation. But I guess this is the problem. It is difficult to decide where to start and where to end. I am thankful for support in different areas that a are connected to writing a dissertation and I am thankful for the support of different people. So we have a two-dimensional setup and after countless hours of working in two dimensional matrices, my first intuition was to sketch out a matrix with two thankfulness dimensions: people and topics. Then again, showing a table in the acknowledgements might also be a bit too nerdy and might not capture my emotions of deeply feeling thankful. As some of you know, when iterating through a matrix we can

(a) put the people dimension on the outer loop and thank everyone for all the support they have given, which might lead to a lot of redundancy in the topics, or

(b) put the topic dimension on the outer loop and in every topic thank all people involved, which might feel a bit scattered if the same persons are mentioned at multiple places, or

(c) use some kind of diagonal approach (e.g. inspired by Cantor's diagonal argument), which would confuse everyone here.

Since I do not need to proof completeness of my acknowledgements, I introduce an heuristic where I only select the primary topic for each person or group and then go topic by topic as indicated in b).

Content & discussions: First of all, I am very thankful to have worked on very interesting topics that have been a real intellectual challenge. The main person to thank here is my supervisor, Prof. Ralf Möller. Ralf thank you for giving me, as someone who only had a basic background in the theoretical realm of computer science, the chance to work together with you. I enjoyed diving very deep into the theoretical problems of query answering and having discussions with you on all topics around artificial intelligence. You made me think sharper and more structured on syntax, semantics and algorithms. I really look forward to continuing working with you on many more interesting topics and research questions. I also thank all colleagues at the Institute of Information Systems for the coworking and discussions. Special thanks here to Tanya, who worked with me on two papers and read the first version of this dissertation. Having smart and motivated people around you makes you do better research and of course makes working more fun.

Setup: During the two and a half year I was an external doctoral student which gave me the freedom to pursue also a lot of personal topics that where important to me. Two organisations made this possible. McKinsey & Company as my employer supported the first year of the dissertation. In general, working for McKinsey has taught me incredibly much. I am thankful for all the colleagues who have worked with me countless hours during day and night. Many of you have become friends and since it is clear now that I will not go back to McKinsey after the dissertation, I am even more thankful to be part of that special group of people. The second organization is the "Studienstiftung des deutschen Volkes", a German scholarship organisation, who have supported me in the second half of the journey. It is awesome to be part of that network, to meet all these different people with different backgrounds but the common understanding of working on topics relevant for the society. Also thanks to Prof. Achim Peters. You were assigned as my responsible professor and we developed an unexpected but very valuable relationship.

Emotional support: In an endeavour like writing this dissertation, I had also emotional ups and downs. Especially when I started into my first year, I had no clue where to start and it was (and still sometimes is) very easy to feel overwhelmed. Thanks to all friends and colleagues for having an open ear and for sharing each others feelings from time to time. Also thanks to all the friends from climbing, surfing and gymnastics. Sport has been an important thing my whole life and value all the fun and exhausting moments we have together. Special thanks go to you Jen. You have stayed with me over many years through all kinds of fun and difficult moments. When I get stressed because something does not go as intended you ease my mind and when we need to relax we go on exciting holidays with our "Volkswagen Bulli". I look forward to our future.

Where it started: Last but not least, I would like to thank my mother and father who have taught me to switch perspectives and my brothers who have, despite the age difference, engaged with me deeply and challenged me early. Also thanks to Dietmar Scholz. You were the first teacher who understood my interest in math and then supported me exceptionally through my school journey.

What comes next: Going forward, I will stay in research but will also go back to take the position as a managing director in my company singularIT. Which brings me to the last topic. Thank you Felix for being the best friend I could wish for and for taking me back into our jointly founded company. I look forward to working with you again.

Thank you!

Mattis Hartwig

Contents

Abstract iii

Kurzfassung v

Acknowledgements vii

List of Algorithms xiii

List of Figures xiv

List of Abbreviations xv

List of Symbols xvii

1 Introduction 1
- 1.1 The Problem of Query Answering 1
- 1.2 Contributions 3
- 1.3 Structure 3

2 Overarching Technical Preliminaries 7
- 2.1 Fundamentals of Probability Distributions 7
- 2.2 Probabilistic Graphical Models 8
- 2.3 The (Multivariate) Gaussian Distribution 9
- 2.4 Gaussian Probabilistic Graphical Networks 10

3 Lifting Gaussian Bayesian Networks 15
- 3.1 Lifting Preliminaries 16
  - 3.1.1 Parameterized randvars 16
  - 3.1.2 Parameterized Gaussian Bayesian Networks 17
- 3.2 Introducing a Running Example 18
- 3.3 Constructing the Lifted Joint Distribution 21
  - 3.3.1 The Base Case Without Overlaps - Non-matrix Notation 21
  - 3.3.2 The Base Case Without Overlaps - Matrix Notation 24
  - 3.3.3 Allowing for Overlaps 29
- 3.4 Operators for the Lifted Joint Distribution 38
  - 3.4.1 Addition and Subtraction 38

3.4.2 Multiplication 38
3.4.3 Lifted Matrix Inversion 40
3.5 Lifted Query Answering 43
3.5.1 Lifted Answering of a Marginal Query 43
3.5.2 Lifted Answering of a Conditional Query 44
3.5.3 Grouping the Evidence 45
3.5.4 Example Queries and Calculations 47
3.6 Complexity Analysis 49
3.6.1 Complexity for Constructing the Joint Distribution 49
3.6.2 Complexity for Conditional Query Answering 50
3.7 Empirical results 51
3.7.1 Experimental Constructions 51
3.7.2 Experimental Query Answering 52
3.8 Intermediate Discussion 52

4 Gaussian Probabilistic Graphical Models and Gaussian Processes 55
4.1 Gaussian Process Specific Preliminaries 56
4.1.1 Dynamic Gaussian Bayesian Networks 56
4.1.2 Gaussian Processes 58
4.1.3 Kernel Functions 60
4.1.4 Multi-output Kernels 62
4.1.5 Comparing Probabilistic Graphical Models to Gaussian Processes 65
4.2 Related Work on the Relationship Between PGMs and GPs 65
4.3 Gaussian Processes for Gaussian Markov Chains 66
4.3.1 Constructing the Kernel 66
4.3.2 Kernel Validity 69
4.3.3 Defining the Gaussian Process 69
4.4 Gaussian Processes for Hidden Markov Models 71
4.4.1 Constructing the Kernel 71
4.4.2 Kernel Validity 71
4.4.3 Defining the Gaussian Process 72
4.5 Gaussian Processes for Dynamic Gaussian Bayesian Networks 72
4.5.1 Constructing the Kernel 73
4.5.2 Continuity Discussion 75
4.5.3 Kernel Validity 75
4.5.4 Defining the Gaussian Process 75
4.6 Intermediate Discussion 76

5 Approximate Query Answering in Complex Gaussian Mixture Models 79
5.1 GMM Preliminaries 80
5.1.1 Gaussian Mixture Models 80

5.1.2 Motivating High-Complexity Mixtures 81
5.1.3 Query Answering 81
5.2 Approximation Approach 82
5.2.1 Detailed Algorithm 83
5.2.2 Error Bound 85
5.3 Complexity Analysis 86
5.4 Experimental Evaluation 87
5.4.1 Evaluation 1: Top-L Selection 87
5.4.2 Evaluation 2: Time Savings 87
5.4.3 Evaluation 3: Used Components 88
5.5 Intermediate Discussion 89

6 Final Discussion 91
6.1 Contributions 91
6.2 Conclusion 92
6.3 Future Research 93

A Further Lifting Examples 95
A.1 Ground Level Covariance Construction 95
A.2 Lifted Covariance Construction - Matrix Notation 100
A.3 Full Lifted Joint Covariance 102

Bibliography 105

Publications 111

Curriculum Vitae 113

List of Algorithms

1 Converting a GBN to a multivariate Gaussian joint distribution (matrix notation) 13

2 Lifted recursive block matrix inversion 42

3 Recursive select top-L elements function 83

List of Figures

2.1 Example (G)BN with four randvars 12

3.1 Running Example 18
3.2 Grounding parts of the running example 20
3.3 Evaluation of experiments 53

4.1 Three examples for different types of DGBNs: a) Gaussian Markov chain b) Gaussian hidden Markov model c) two-timeslice DGBN 57
4.2 A DGBN represented in two-timeslice notation with a transition matrix 59
4.3 A GP with 3 evidence points at timepoints 2, 6, and 8, where a) contains a sparse query set ($\boldsymbol{t}_Q = 1, 3, 4, 5, 7, 9, 10$) of the posterior and b) a relatively continuous one (1000 steps in the interval between 0 and 10) (Roberts *et al.*, 2013) 61
4.4 Illustration of the influence of kernels on the posterior distribution generated by a GP with one point of evidence at $t_E = 1$. The red area corresponds to the confidence interval of two standard deviations and the read line corresponds to the most probable assignment (the means at each timepoint in the posterior). 63
4.5 Illustration of multi-output covariance matrices for two example timesteps 64
4.6 Illustration of the constructed GP for one-dimensional Gaussian Markov chains 70
4.7 Combination of a periodic kernel with the one-dimensional Markov chain kernel from Expression (4.32) 76

5.1 Runtime [s] (y-axis) for different list sizes [#] (x-axis) 88

List of Abbreviations

randvar	random variable
tdvar	time dependent random variable
PGM	Probabilistic graphical model
BN	Bayesian network
GBN	Gaussian Bayesian network
DGBN	Dynamic Gaussian Bayesian network
GMM	Gaussian mixture model
logvar	Logical variable
PRV	Parameterized random variable
GP	Gaussian Process
MPE	Most probable explanation

List of Symbols

X	Randvars or vertices in a PGM
$\boldsymbol{X}$	Set of randvars
$\mathcal{X}$	Full set of all randvars
N	Number of randvars
$R(X)$	Range of a randvar
$X_i = x_i$	Event
G	Graph
$\boldsymbol{\xi}$	Edges in a graph
$\boldsymbol{Pa}(X)$	Set of parents of a randvar
$\boldsymbol{E} = \boldsymbol{e}$	Evidence set including the evidence values
$\boldsymbol{Q}$	Query set
$\beta_{i,j}$	Linear dependency
$\boldsymbol{T}$	Linear dependency matrix
$\boldsymbol{\Sigma}$	Covariance matrix
μ	Marginal mean, node mean
σ^2	Variance, node variance
δ	Kronecker delta function
L	Logvar
l	Logvar constant
$\boldsymbol{L}$	Sequence of logvars
$\mathcal{L}$	Full sequence of all logvars
$Y = X(\mathbf{L})$	PRV
$\boldsymbol{Y}$	Set of PRVs
$\mathcal{Y}$	Full set of all PRVs
M	Number of PRVs
Λ	Number of logvars
S	Number of longest logvar sequence
$\boldsymbol{\eta}$	Lifted mean vector
$\boldsymbol{\rho}$	Lifted covariance matrix/tensor
λ	Lifted node variance
ζ	Lifted linear relations
$\boldsymbol{\tau}$	Cardinality vector for logvar domain sizes
$\boldsymbol{D}(\boldsymbol{L})$	Domain of a logvar sequence

$lv(Y)$	Involved logvars
$lif(X)$	PRV of a instantiated randvar
$\boldsymbol{gr}(Y)$	Grounding
$\mathbf{I}$	Identity matrix
$\mathbf{J}$	All-ones matrix
$\boldsymbol{\Phi}$	Permutation set for Kronecker sequences
$\boldsymbol{q}$	Index vector for a Kronecker sequence
ν	Mean summand in posterior mean calculation
$m(t)$	Mean function in a GP
$k(t, t')$	Kernel
t	Time
$\tilde{T}$	Number of time-steps
D	Number of tdvars as output dimensions in a GP
$\mathbb{T}$	Feature space in a GP
$\mathbf{M}$	Transition matrix in a DGBN
$\mathbf{A}$	Variances of all tdvars in a diagonal matrix
K	Number of mixture components
$w(k)$	Weight of mixture component
$\boldsymbol{\theta}_k$	Parameters of mixture component

Chapter 1

Introduction

The normal (or Gaussian) distribution might be the most famous and most used statistical distribution (Forbes *et al.*, 2011). The normal distribution was first described by Abraham de Moivre (1738), who had seen it as an approximation for the binomial distribution. Carl Friedrich Gauss (1857) refined the understanding of the normal distribution as a statistical distribution.

Reasons for its wide applicability are twofold. First, we find natural phenomena that follow exactly the normal distribution like the position of a particle subject to diffusion (Maxwell, 1860) or that follow approximately the normal distribution, e.g., the height distribution of women (Schilling *et al.*, 2002). Second, representing the normal distribution can be done by using only two statistics - the mean and the variance - and calculations are relatively easy and fast, making it a suitable distribution for machine learning with good time performance when modeling continuous behavior.

Probabilistic graphical models (PGMs) have their origin in discrete environments and are used to describe the dependence (or conditional independence) of random variables (randvars) in a graphical structure (Koller and Friedman, 2009). One well-known example of a PGM in action is the application called Pathfinder, which is assisting practitioners with diagnosis of lymph-node diseases (Heckerman *et al.*, 1992). Unfortunately, performing inference in PGMs can be computationally costly, and thus efficient algorithms for inference in discrete models have been an active research area over the last decades (Madsen *et al.*, 2005; Frey and Jojic, 2005; Mooij, 2010; Ankan and Panda, 2015; Shih and Ermon, 2020). Many real-world examples require modelling with continuous randvars. Consequently, Gaussian PGMs and their related model types experience an increase in attention from the machine learning community. Developing new approaches for efficient query answering in Gaussian PGMs with focus on Gaussian Bayesian networks (GBNs) and describing the connection to related models like Gaussian mixture models (GMMs) as well as Gaussian processes (GPs) is the core of this dissertation.

1.1 The Problem of Query Answering

Query answering in general is the basic problem defined on every model. Query answering can be interpreted as asking a question (query) to the model and expecting a response (query answer). In PGMs, queries work with probability distributions. Focus of this dissertation is the conditional probability query that contains evidence about randvars and expects the conditional distribution of the queried randvars as an answer (see Section 2.2). Efficiency in query answering is referring to answering a query with as little computational resources as possible in terms of time and space. For Gaussian PGMs and multivariate Gaussians, the query answering algorithm has a cubic complexity with respect to the number of randvars in the graph because of matrix inversion and multiplication (Liu *et al.*, 2020). Cubic complexity for query answering is already an improvement compared to query answering in discrete PGMs. The problem of exact conditional probability query answering in discrete PGMs is $\mathcal{NP}$-hard (Koller and Friedman, 2009). High runtime complexities result in high runtimes of query answering algorithms when many randvars are involved in the model. When the algorithms are used in real-world applications, high runtimes can result in failure of the desired task because the environment changes faster than the query answer is produced.

Researchers have been working on finding efficient algorithms for the query answering problem in PGMs for many years (Lauritzen and Spiegelhalter, 1988; Kschischang *et al.*, 2001; Braun and Möller, 2016). In general, special structure being present either in the queries or in the models is used to speed up exact query answering or clever heuristics are used for approximation. In this dissertation, we look into three different kinds of special structures. First, we work on query answering in Gaussian PGMs that contain indistinguishable randvars. The aim is to work with representatives of the indistinguishable randvars instead of using all individual randvars (ground level) themselves which is known as lifting (Poole, 2003). The crucial point is that we still want to be able to make individual observations of randvars. Randvars are only indistinguishable as long as there is no further evidence. An example (that we will detail out in Section 3.2) is that the health status of multiple patients is modeled. As long as there is no further evidence about individual patients, they are treated and modeled in the same way and have identical influence on other randvars in the model. As soon as the health status for individuals is known, this knowledge should be usable in the graph. The runtime improvements of lifted algorithms are realized when the models contain very many randvars. Second, we look at dynamic versions of Gaussian PGMs. The development of randvars along a time dimension introduces structure that can be exploited for more efficient query answering. We investigate the relationship between dynamic Gaussian Bayesian networks (DGBNs) and GPs. Third, we look into a more general structure where randvars can be Gaussian or discrete. These graphical models are called hybrid PGMs and can be converted into GMMs. One interpretation for hybrid or mixture models is a combination of multiple expert models into one combined model (ensemble). We develop a new algorithm for approximate query answering that can be used when the number of mixture components gets very high.

1.2 Contributions

This dissertation contains a number of contributions to inference in Gaussian PGMs. The contributions are summarised as follows.

(1) Lifting of joint representations A Gaussian PGM can be converted into a multivariate Gaussian distribution. With this contribution, we develop a lifted representation for the multivariate Gaussian distribution and introduce an approach to construct the distribution from a GBN that contains indistinguishable randvars.

(2) Lifted operations lifted query answering When working with a joint distribution, operations for adding, multiplying, and inverting elements of the covariance matrix are needed. With this contribution, we develop the lifted version of the algebraic operations needed and avoid grounding by using only the lifted joint representation, thus decreasing the runtime complexity of the operations.

(3) Lifted query answering We develop two approaches for lifted query answering, significantly improving the runtime complexity for answering queries compared to query answering on ground level. We perform a theoretical complexity analysis and a experimental evaluation of the developed algorithms.

(4) GP representations To combine PGMs with GPs, we develop GP representations, consisting of a mean and a kernel function, for three different types of DGBNs: One-dimensional Gaussian Markov chains, Gaussian hidden Markov models, and two-timeslice DGBNs, resulting in more flexible kernel representations of the time transition properties.

(5) Approximate query answering in GMMs We contribute an approximate query answering algorithm for highly complex GMMs, i.e., GMMs with a high number of dimensions and components. We perform a theoretical complexity analysis and a experimental evaluation of the developed algorithms, showing a significant speed-up.

1.3 Structure

After this introduction, we start with a chapter on preliminaries for probability distributions in general and more specifically PGMs and their Gaussian based counterparts. Following the preliminaries, the dissertation is divided into three chapters covering different topics related to efficient query answering.

- Chapter 3 presents lifting for GBNs (Contributions **1**, **2**, and **3**).
  - Section 3.1 presents lifting specific preliminaries.

  - Section 3.2 introduces a running example for a lifted/parameterized GBN that is used throughout the chapter to illustrate the approaches and operations developed.
  - Section 3.3 presents algorithms for constructing a lifted representation of the joint distribution given a lifted/parameterized GBN.
  - Section 3.4 presents lifted implementations of operations needed for working with the lifted joint distribution.
  - Section 3.5 presents algorithms for lifted query answering using a lifted joint distribution.
  - Section 3.6 presents complexity analyses for constructing the lifted joint distribution and for lifted query answering.
  - Section 3.7 presents an experimental evaluation of the algorithms developed for constructing the lifted joint distribution and lifted query answering.
  - Section 3.8 contains a discussion of the results in lifting GBNs and potential next steps.

  The third chapter is based on the following publications:

  Mattis Hartwig and Ralf Möller. Lifted Query Answering in Gaussian Bayesian Networks. In *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 233–244. PMLR, 2020

  Mattis Hartwig, Tanya Braun, and Ralf Möller. Handling Overlaps When Lifting Gaussian Bayesian Networks. In *IJCAI-21 Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 4228–4234. IJCAI Organization, 2021

  Mattis Hartwig, Ralf Möller, and Tanya Braun. An Extended View on Lifting Gaussian Bayesian Networks. Submitted to Elsevier Artificial Intelligence Journal

- Chapter 4 contains the work on connecting DGBNs to GP for time series modelling (Contribution **4**).
  - Section 4.1 presents the GP specific preliminaries, covering GPs, kernel functions and multi-output settings.
  - Section 4.3 presents a GP for a one-dimensional Gaussian Markov chain.
  - Section 4.4 presents a GP for a Gaussian hidden Markov model.
  - Section 4.5 presents a GP for a multi-dimensional two-timeslice DGBN.

  - Section 4.6 discusses the results in encoding specific DGBNs as GPs for time series modelling.

  The fourth chapter is based on the following publications:

  > Mattis Hartwig, Marisa Mohr, and Ralf Möller. Constructing Gaussian Processes for Probabilistic Graphical Models. In *FLAIRS-20 Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference.* AAAI Press, 2020
  >
  > Mattis Hartwig and Ralf Möller. How to Encode Dynamic Gaussian Bayesian Networks as Gaussian Processes? In *AJCAI-20 Proceedings of the Australasian Joint Conference on Artificial Intelligence*, pages 371–382. Springer, 2020

- Chapter 5 presents an approximate query answering algorithm for GMMs (Contribution **5**).
  - Section 5.1 presents the preliminaries for GMMs.
  - Section 5.2 presents the developed approximation approach.
  - Section 5.3 presents the complexity analysis for the approximation algorithm.
  - Section 5.3 presents the experimental results the approximation algorithm.
  - Section 5.5 discusses the the results of this chapter.

  The fifth chapter is based on the following publication.

  > Mattis Hartwig, Marcel Gehrke, and Ralf Möller. Approximate Query Answering in Complex Gaussian Mixture Models. In *ICBK-19 Proceedings of the 2019 IEEE International Conference on Big Knowledge*, pages 81–86. IEEE, 2019

The dissertation concludes with Chapter 6, where we discuss the contributions and give an outlook to further research.

Chapter 2

Overarching Technical Preliminaries

This chapter contains the basic definitions and concepts for PGMs and Gaussian distributions, which will be used in all following chapters. Preliminaries that are only used within one chapter will be introduced in that respective chapter. Throughout this dissertation, we use bold symbols for vectors, sets, sequences, and matrices, and thin symbols for scalars or individual elements. For sets where we want to emphasize that this is the super set from which following subsets are drawn, we use the calligraphic font (e.g., $\mathcal{X}$).

2.1 Fundamentals of Probability Distributions

The syntax for working with probability distribution is as follows. Given a set of randvars $\mathcal{X} = \{X_1, ..., X_N\}$, the joint probability distribution over the randvars is denoted as $P(\mathcal{X}) = P(X_1, \dots, X_n)$. The values a randvar X can take are given by its range $R(X)$. If a particular randvar is given a value, we call it an event $X_i = x_i$.

Given a joint probability distribution $P(\mathcal{X})$, there are two main semantic constructs that are interesting. First, the marginal distribution, which contains the probability distribution over a subset of possible events for a subset of random variables. The marginal distribution is denoted as $P(X_i = x_i)$, where $X_i \in \mathcal{X}$ and $x_i \in R(X_i)$. The second is the conditional probability distribution, which contains the probability of a set of random variables given a set of events. The conditional probability is denoted as $P(X_i = x_i | X_j = x_j)$ where X_j denote the variables for which we have evidence. The marginal distribution can be calculated by summing up all probabilities that lead to the specific event

$$P(X_i = x_i) = \sum_{\boldsymbol{x} \in R(\boldsymbol{X} \in \mathcal{X} \setminus X_i)} P(X_i = x_i, \boldsymbol{X} = \boldsymbol{x}). \tag{2.1}$$

The summing process is also referred to as marginalization. For continuous ranges, the randvars in $\boldsymbol{X}$ can take infinitely many values, resulting in an integral over the range.

The conditional probability distribution can be calculated by using two marginal distributions:

$$P(X_i = x_i | X_j = x_j) = \frac{P(X_i = x_i \wedge X_j = x_j)}{P(X_j = x_j)} \tag{2.2}$$

In probability theory one well-known rule is the Bayes rule that follows from Expression (2.2):

$$P(X_i = x_i | X_j = x_j) = \frac{P(X_j = x_j | X_i = x_i) P(X_i)}{P(X_j)} \tag{2.3}$$

.

2.2 Probabilistic Graphical Models

The fundamental structure for PGMs are graphs. We follow the definitions from Koller and Friedman (2009).

Definition 2.2.1 (Graph, path)**.** A *graph G* consists of a set of vertices $\mathcal{X} = \{X_1, ..., X_N\}$ and a set of edges $\boldsymbol{\xi}$. Edges can be directed between two vertices $X_i \rightarrow X_j$ or undirected X_i —X_j. We say that $X_1, ..., X_k$ form a path in graph $G(\mathcal{X}, \boldsymbol{\xi})$ if, for every $i = 1, ..., k$ there is either a directed edge $X_i \rightarrow X_{i+1}$ or undirected edge X_i —X_{i+1}. A path is directed if at least one edge along the path is directed.

In general, graphs can also contain directed and undirected edges and thus be mixed graphs. In this dissertation we focus on directed graphs.

Definition 2.2.2 (Cycle, directed acyclic graph, parents)**.** A *cycle* is a directed path $X_1, ..., X_k$ where $X_1 = X_k$. A graph $G(\mathcal{X}, \boldsymbol{\xi})$ is called a *directed acyclic graph* if all edges in $\boldsymbol{\xi}$ are directed edges and if the graph contains no *cycles*. The set of *parents* $\boldsymbol{Pa}(X_i)$ of a randvar X_i is defined as all $\{X_k | X_k \rightarrow X_i \in \boldsymbol{\xi}$.

PGMs model the stochastic behavior of randvars. In general, PGMs serve as a compact representation of the probabilistic joint distribution of the randvars and allow for a factorisation of the joint distribution. There are multiple types of PGMs, e.g. Markov random fields, Bayesian networks (BNs) and factor graphs Koller and Friedman (2009). In this dissertation we focus on Bayesian networks.

Definition 2.2.3 (Factorization, Bayesian network)**.** Let G be a directed acyclic graph $G(X, E)$ whose vertices represent randvars $\mathcal{X} = \{X_1, ..., X_N\}$. A distribution P over the same space *factorizes* according to G if P can be expressed as a product

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i | \boldsymbol{Pa}(X_i)). \tag{2.4}$$

A *Bayesian network* is a pair (G, P) of the graph G and a set of conditional probability tables $P(X_i | \boldsymbol{Pa}(X_i))$ associated with G's nodes, where P factorizes over G.

Since vertices in a BN represent randvars, we use the same symbol X for vertices and randvars. The BN structure encodes conditional independencies between randvars. Any randvar X_i is conditionally independent of all non-descending randvars given its parents $\boldsymbol{Pa}(X_i)$. For iterating over a set of randvars that are part of a BN, we use a topological ordering of the randvars.

Definition 2.2.4 (Topological ordering)**.** Let $\mathcal{X}$ be a set of randvars within a BN. A topological ordering of the randvars in $\mathcal{X}$ is any ordering that ensures that a child node X_i comes always after its parent node X_k. In a BN, there is always at least one topological ordering possible.

In general, BNs are used to represent join probability distributions in a sparse way (by using the factorization implied by the network structure). Learning or choosing a networks structure comes with trade-offs. A very simple structure that allows for relatively quick query answering might not capture all characteristics of the true joint probability distribution. On the other hand choosing a simple structure might be beneficial if the underlying joint probability distribution contains many conditional independencies.

Having introduced a representation for a joint probability distribution, we now define how to use the distribution with query answering. In general, there are different types of queries. When we refer to queries in this dissertation, we mean conditional probability queries (marginal queries being a special type of conditional probability queries). One other example for a query type is the most probable explanation (MPE) query, also known as the maximum a posterior (MAP) query.

Definition 2.2.5 (Query)**.** A *query* $P(\boldsymbol{Q}|\boldsymbol{E} = \boldsymbol{e})$ consists of a query randvar set $\boldsymbol{Q} \subseteq \mathcal{X}$, and a set of events $\boldsymbol{E} = \boldsymbol{e}$ with $\boldsymbol{E} \subseteq \mathcal{X}$.

We use the special term *marginal query* for queries that do not contain any evidence, i.e., $\boldsymbol{E} = \varnothing$ and the term conditional query for queries that contain evidence.

2.3 The (Multivariate) Gaussian Distribution

A single Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ is defined by its mean μ and its variance σ^2. The distribution of a single Gaussian randvar X_i is described with the probability density function

$$f_{X_i,\mu,\sigma}(x_i) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} \exp\left(-\frac{1}{2\sigma^2}(x_i - \mu)^2\right). \tag{2.5}$$

In short, we write

$$X_i \sim \mathcal{N}\left(\mu, \sigma^2\right). \tag{2.6}$$

For continuous variables probability is only defined for a range of values. The probability that a random variable X_i takes a value of less than x_i is defined as

$$P(X_i \leq x_i) = \int_{-\infty}^{x_i} f_{X_i,\mu,\sigma}(x_i)dx \tag{2.7}$$

With Expression (2.7) we can calculate the probability over any interval for X_i.

An N-dimensional multivariate Gaussian distribution over values $\boldsymbol{x}$ of a set of N randvars $\boldsymbol{X} = \{X_1, ..., X_N\}$ is defined by an N-dimensional mean vector $\boldsymbol{\mu}$ and an $N \times N$-dimensional symmetric covariance matrix $\boldsymbol{\Sigma}$. The probability density function is given by

$$f_{\boldsymbol{X},\boldsymbol{\mu},\boldsymbol{\Sigma}}(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{N}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right). \tag{2.8}$$

In short, we write again

$$\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{2.9}$$

Gaussian distributions are easy to use for calculations because they are closed under multiplication and the integral can be calculated analytically. Both characteristics are useful for working with probabilities.

When we have a multivariate Gaussian Distribution as our model, we ask queries with respect to the model. The query is defined in Definition 2.2.5 and we can calculate all integrals to come up with the marginal distributions and calculate the new distribution based on Expression (2.2). Fortunately, there exists a better algorithm for calculating the conditional probability distribution queried by $P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e})$ in a multivariate Gaussian Distribution (Koller and Friedman, 2009). The conditional distribution is again a Gaussian distribution

$$P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e}) = \mathcal{N}(\boldsymbol{\mu}^\star, \boldsymbol{\Sigma}^\star), \tag{2.10}$$

where $\boldsymbol{\mu}^\star$ and $\boldsymbol{\Sigma}^\star$ can be calculated analytically with

$$\boldsymbol{\mu}^* = \boldsymbol{\mu}_{\boldsymbol{Q}} + \boldsymbol{\Sigma}_{\boldsymbol{QE}}\boldsymbol{\Sigma}_{\boldsymbol{EE}}^{-1}(\boldsymbol{e} - \boldsymbol{\mu}_{\boldsymbol{E}}), \tag{2.11}$$

$$\boldsymbol{\Sigma}^* = \boldsymbol{\Sigma}_{\boldsymbol{QQ}} - \boldsymbol{\Sigma}_{\boldsymbol{QE}}\boldsymbol{\Sigma}_{\boldsymbol{EE}}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{EQ}}. \tag{2.12}$$

The subscripts containing sets refer to the subset of entries in $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ corresponding to the randvars in given sets.

The upper bound for the time complexity of query answering with respect to a multivariate Gaussian distribution is in $O(N^3)$ where N is the number of randvars. The time complexity is driven by the matrix multiplications and matrix inversions necessary for Expressions (2.11) and (2.12).

2.4 Gaussian Probabilistic Graphical Networks

Having continuous randvars in a BN is not trivial. Marginalizing out randvars in the continuous case uses integrals over the randvars that should be marginalized. In the general form, there is no algorithm known to solve those integrals, which motivates more restrictive forms that avoid intractability. Shachter and Kenley (1989) have introduced Gaussian Bayesian networks (GBNs) under the name Gaussian influence diagrams as a possibility to have continuous distributed randvars in a BN, that allows for tractable inference.

Definition 2.4.1. A *Gaussian Bayesian network* is a BN in which all randvars are continuous and normally distributed. The edges represent linear relationships between the randvars. As in BNs, the joint density can be factorized using the conditional probability densities of X_i with $i = 1, ..., N$ given it parents $\boldsymbol{Pa}(X_i)$:

$$P(X_i|\boldsymbol{Pa}(X_i)) \sim \mathcal{N}\left(\mu_i + \sum_{X_k \in \boldsymbol{Pa}(X_i)} \beta_{k,i}(x_k - \mu_k), \sigma_i^2\right), \tag{2.13}$$

where μ_k and μ_i are node means, σ_i^2 is the node variance, $\beta_{k,i}$ represents the influence of parent X_k on its child X_i and x_k refers to the observed value for X_k.

Later in the thesis we also might write μ_{X_i} instead of μ_i. Whenever it is clear to which randvar the mean value is referring, we use the short form. Remark: The linear influence β is here multiplied with the parent's deviation from the mean $(x_k - \mu_k)$, which is following the structure by Shachter and Kenley (1989). Other authors multiply the linear factor directly with the variable value μ_k (Koller and Friedman, 2009). Using the deviation has the benefit, that the marginal means are equivalent to the node means, whereas using the variable name requires a further conversion step. Semantically both structures can be converted into multivariate Gaussian distributions.

Definition 2.4.2 (Kronecker delta)**.** Given two items A and B, the Kronecker delta function is defined as

$$\delta_{A,B} = \begin{cases} 1 & \text{if } A = B \\ 0 & \text{otherwise} \end{cases}. \tag{2.14}$$

In this dissertation, items are for example indexes or randvars.

Similarly to the discrete case, the GBN is a sparse representation for the multivariate Gaussian distribution described in Expression (2.8). The multivariate joint Gaussian distribution can be constructed from a GBN using an algorithm by Shachter and Kenley (1989), which has been implemented both in matrix notation and non-matrix notation. Using non-matrix notation, we can loop twice over the randvars and use the following equation for calculating the entries within the covariance matrix inductively,

$$\Sigma_{i,j} = \sum_{X_k \in \boldsymbol{Pa}(X_j)} \Sigma_{i,k}\beta_{k,j} + \delta_{i,j}\sigma_i^2, \tag{2.15}$$

where $i, j \in 1, \ldots, N$ and $i < j$ and δ is the Kronecker delta function as defined in Definition 2.4.2.

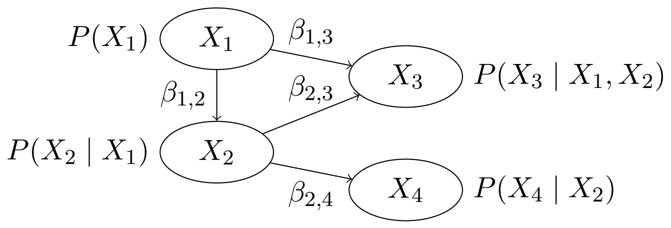


Figure 2.1: Example (G)BN with four randvars

Example 2.4.1 (Construction of the covariance matrix)**.** Given a network with four randvars X_1, X_2, X_3 and X_4 as visualized in Fig. 2.1, we use Expression (2.15) to calculate the joint distribution inductively:

$$\begin{aligned}
&\Sigma_{1,1} = \sigma_1^2 \\
&\Sigma_{2,1} = \Sigma_{1,2} = \Sigma_{1,1}\beta_{1,2} \\
&\Sigma_{2,2} = \Sigma_{2,1}\beta_{1,2} + \sigma_2^2 \\
&\Sigma_{3,1} = \Sigma_{1,3} = \Sigma_{1,1}\beta_{1,3} + \Sigma_{1,2}\beta_{2,3} \\
&\Sigma_{3,2} = \Sigma_{2,3} = \Sigma_{2,1}\beta_{1,3} + \Sigma_{2,2}\beta_{2,3} \\
&\Sigma_{3,3} = \Sigma_{3,1}\beta_{1,3} + \Sigma_{3,2}\beta_{2,3} + \sigma_3^2 \\
&\Sigma_{4,1} = \Sigma_{1,4} = \Sigma_{1,2}\beta_{2,4} \\
&\Sigma_{4,2} = \Sigma_{2,4} = \Sigma_{2,2}\beta_{2,4} \\
&\Sigma_{4,3} = \Sigma_{3,4} = \Sigma_{3,2}\beta_{2,4} \\
&\Sigma_{4,4} = \Sigma_{4,2}\beta_{2,4} + \sigma_4^2
\end{aligned}$$

Using matrix notation, we store the linear dependencies in a transition matrix $\boldsymbol{T}$, where an entry $T_{i,j}$ contains the value $\beta_{i,j}$ and describes the linear relationship between parent X_i and child X_j. A zero entry is equivalent to no direct edge between the randvars in the GBN. Using matrix notation we can calculate the off-diagonal entries of the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{i},j}$ belonging to a randvar X_j in one step by

$$\boldsymbol{\Sigma}_{\boldsymbol{i},j} = \boldsymbol{\Sigma}_{\boldsymbol{i},\boldsymbol{i}}\boldsymbol{T}_{\boldsymbol{i},j} \tag{2.16}$$

and the transpose

$$\boldsymbol{\Sigma}_{j,\boldsymbol{i}} = \boldsymbol{\Sigma}_{\boldsymbol{i},j}^T, \tag{2.17}$$

where $\boldsymbol{i} = \{1, ..., j-1\}$. The on-diagonal entries are calculated with

$$\Sigma_{j,j} = \boldsymbol{\Sigma}_{j,\boldsymbol{i}}\boldsymbol{T}_{\boldsymbol{i},j} + \sigma_j^2. \tag{2.18}$$

The starting point is

Algorithm 1 Converting a GBN to a multivariate Gaussian joint distribution (matrix notation)

```
procedure ConstructJoint(Topologically sorted randvars $\boldsymbol{X}$)
    $N \leftarrow |\boldsymbol{X}|$
    Initialize $N \times N$-dimensional covariance matrix $\boldsymbol{\Sigma}$
    $\Sigma_{1,1} \leftarrow \sigma^2_{X_1}$
    for $i \in 1, ..., N$ do
        $\boldsymbol{j} \leftarrow 1, \dots, i-1$
        $\boldsymbol{\Sigma}_{\boldsymbol{j},i} \leftarrow \boldsymbol{\Sigma}_{\boldsymbol{j},\boldsymbol{j}} \boldsymbol{T}_{\boldsymbol{j},i}$
        $\boldsymbol{\Sigma}_{i,\boldsymbol{j}} \leftarrow \boldsymbol{\Sigma}^T_{\boldsymbol{j},i}$
        $\Sigma_{i,i} \leftarrow \boldsymbol{\Sigma}_{i,\boldsymbol{j}} \boldsymbol{T}_{\boldsymbol{j},i} + \sigma^2_{X_i}$
    return $\boldsymbol{\Sigma}$
```

$$\Sigma_{1,1} = \sigma_1^2. \tag{2.19}$$

The algorithm for constructing the covariance matrix using matrix notation is described in Alg. 1.

Example 2.4.2 (Construction of the covariance matrix with matrix notation)**.** Converting Example 2.4.2 into matrix notation results in

$$\begin{aligned}
\Sigma_{1,1} &= \sigma_1^2, \\
\Sigma_{2,1} &= \Sigma_{1,2} = \Sigma_{1,1} T_{1,2}, \\
\Sigma_{2,2} &= \Sigma_{2,1} T_{1,2} + \sigma_2^2, \\
\boldsymbol{\Sigma}_{3,1:2} &= \boldsymbol{\Sigma}_{1:2,3} = \boldsymbol{\Sigma}_{1:2,1:2} \boldsymbol{T}_{1:2,3}, \\
\Sigma_{3,3} &= \boldsymbol{\Sigma}_{3,1:2} \boldsymbol{T}_{1:2,3} + \sigma_3^2, \\
\boldsymbol{\Sigma}_{4,1:3} &= \boldsymbol{\Sigma}_{1:3,4} = \boldsymbol{\Sigma}_{1:3,1:3} \boldsymbol{T}_{1:3,4}, \\
\Sigma_{4,4} &= \boldsymbol{\Sigma}_{4,1:3} \boldsymbol{T}_{1:3,4} + \sigma_4^2,
\end{aligned}$$

where

$$\boldsymbol{T} = \begin{bmatrix} 0 & \beta_{1,2} & \beta_{1,3} & 0 \\ 0 & 0 & \beta_{2,3} & \beta_{2,4} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

It is important to remark that we use the joint distribution for query answering is this thesis. This might seem odd because we do not use a local propagation scheme which is a core feature of PGMs. In the next chapter, we will see that we still use the structure of the network, when developing query answering operations. Nevertheless investigating local schemes — especially for lifted query answering — is an interesting path for further research.

Chapter 3
Lifting Gaussian Bayesian Networks

One way to improve efficiency of query answering is to exploit certain characteristics of the PGM at hand. Poole (2003) introduces first-order probabilistic inference, which exploits symmetries in a model by combining indistinguishable instances using logical variables (logvars) to reason with representatives for the represented instances. The instances are only indistinguishable as long as no evidence is introduced, i.e., as long as the instances are not observed. Using a compact representation and performing inference with representatives is also referred to as lifting and has been an active research field in the past years (Kimmig *et al.*, 2015; Sharma *et al.*, 2018; Holtzen *et al.*, 2020). Lifting has its benefits if the groups of randvars are large, which is usually the case when relationships of randvars on instance level are concerned. In discrete settings, lifting has shown remarkable efficiency gains for query answering. For discrete PGMs, Taghipour *et al.* (2013) present a lifted variable elimination algorithm and for repeated query answering Braun and Möller (2016) develop a lifted version of the junction tree algorithm originally proposed for proposiitional PGMs by Lauritzen and Spiegelhalter (1988). In the continuous setting, Choi *et al.* (2010) present a lifted version of variable elimination in factor graphs (a different type of PGMs that is not focus of this dissertation) with Gaussian pairwise potentials. In this chapter, we look at how to apply lifting to GBNs which is a different class of PGMs that has not been lifted before. This chapter is based on the following publications:

> Mattis Hartwig and Ralf Möller. Lifted Query Answering in Gaussian Bayesian Networks. In *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 233–244. PMLR, 2020
>
> Mattis Hartwig, Tanya Braun, and Ralf Möller. Handling Overlaps When Lifting Gaussian Bayesian Networks. In *IJCAI-21 Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 4228–4234. IJCAI Organization, 2021
>
> Mattis Hartwig, Ralf Möller, and Tanya Braun. An Extended View on Lifting Gaussian Bayesian Networks. Submitted to Elsevier Artificial Intelligence Journal

This chapter contains the first three contributions of this dissertation. We start by explaining the lifting-specific preliminaries and introducing a running example used throughout this chapter. Then, we develop a lifted version of Shachter and Kenley's (1989) ground-level algorithm to construct a joint distribution from a GBN (Contribution **1**) and how to work with the lifted joint distribution using lifted operations (Contribution **2**). Afterwards, we describe an algorithm using the lifted joint distribution for lifted query answering (Contribution **3**). The new approaches are evaluated with complexity analyses and experiments measuring runtimes.

3.1 Lifting Preliminaries

This section covers preliminaries specific for lifting GBNs. In slight abuse of notation, we use set operations for sequences, where we apply the operation to the set containing the sequence elements but keep the sequence order intact.

3.1.1 Parameterized randvars

Parameterized randvars (PRVs) allow for grouping of indistinguishable randvars. Indistinguishable randvars share the same characteristics in terms of node means, node variances and parent structure. Logvars identify groups and PRVs to represent a set of indistinguishable randvars. The aim of lifting is to work solely with the PRVs as representatives for the randvars we get when the logvars of the PRVs are instantiated. The model in which all PRVs are instantiated with the logvar constants is called the ground model and the model that only contains PRVs the lifted model.

Definition 3.1.1 (Parameterized randvar, grounding)**.** Let $\boldsymbol{\xi}$ be a set of randvar names and $\mathcal{D}$ be a set of logvar names. A *Parameterized random variable* Y is a syntactical construct of a randvar name $\xi \in \boldsymbol{\xi}$ combined with a sequence of logvar names $\mathcal{F} \subseteq \mathcal{D}$ into $\xi(\mathcal{F})$. We use $\mathcal{X}$ to refer to all ground level randvars and $\mathcal{L}$ to refer to sequence of all logvars with a fixed ordering. The domain $\boldsymbol{D}(L) = \{l^1, ..., l^H\}$ contains the constants of logvar $L \in \mathcal{L}$. The domain of a sequence of logvars is defined as $\boldsymbol{D}(\boldsymbol{L}) = \times_{L \in \boldsymbol{L}} \boldsymbol{D}(L)$. The term $lv(Y)$ refers to the logvars of Y, i.e., $\boldsymbol{L}$. *Grounding* a PRV Y_s with a corresponding logvar L_s results in a set of randvars $\boldsymbol{gr}(Y_s) = \{\xi_s(l^1), ..., \xi_s(l^H)\}$, with $\boldsymbol{D}(L_s) = \{l^1, ..., l^H\}$.

The number of randvars represented by a PRV Y is determined by the domain size of its logvars, i.e., $|\boldsymbol{D}(lv(Y))|$. A propositional randvar X can be interpreted as a PRV $Y = X(L)$ with $|\boldsymbol{D}(L)| = 1$. When we say a randvars X *belongs* to a PRV Y, we mean that $X \in \boldsymbol{gr}(Y)$. Each PRV Y also has a range denoted as $R(Y)$ that contains the possible values for randvars $\boldsymbol{gr}(Y)$ analogously to the range of individual randvars $R(X)$. In this dissertation, we work with continuous normally distributed randvars which results in the general $R(X) = \mathbb{R}$. Instead of specifying a discrete conditional probability table for each PRV, we specify a lifted node mean η and lifted node variance λ, that apply to all randvars in $\boldsymbol{gr}(\mathrm{Y})$:

$$\lambda = \sigma_X^2, \text{for } X \in \boldsymbol{gr}(Y). \tag{3.1}$$

Parents of Y_s are denoted as $Y_u \in \boldsymbol{Pa}(Y_s)$.

3.1.2 Parameterized Gaussian Bayesian Networks

Instead of N ground randvars $X_1, \dots, X_N$, a parameterized GBN contains M PRVs $Y_1, \dots, Y_M$. Each PRV Y_s, with $s = 1, ..., M$, has a lifted node mean η_s and lifted node variance λ_s. Edges are also defined between PRV nodes. An edge between a parent Y_u and a child Y_s has a corresponding variable $\zeta_{Y_u,Y_s} \neq 0$ that describes the linear relationship between the PRVs analogously to the β in propositional GBNs of Definition 2.4.1. Grounding a PRV Y_s leads to a set of ground randvars $\boldsymbol{gr}(Y_s)$ with $|\boldsymbol{gr}(Y_s)| = |\boldsymbol{D}(\boldsymbol{L}_s)|$ that have the same mean and variance. Grounding a topologically ordered list of PRVs results in a topologically ordered list of ground randvars because the parent-child relationships for each ground randvar are defined by the parent-child relationships of the corresponding PRV. A propositional randvar X can be interpreted as a PRV Ys with $|\boldsymbol{D}(L_s)| = 1$. Given a parent PRV Y_s and a child PRV Y_t, the logvar sequences can be either disjoint or overlapping resulting in different conditional independencies between parent and child randvars.

Disjoint logvar sets: The logvars $\boldsymbol{L}_s$ of the parent PRV Y_s and the logvars $\boldsymbol{L}_t$ of the child PRV Y_t are disjoint, i.e. $\boldsymbol{L}_s \cap \boldsymbol{L}_t = \emptyset$. Disjoint logvar sets result in a relationship from every randvar in $\boldsymbol{gr}(Y_s)$ to every randvar in $\boldsymbol{gr}(Y_t)$.

Overlapping logvar sets: The logvars $\boldsymbol{L}_s$ of the parent PRV Y_s and the logvars $\boldsymbol{L}_t$ of the child PRV Y_t overlap, i.e., $\boldsymbol{L}_O = \boldsymbol{L}_s \cap \boldsymbol{L}_t \neq \emptyset$. Grounding results in a relation where each child node is influenced by all $|\boldsymbol{D}(\boldsymbol{L}_s \setminus \boldsymbol{L}_t)|$ parents that share the same grounding of $\boldsymbol{L}_O$.

To get better intuition of the influence of overlaps on the ground network, see Fig. 3.2. For ease of notation, we assume that all sequences $\boldsymbol{L} \subset \boldsymbol{\mathcal{L}}$ are in line with the ordering in $\boldsymbol{\mathcal{L}}$. This sorting places no restriction on the expressivity of the model. In the course of this dissertation, we need the following helper functions regarding logvar sequences.

Definition 3.1.2 (Seq-, dim- and ov-function)**.** Let L be a logvar and $\boldsymbol{L}_s \subseteq \boldsymbol{\mathcal{L}}$ and $\boldsymbol{L}_t \subseteq \boldsymbol{\mathcal{L}}$ two sequences of logvars. Then, we define

$$dim(L, \boldsymbol{L}_s) = \begin{cases} |\boldsymbol{D}(L)| & \text{if } L \in \boldsymbol{L}_i \\ 1 & \text{otherwise} \end{cases} \tag{3.2}$$

$$ov(L, \boldsymbol{L}_s, \boldsymbol{L}_t) = \begin{cases} 1 & \text{if } L \in (\boldsymbol{L}_s \cap \boldsymbol{L}_t) \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

Additionally, we need two helper functions when working with randvars and PRVs.

Definition 3.1.3 (Any- and lif-function)**.** For a PRV Y and a randvar X with $X \in \boldsymbol{gr}(Y)$, the operation $any(Y)$ returns any individual randvar belonging to PRV Y. The operation $lif(X)$ returns the corresponding PRV, here $lif(X) = Y$.

3.2 Introducing a Running Example

We setup a running example to illustrate the operations performed in this chapter, which can be seen in Fig. 3.1. The example is of course designed to show-case what is going on in the formulas throughout the chapter and not to be fully realistic, also containing not too serious elements like modelling the coffee consumption of doctors and nurses.

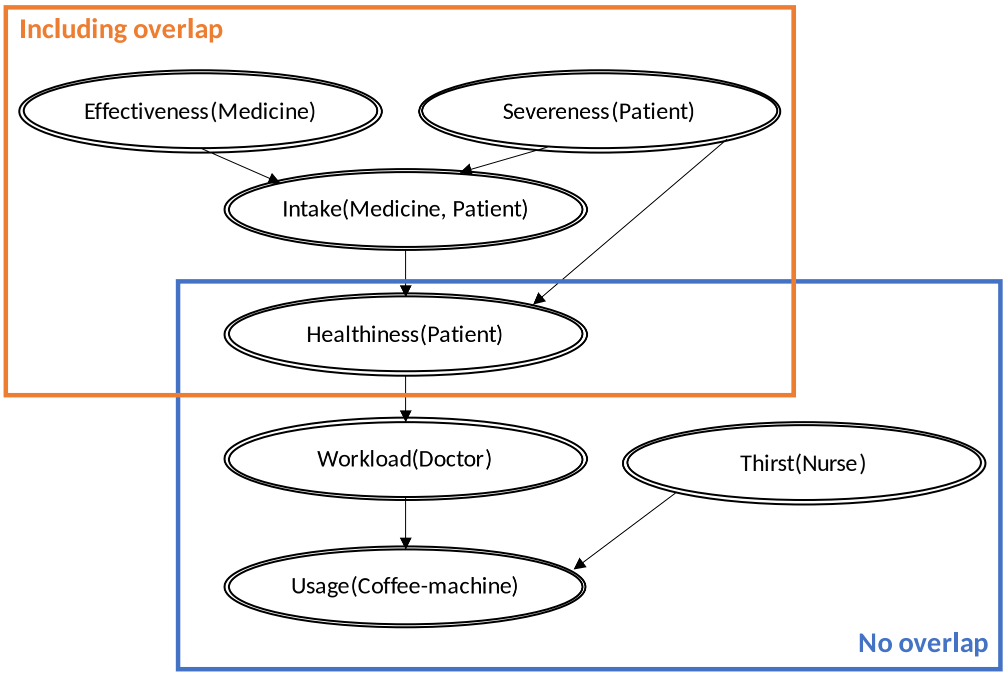


Figure 3.1: Running Example

The example contains the randvars *Effectiveness (E)*, *Intake (I)*, *Severeness (S)*, *Healthiness (H)*, *Workload (W)*, *Thirstiness (T)*, and *Usage (U)* and the logvars *Medicine (M)*, *Patient (P)*, *Doctor (D)*, *Nurse (N)*, and *Coffee-machine (C)*. Randvars and logvars are combined into PRVs and put into dependence as follows: The *Effectiveness* of a *Medicine*, $E(M)$, and the *Severeness* of a *Patient*'s disease, $S(P)$, have an influence on the *Intake* of a specific *Medicine* for a *Patient*, $I(M, P)$. The *Severeness* of a *Patient*'s disease and the *Intake* of a specific *Medicine* for a *Patient* have then an influence on the *Healthiness* of a *Patient*, $H(P)$, after the treatment, which then influence the *Workload* of the *Doc-*

tors $W(D)$. The *Workload* of the *Doctors* and the *Thirst* of the *Nurses*, $T(N)$, have an influence on the *Usage* of the *Coffee-machine*, $U(C)$. In the following sections, we will use individual parts of this network as a demonstrator. The starting point for us is the blue box, where PRVs do not contain an overlap between their logvar sequences, followed by a generalization for the parts that contain overlaps (orange box). We use a topological ordering of $(E(M), S(P), I(M,P), H(P), T(N), W(D), U(C))$ for our example, together with a global logvar sequence $\mathcal{L} = (M, P, N, D, C)$ defining the global logvar ordering. In the formal descriptions, we always have an iterator (often denoted as $s = 1, \ldots, M$) over the PRVs. In our example calculations, we try to use the PRV names as often as possible, but when iterating over example PRVs, e.g., within a sum, we use $Y_1, \ldots, Y_7$ as synonyms for $E(M)$, $S(P)$, $I(M,P)$, $H(P)$, $T(N)$, $W(D)$, $U(C)$ respectively, to ensure better readability. We also define two helper functions:

Definition 3.2.1 (PRV position, preceding PRVs)**.** Given a set of PRVs $\boldsymbol{Y}$ and an ordering O, we define *pos*(Y_s) of a PRV $Y_s \in \boldsymbol{Y}$ to return the position in the ordering

$$pos(Y_s) = s \tag{3.4}$$

and *pre*(Y_s) to return all PRVs preceding Y_s

$$pre(Y_s) = \{Y_r | r < s\}. \tag{3.5}$$

The parameters $\boldsymbol{\lambda}, \boldsymbol{\eta}$ and $\boldsymbol{\zeta}$ of the parameterized GBN are as follows

$$\boldsymbol{\lambda} = \begin{pmatrix} \lambda_{E(M)} \\ \lambda_{S(P)} \\ \lambda_{I(M,P)} \\ \lambda_{H(P)} \\ \lambda_{T(N)} \\ \lambda_{W(D)} \\ \lambda_{U(C)} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}, \boldsymbol{\eta} = \begin{pmatrix} \eta_{E(M)} \\ \eta_{S(P)} \\ \eta_{I(M,P)} \\ \eta_{H(P)} \\ \eta_{T(N)} \\ \eta_{W(D)} \\ \eta_{U(C)} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 3 \\ 0 \\ 2 \\ 3 \\ 4 \end{pmatrix}, \text{ and } \begin{aligned} \zeta_{E(M),I(M,P)} &= 2 \\ \zeta_{S(P),I(M,P)} &= 5 \\ \zeta_{S(P),H(P)} &= -3 \\ \zeta_{I(M,P),H(P)} &= 4. \\ \zeta_{H(P),W(D)} &= -2 \\ \zeta_{T(N),U(C)} &= 2 \\ \zeta_{W(D),U(C)} &= 3 \end{aligned} \tag{3.6}$$

The domain size of the PRVs in the network is defined by the cardinality of the logvar domains, stored in what we later refer to as a cardinality vector $\boldsymbol{\tau}$. In our example calculations, we use the following domain sizes:

$$\boldsymbol{\tau} = \begin{pmatrix} \tau_M \\ \tau_P \\ \tau_N \\ \tau_D \\ \tau_C \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \\ 3 \\ 2 \\ 3 \end{pmatrix}. \tag{3.7}$$

Behind this parameterized GBN lies a grounded GBN. If the logvar M has three constants, $\boldsymbol{D}(M) = \{Medicine1, Medicine2, Medicine2\}$, and the logvar P two constants,

$\boldsymbol{D}(P) = \{John, Mary\}$, the resulting ground network of the first three PRVs $E(M)$, $S(P)$, and $I(M, P)$ would consist of 11 randvars, each with their own μ, σ^2, and β values to store. Figure 3.2 shows a partial visualisation of the ground network. With a domain size of ten medicines and 100 patients the ground network of only the first three PRVs would already contain 1,110 randvars. This explosion of the number of randvars shows how important efficient approaches to handling those networks are.

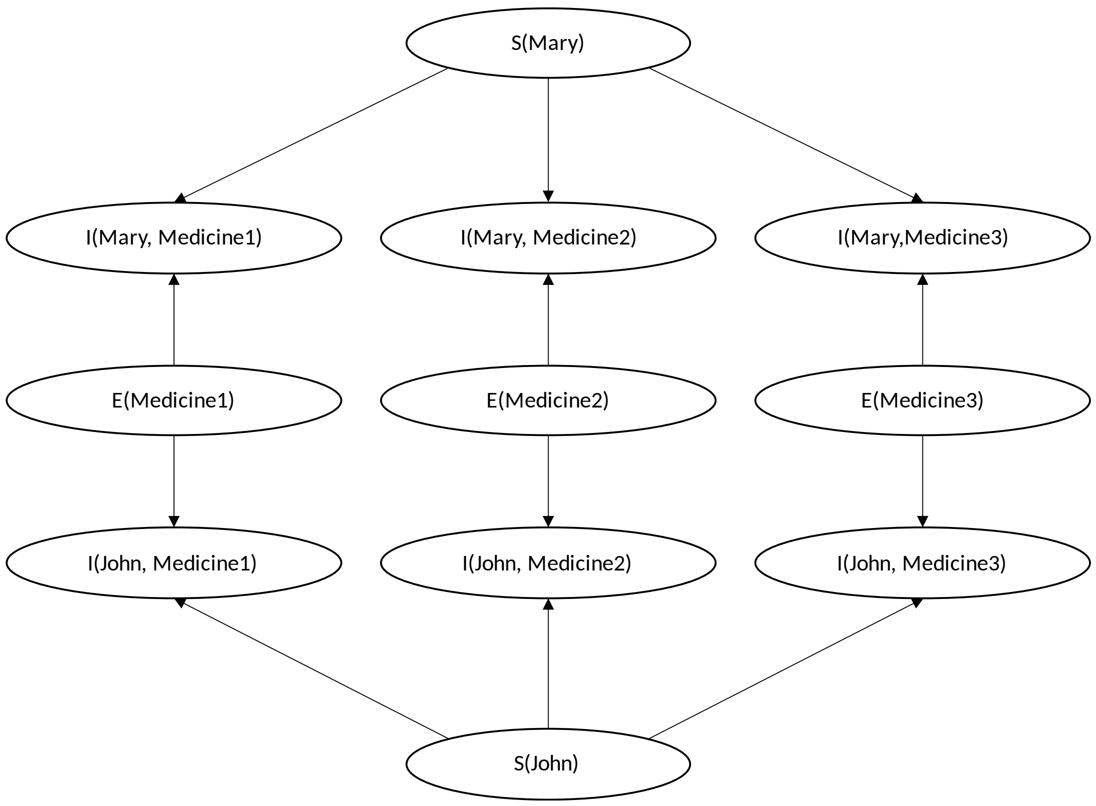


Figure 3.2: Grounding parts of the running example

Of course there are also unrealistic assumptions occurring in the modeling of this simplified real-world scenario. For example we assume that the health levels of all patients influence the workload of all doctors, whereas in reality probably certain doctors are responsible for a specific group of patients. This assumption could be relaxed by introducing a new logvar that could be the station or department where a doctor works and where a patient is treated. But as said, our example should be very simple and focus on containing different types of relationships to better illustrate the upcoming ideas.

3.3 Constructing the Lifted Joint Distribution

The starting point for constructing a lifted joint given a parameterized GBN is the grounded implementation of Alg. 1 originally developed by Shachter and Kenley (1989). This section describes a lifted version of the algorithm. If the number of indistinguishable instances is high, the covariance matrix of the multivariate normal distribution is filled with many duplicate values. We present a lifted representation for more efficient memory usage and faster calculations. We start by explaining the basic case where PRVs are not allowed to have overlapping logvar sets using the non-matrix notation, followed by a matrix version of the same algorithms. Both approaches have their own intuition, adding their own contribution to the discussion of (parameterized) GBNs. The non-matrix notation gives better intuition of how conditional independencies in the network result in less summations, when summing up all influences caused by parent randvars. It also shows very clearly how identical values occur in the case of indistinguishable randvars and how identical values can simplify calculations. The matrix notation however is better suited for generalizing the algorithm for the overlapping case, because we can exploit block structure within the matrices to simplify equations, enabling scenarios such as one part of the running example, where the severeness of a patient's illness only influences the intake of medicine for that specific patient.

3.3.1 The Base Case Without Overlaps - Non-matrix Notation

The algorithm by Shachter and Kenley (1989) starts with a topologically ordered list of randvars in the GBN. As already described in the previous chapter in Expression (2.15), the symmetric covariance matrix $\mathbf{\Sigma}$ of the joint distribution $P(\mathcal{X}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$ is created inductively by

$$\Sigma_{X_j,X_i} = \Sigma_{X_i,X_j} = \left(\sum_{X_k \in \boldsymbol{Pa}(X_j)} \Sigma_{X_i,X_k} \beta_{X_k,X_j} \right) + \delta_{X_i,X_j} \sigma^2_{X_i}, \tag{3.8}$$

here indexed using full randvar names, with $i < j$ meaning X_i comes before X_j in the topological ordering. For better reference during this chapter, we have repeated the equation but instead of using the short indexing with i and j, we index with the full randvar name, to be more consistent when introducing PRVs into the equation.

Instead of looking into propositional randvars X_i and X_j, we are now looking at two sets of equally behaving randvars $\boldsymbol{gr}(Y_s)$ and randvars $\boldsymbol{gr}(Y_t)$ grouped into PRVs Y_s and Y_t with $s, t = 1, ..., M$, respectively. In the ground case, we would calculate the covariance between all $|\boldsymbol{D}(\boldsymbol{L}_s)|$ randvars $\boldsymbol{gr}(Y_s)$ and all $|\boldsymbol{D}(\boldsymbol{L}_t)|$ randvars $\boldsymbol{gr}(Y_t)$. If there exists a parent child relationship between Y_u and Y_s, all indistinguishable randvars $\boldsymbol{gr}(Y_u)$ are parents of all randvars $\boldsymbol{gr}(Y_s)$ due to the non-overlap in the logvar sequences. Based on these known relationships we can reformulate Expression (3.8) into

$$\Sigma_{X_i,X_j} = \left[\sum_{Y_u \in \boldsymbol{Pa}(lif(X_j))} \left(\sum_{X_k \in \boldsymbol{gr}(Y_u)} \Sigma_{X_i,X_k} \beta_{X_k,X_j} \right) \right] + \delta_{X_i,X_j} \sigma^2_{X_i}. \tag{3.9}$$

The sum in parentheses works on ground level. Based on Section 3.1.2, all β_{X_k,X_j} for $X_k \in \boldsymbol{gr}(Y_u)$ are equal to $\zeta_{lif(X_k),lif(X_j)}$ in the non-overlapping case. If Σ_{X_i,X_k} was equal for all X_i and X_k, we could calculate it once and multiply it with the number of randvars $|\boldsymbol{D}(\boldsymbol{L}_u)|$ in $\boldsymbol{gr}(Y_u)$ to replace the sum. However, if X_i is equal to X_k, the recursive Σ_{X_i,X_k} contains a different value due to the Kronecker delta term resulting in an additional summand. The delta term within Σ_{X_i,X_k} prevents us from multiplying $\Sigma_{X_i,X_k} \zeta_{lif(X_k),lif(X_j)}$ with the number of randvars $|\boldsymbol{D}(\boldsymbol{L}_u)|$ to replace the sum. To get rid of the sum in parentheses nevertheless, we differentiate between X_i being a parent of X_j and X_i not being a parent X_j:

Case 1: If the randvar X_i is no parent of the randvar X_j, the $\delta_{X_i,X_k}\sigma^2_{X_i}$ term in the referenced Σ_{X_i,X_k} is always zero, resulting in a fully equal covariance Σ_{X_i,X_k} for all randvars $X_k \in \boldsymbol{gr}(Y_u)$, which reduces the second summation to a product between any covariance $\Sigma_{i,any(Y_u)}$, the number of parent randvars $|\boldsymbol{D}(\boldsymbol{L}_u)|$, and the linear dependency $\zeta_{Y_u,lif(X_j)}$:

$$\Sigma_{X_i,X_j} = \left[\sum_{Y_u \in \boldsymbol{Pa}(lif(X_j))} \Sigma_{X_i,any(Y_u)} |\boldsymbol{D}(\boldsymbol{L}_u)| \zeta_{Y_u,lif(X_j)} \right] + \delta_{X_i,X_j} \sigma^2_{X_i}. \tag{3.10}$$

Case 2: If the randvar X_i is a parent of the randvar X_j, then exactly one randvar in $\boldsymbol{gr}(Y_u)$ will be equal to randvar X_i, which would result in a different covariance $\Sigma_{i,k}$. The key is, that if randvar X_i is a parent of randvar X_j, all other randvars $\boldsymbol{gr}(lif(X_i))$ of the corresponding PRV $lif(X_i)$ are also parents of randvars $\boldsymbol{gr}(lif(X_j))$. This means that, independent of the specific randvar in the covariance function, there will always be exactly one covariance Σ_{X_i,X_k} where the $\delta_{X_i,X_k}\sigma^2_{X_i}$ is nonzero.

Based on the two cases, we can reformulate Equation 3.9 into

$$\Sigma_{X_i,X_j} = \left[\sum_{Y_u \in \boldsymbol{Pa}(lif(X_j)} \left(\Sigma_{X_i,any(Y_u)} |\boldsymbol{D}(\boldsymbol{L}_u)| + \delta_{Y_u,lif(X_j)} \lambda_{lif(X_k)} \right) \zeta_{Y_u,lif(X_j)} \right] + \delta_{X_i,X_j} \lambda_{lif(X_i)}. \tag{3.11}$$

Since the sum over $Y_u \in \boldsymbol{Pa}(lif(X_j))$ is equal for all combinations between randvars $\boldsymbol{gr}(lif(X_i))$ and randvars $\boldsymbol{gr}(lif(X_j))$, we introduce a new symbol ρ to store this value that is equal for all randvars belonging to the same PRV. $\boldsymbol{\rho}$ can be seen as a PRV covariance matrix. Introducing this new matrix as a result of the sum in Expression (3.11) results in

$$\rho_{Y_s,Y_t} = \sum_{Y_u \in \boldsymbol{Pa}(Y_t)} \left(\rho_{Y_s,Y_u} |\boldsymbol{D}(\boldsymbol{L}_u)| + \delta_{Y_u,X_t} \lambda_{Y_u}\right) \zeta_{Y_u,Y_t}. \tag{3.12}$$

The last term $\delta_{X_i,X_j}\lambda_{lif(X_i)}$ of Expression (3.11) only needs to be added if the ground covariance between two equal randvars X_i and X_i is calculated. This additional value is already stored in the $\boldsymbol{\lambda}$ vector. We can use the PRV covariance $\boldsymbol{\rho}$ matrix and the $\boldsymbol{\lambda}$ vector to calculate a ground covariance with

$$\Sigma_{X_i,X_j} = \rho_{lif(X_i),lif(X_j)} + \delta_{X_i,X_j}\lambda_{lif(X_i)}. \tag{3.13}$$

Expressions (3.12) and (3.13) show that we can calculate the ground covariance matrix only using $\boldsymbol{\rho}$ and $\boldsymbol{\lambda}$. All lifted covariances ρ_{Y_s,Y_t} can be stored in an $M \times M$-dimensional matrix $\boldsymbol{\rho}$ and the variances λ_{X_s} can be stored in an M-dimensional vector $\boldsymbol{\lambda}$, no longer requiring storage depending on domain sizes.

In addition to the covariance matrix, the multivariate Gaussian also needs a mean vector $\boldsymbol{\mu}$. The ground mean-vector consists of the means of all randvars. Since the randvars $\boldsymbol{gr}(Y_s)$ of a PRV Y_s have the same mean, the M-dimensional PRV mean vector $\boldsymbol{\eta}$ is a lifted version of $\boldsymbol{\mu}$. We get the mean for a ground randvar X_i by expanding the lifted mean vector $\boldsymbol{\eta}$ as $\mu_{X_i} = \eta_{lif(X_i)}$.

In summary, to store all information of the lifted joint distribution over M PRVs $Y_1, ..., Y_M$, we need an M-dimensional lifted mean vector $\boldsymbol{\eta}$, an $M \times M$-dimensional PRV covariance matrix $\boldsymbol{\rho}$, and an M-dimensional node variance vector $\boldsymbol{\lambda}$. Additionally, we need to store the cardinalities of the randvars, i.e., the domain sizes of the corresponding logvar sequences, needed in Expression (3.12). In the case with no overlap, each PRV has its own set of logvars so we could store cardinalities in an M-dimensional vector but in preparation of the general case, we store individual domain sizes in a $|\mathcal{L}|$-dimensional cardinality vector $\boldsymbol{\tau}$. Let us look at our example to see Expression (3.12) at work.

Example 3.3.1. As a basis, we use the blue part with no overlapping logvars of the parameterized GBN visualized in Fig. 3.1. The parameters for $\boldsymbol{\lambda}$ and $\boldsymbol{\zeta}$ are defined in Expressions (3.6) and (3.7). The lifted calculations from Expression (3.12) to construct the joint covariance matrix are as follows:

$$\begin{aligned}
\rho_{H(P),H(P)} &= 0, \text{ because } H(P) \text{ has not parent} \\
\rho_{H(P),T(N)} &= 0, \text{ because } T(N) \text{ has not parent} \\
\rho_{T(N),T(N)} &= 0, \text{ because } T(N) \text{ has not parent} \\
\rho_{H(P),W(D)} &= (\rho_{H(P),H(P)}\tau_P + \lambda_{H(P)})\zeta_{H(P),W(D)} = (0 \cdot 2 + 1)(-2) = -2 \\
\rho_{T(N),W(D)} &= (\rho_{T(N),H(P)}\tau_P + 0)\zeta_{H(P),W(D)} = (0 \cdot 2 + 0)(-2) = 0 \\
\rho_{W(D),W(D)} &= (\rho_{W(D),H(P)}\tau_P + 0)\zeta_{H(P),W(D)} = (-2 \cdot 2 + 0)(-2) = 8 \\
\rho_{H(P),U(C)} &= (\rho_{H(P),T(N)}\tau_N + 0)\zeta_{T(N),U(C)} + (\rho_{H(P),W(D)}\tau_D + 0)\zeta_{W(D),U(C)} \\
&= (0 \cdot 3 + 0)2 + (-2 \cdot 2 + 0)3 = -12 \\
\rho_{T(N),U(C)} &= (\rho_{T(N),T(N)}\tau_N + \lambda_{T(N)})\zeta_{T(N),U(C)} + (\rho_{T(N),W(D)}\tau_D + 0)\zeta_{W(D),U(C)} \\
&= (0 \cdot 3 + 2)2 + (0 \cdot 2 + 0)3 = 4 \\
\rho_{W(D),U(C)} &= (\rho_{W(D),T(N)}\tau_N + 0)\zeta_{T(N),U(C)} + (\rho_{W(D),W(D)}\tau_D + \lambda_{W(D)})\zeta_{W(D),U(C)} \\
&= (0 \cdot 3 + 0)2 + (8 \cdot 2 + 3)3 = 57 \\
\rho_{U(C),U(C)} &= (\rho_{U(C),T(N)}\tau_N + 0)\zeta_{T(N),U(C)} + (\rho_{U(C),W(D)}\tau_D + 0)\zeta_{W(D),U(C)} \\
&= (4 \cdot 3 + 0)2 + (57 \cdot 2 + 0)3 = 366
\end{aligned}$$

The resulting $\boldsymbol{\rho}$ matrix is then

$$\boldsymbol{\rho} = \begin{bmatrix} 0 & 0 & -2 & -12 \\ 0 & 0 & 0 & 4 \\ -2 & 0 & 8 & 57 \\ -12 & 4 & 57 & 366 \end{bmatrix}.$$

Getting the variance of $H(Doctor1)$ results in a grounding of

$$\Sigma_{W(Doctor1),W(Doctor1)} = \rho_{W(D),W(D)} + \lambda_{W(D)} = 8 + 3 = 11.$$

Appendix A.1 contains the same example calculated on ground level using Expression (2.15). In the ground matrix, the values from $\boldsymbol{\lambda}$ are added to the diagonal resulting for example in a value of 370.

In the next section, we will look at the same case of a parameterized GBN without overlap using the matrix notation.

3.3.2 The Base Case Without Overlaps - Matrix Notation

In this section, we use the matrix notation of Shachter and Kenley's (1989) approach for which pseudocode can be found in Alg. 1. Before diving into the algorithm, we introduce two basic rules for working with identity and all-ones matrices that are essential for the matrix notation version. An all-ones matrix $\mathbf{J}$ contains one values in every entry and an identify matrix $\mathbf{I}$ contains one values on the diagonal and zeros everywhere else.

The first rule relates to the multiplication of all-ones matrices and has been covered by Timm (2002). The second rule follows directly from the first one.

Lemma 3.3.1. *Given two all-ones matrices $\boldsymbol{A} = \mathbf{J}_{E\times F}$ and $\boldsymbol{B} = \mathbf{J}_{F\times G}$ with $E, F, G \in \mathbb{N}$, the product of the two matrices is again an all-ones matrix multiplied by F*

$$\boldsymbol{AB} = \mathbf{J}_{E\times F}\mathbf{J}_{F\times G} = F\mathbf{J}_{E\times G} \tag{3.14}$$

Proof. With matrix algebra, each entry at position (i, j), with $i \in \{1, ..., E\}$ and $j \in \{1, ..., G\}$ of the resulting matrix is calculated by the sum

$$\sum_{k=1}^{F} a_{i,k}b_{k,i} = \sum_{k=1}^{F} 1 = F. \tag{3.15}$$

An $E \times G$ dimensional matrix whose elements are all equal to F can be written as $F\mathbf{J}_{E\times G}$. □

The second lemma follows directly from Lemma 3.3.1.

Lemma 3.3.2. *Given matrices*

$$\boldsymbol{A} = p\mathbf{J}_{E\times E} + q\mathbf{I}_{E\times E}, \tag{3.16}$$

$$\boldsymbol{B} = r\mathbf{J}_{E\times E} + s\mathbf{I}_{E\times E} \text{ and} \tag{3.17}$$

$$\boldsymbol{C} = t\mathbf{J}_{E\times G} \tag{3.18}$$

it holds that

$$\boldsymbol{AB} = u\mathbf{J}_{E\times E} + v\mathbf{I}_{E\times E} \text{ and} \tag{3.19}$$

$$\boldsymbol{AC} = w\mathbf{J}_{E\times G} \tag{3.20}$$

with

$$u = Epr + ps + qr, \tag{3.21}$$

$$v = qs \text{ and} \tag{3.22}$$

$$w = Ept + qt. \tag{3.23}$$

Proof. Using the distributivity characteristics of matrix multiplication and Lemma 3.3.1, it holds that

$$\begin{aligned}\boldsymbol{AB} &= u\mathbf{J}_{E\times E} + v\mathbf{I}_{E\times E}\\ &= (p\mathbf{J}_{E\times E} + q\mathbf{I}_{E\times E})(r\mathbf{J}_{E\times E} + s\mathbf{I}_{E\times E})\\ &= Epr\mathbf{J}_{E\times E} + ps\mathbf{J}_{E\times E} + qr\mathbf{J}_{E\times E} + qs\mathbf{I}_{E\times E}\\ &= (Epr + ps + qr)\mathbf{J}_{E\times E} + qs\mathbf{I}_{E\times E}\end{aligned} \tag{3.24}$$

and

$$\begin{aligned}\boldsymbol{AC} &= u\mathbf{J}_{E\times E} + v\mathbf{I}_{E\times E} \\ &= (p\mathbf{J}_{E\times E} + q\mathbf{I}_{E\times E})(t\mathbf{J}_{E\times G}) \\ &= Ept\mathbf{J}_{E\times E} + qt\mathbf{J}_{E\times E} \\ &= (Ept + qt)\mathbf{J}_{E\times E}.\end{aligned} \tag{3.25}$$

□

For better readability, we restate Expressions (2.16) to (2.19) as the starting point for the lifted construction using matrix notation:

$$\Sigma_{X_1,X_1} = \sigma^2_{X_1}, \tag{3.26}$$
$$\boldsymbol{\Sigma}_{\boldsymbol{X_i},X_j} = \boldsymbol{\Sigma}_{\boldsymbol{X_i},\boldsymbol{X_i}}\boldsymbol{T}_{\boldsymbol{X_i},X_j}, \tag{3.27}$$
$$\boldsymbol{\Sigma}_{X_j,\boldsymbol{X_i}} = \boldsymbol{\Sigma}^T_{\boldsymbol{X_i},X_j}, \tag{3.28}$$
$$\Sigma_{X_j,X_j} = \boldsymbol{\Sigma}_{X_j,\boldsymbol{X_i}}\boldsymbol{T}_{\boldsymbol{X_i},X_j} + \sigma^2_{X_j}. \tag{3.29}$$

For lifting the Shachter and Kenley algorithm, we structure the covariance matrix $\boldsymbol{\Sigma}$ and the transition matrix $\boldsymbol{T}$ into $M \times M$ blocks. For $s, t \in \{1, \dots, M\}$, we identify the blocks with $\boldsymbol{\Sigma}_{\boldsymbol{gr}(Y_s),\boldsymbol{gr}(Y_t)}$ and $\boldsymbol{T}_{\boldsymbol{gr}(Y_s),\boldsymbol{gr}(Y_t)}$. In the non-overlapping scenario, the blocks of the $\boldsymbol{T}$ matrix are either filled with zeros or filled with the corresponding ζ_{Y_s,Y_t} values for the relationship between two PRVs Y_s and Y_t:

$$\boldsymbol{T}_{\boldsymbol{gr}(Y_s),\boldsymbol{gr}(Y_t)} = \zeta_{Y_s,Y_t} \cdot \mathbf{J}_{|\boldsymbol{D}(\boldsymbol{L}_s)|\times|\boldsymbol{D}(\boldsymbol{L}_t)|}, \tag{3.30}$$

where $\mathbf{J}_{|\boldsymbol{D}(\boldsymbol{L}_s)|\times|\boldsymbol{D}(\boldsymbol{L}_t)|}$ is the $|\boldsymbol{D}(\boldsymbol{L}_s)| \times |\boldsymbol{D}(\boldsymbol{L}_t)|$-dimensional all-ones matrix.

The diagonal blocks of $\boldsymbol{T}$ are always zero, because self-reference is not allowed in (parameterized) GBNs. This structure allows us to go blockwise through Expressions (3.26) to (3.29). For any $t \in \{1, ..., M\}$ and $\boldsymbol{s} = \{1, ..., t-1\}$, it holds that

$$\boldsymbol{\Sigma}_{\boldsymbol{gr}(Y_1),\boldsymbol{gr}(Y_1)} = \lambda_{Y_1}\mathbf{I}_{|\boldsymbol{D}(\boldsymbol{L}_1)|\times|\boldsymbol{D}(\boldsymbol{L}_1)|}, \tag{3.31}$$
$$\boldsymbol{\Sigma}_{\boldsymbol{gr}(\boldsymbol{Y_s}),\boldsymbol{gr}(Y_t)} = \boldsymbol{\Sigma}_{\boldsymbol{gr}(\boldsymbol{Y_s}),\boldsymbol{gr}(\boldsymbol{Y_s})}\boldsymbol{T}_{\boldsymbol{gr}(\boldsymbol{Y_s}),\boldsymbol{gr}(Y_t)}, \tag{3.32}$$
$$\boldsymbol{\Sigma}_{\boldsymbol{gr}(Y_t),\boldsymbol{gr}(\boldsymbol{Y_s})} = \boldsymbol{\Sigma}^T_{\boldsymbol{gr}(\boldsymbol{Y_s}),\boldsymbol{gr}(Y_t)}, \tag{3.33}$$
$$\boldsymbol{\Sigma}_{\boldsymbol{gr}(Y_t),\boldsymbol{gr}(Y_t)} = \boldsymbol{\Sigma}_{\boldsymbol{gr}(Y_t),\boldsymbol{gr}(\boldsymbol{Y_s})}\boldsymbol{T}_{\boldsymbol{gr}(\boldsymbol{Y_s}),\boldsymbol{gr}(Y_t)} + \lambda_{Y_t}\mathbf{I}_{|\boldsymbol{D}(\boldsymbol{L}_t)|\times|\boldsymbol{D}(\boldsymbol{L}_t)|}. \tag{3.34}$$

The initial block is always the identity matrix $\mathbf{I}$ multiplied with the node variance of the first PRV. We use an induction-like approach to show that the covariance matrix can be again calculated and stored in a lifted way. We show how the first iteration of Expressions (3.31) to (3.34) works out and then generalize further.

Expression (3.32) is a multiplication of the first block $\boldsymbol{\Sigma}_{\boldsymbol{gr}(Y_1),\boldsymbol{gr}(Y_1)}$ with the transition matrix block $\boldsymbol{T}_{\boldsymbol{gr}(Y_1),\boldsymbol{gr}(Y_2)}$

$$\begin{aligned}\mathbf{\Sigma}_{\boldsymbol{gr}(Y_1),\boldsymbol{gr}(Y_2)} &= \mathbf{\Sigma}_{\boldsymbol{gr}(Y_1),\boldsymbol{gr}(Y_1)}\boldsymbol{T}_{\boldsymbol{gr}(Y_1),\boldsymbol{gr}(Y_2)} \\ &= \lambda_{Y_1}\mathbf{I}_{|\boldsymbol{D}(\boldsymbol{L}_1)|\times|\boldsymbol{D}(\boldsymbol{L}_1)|}\zeta_{Y_1,Y_2}\mathbf{J}_{|\boldsymbol{D}(\boldsymbol{L}_1)|\times|\boldsymbol{D}(\boldsymbol{L}_2)|} \\ &= \lambda_{Y_1}\zeta_{Y_1,Y_2}\mathbf{J}_{|\boldsymbol{D}(\boldsymbol{L}_1)|\times|\boldsymbol{D}(\boldsymbol{L}_2)|},\end{aligned} \tag{3.35}$$

where the matrix multiplication with the identity matrix (as the neutral element) has no effect. The next block on the diagonal using Expression (3.34) is calculated by

$$\begin{aligned}\mathbf{\Sigma}_{\boldsymbol{gr}(Y_2),\boldsymbol{gr}(Y_2)} &= \mathbf{\Sigma}_{\boldsymbol{gr}(Y_2),\boldsymbol{gr}(Y_1)}\boldsymbol{T}_{\boldsymbol{gr}(Y_1),\boldsymbol{gr}(Y_2)} \\ &= \lambda_{Y_1}\zeta_{Y_1,Y_2}\mathbf{J}_{|\boldsymbol{D}(\boldsymbol{L}_2)|\times|\boldsymbol{D}(\boldsymbol{L}_1)|}\zeta_{Y_1,Y_2}\mathbf{J}_{|\boldsymbol{D}(\boldsymbol{L}_1)|\times|\boldsymbol{D}(\boldsymbol{L}_2)|} + \lambda_{Y_2}\mathbf{I}_{|\boldsymbol{D}(\boldsymbol{L}_2)|\times|\boldsymbol{D}(\boldsymbol{L}_2)|} \\ &= \lambda_{Y_1}\zeta^2_{Y_1,Y_2}|\boldsymbol{D}(\boldsymbol{L}_1)|\mathbf{J}_{|\boldsymbol{D}(\boldsymbol{L}_2)|\times|\boldsymbol{D}(\boldsymbol{L}_2)|} + \lambda_{Y_2}\mathbf{I}_{|\boldsymbol{D}(\boldsymbol{L}_2)|\times|\boldsymbol{D}(\boldsymbol{L}_2)|},\end{aligned} \tag{3.36}$$

where we use Lemma 3.3.1 in the multiplication of the two all-ones matrices.

In further steps, when calculating Expression (3.32) or Expression (3.34), there are several PRVs in $\boldsymbol{Y_s}$. With block matrix multiplication rules, we can calculate the resulting blocks $\mathbf{\Sigma}_{\boldsymbol{gr}(\boldsymbol{Y_s}),\boldsymbol{gr}(Y_t)}$ individually. We use $s \in 1, ..., t-1$ as an index to iterate over the blocks:

$$\mathbf{\Sigma}_{\boldsymbol{gr}(Y_s),\boldsymbol{gr}(Y_t)} = \sum_{k=1}^{t-1}\mathbf{\Sigma}_{\boldsymbol{gr}(Y_s),\boldsymbol{gr}(Y_k)}\boldsymbol{T}_{\boldsymbol{gr}(Y_k),\boldsymbol{gr}(Y_t)}. \tag{3.37}$$

Based on Lemma 3.3.2, we know that the resulting block $\mathbf{\Sigma}_{\boldsymbol{gr}(Y_s),\boldsymbol{gr}(Y_t)}$ can be expressed by a multiple of the all-ones matrix, if all $\mathbf{\Sigma}_{\boldsymbol{gr}(\boldsymbol{Y_s}),\boldsymbol{gr}(\boldsymbol{Y_s})}$ have a structure of $a\mathbf{J} + b\mathbf{I}$. For the off-diagonal blocks, we have shown that they follow this structure if their preceding blocks follow the structure as well. The formula for the on-diagonal blocks can also be converted to a summation:

$$\begin{aligned}\mathbf{\Sigma}_{\boldsymbol{gr}(Y_t),\boldsymbol{gr}(Y_t)} &= \mathbf{\Sigma}_{\boldsymbol{gr}(Y_t),\boldsymbol{gr}(\boldsymbol{Y_s})}\boldsymbol{T}_{\boldsymbol{gr}(\boldsymbol{Y_s}),\boldsymbol{gr}(Y_t)} + \lambda_{Y_t}\mathbf{I}_{|\boldsymbol{D}(\boldsymbol{L}_t)|\times|\boldsymbol{D}(\boldsymbol{L}_t)|} \\ &= \sum_{k=1}^{t-1}\mathbf{\Sigma}_{\boldsymbol{gr}(Y_t),\boldsymbol{gr}(Y_k)}\boldsymbol{T}_{\boldsymbol{gr}(Y_k),\boldsymbol{gr}(Y_t)} + \lambda_{Y_t}\mathbf{I}_{|\boldsymbol{D}(\boldsymbol{L}_t)|\times|\boldsymbol{D}(\boldsymbol{L}_t)|}.\end{aligned} \tag{3.38}$$

Based on Lemma 3.3.2, the summation over k results again in a multiple of the all-ones matrix. On the diagonal, there is the identity matrix addition. In this induction-like approach, we only need to show that the starting point is also following the structure, which is the case based on Expression (3.31). The beauty of this approach lies in the closed form of the matrix structure under the operations performed in the algorithm.

The reader might have wondered why we have put such an emphasis on the relatively simple Lemma 3.3.2, but finding a lifted structure that is closed under the operations performed in the algorithm by Shachter and Kenley (1989) will also be key for the following generalizations.

Closing the loop to the previous section, we can now store the factor of the all-ones matrix of each block in the $M \times M$ dimensional matrix $\boldsymbol{\rho}$ and the factor of the identity matrix, only present on the diagonal blocks, in the M dimensional vector $\boldsymbol{\lambda}$ to have a lifted representation of the joint covariance matrix. Additionally, we again store the domain size of each logvar in the cardinality vector $\boldsymbol{\tau}$. Grounding a full block can be done by

$$\boldsymbol{\Sigma}_{\boldsymbol{gr}(Y_s),\boldsymbol{gr}(Y_t)} = \rho_{Y_s,Y_t}\mathbf{J}_{\tau_s \times \tau_t} + \delta_{Y_s,Y_t}\lambda_s \mathbf{I}_{\tau_s \times \tau_s}. \tag{3.39}$$

To get a specific covariance between two ground randvars X_i and X_j, we select the entries of the block matrix in Expression (3.39) by

$$\Sigma_{X_i,X_j} = \rho_{lif(X_i),lif(X_j)} + \delta_{X_i,X_j}\lambda_{lif(X_i)}, \tag{3.40}$$

which is the same equation as Expression (3.13) for the non-matrix notation case. Let us revisit Example 3.3.1, but this time illustrate how calculations can be understood in terms of block matrix operations.

Example 3.3.2. Our ground matrix $\boldsymbol{T}$ has the following structure

$$\boldsymbol{T} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -2 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{\tau_P \times \tau_P} & \mathbf{0}_{\tau_P \times \tau_N} & -2\mathbf{J}_{\tau_P \times \tau_D} & \mathbf{0}_{\tau_P \times \tau_C} \\ \mathbf{0}_{\tau_N \times \tau_P} & \mathbf{0}_{\tau_N \times \tau_N} & \mathbf{0}_{\tau_N \times \tau_D} & 2\mathbf{J}_{\tau_N \times \tau_C} \\ \mathbf{0}_{\tau_D \times \tau_P} & \mathbf{0}_{\tau_D \times \tau_N} & \mathbf{0}_{\tau_D \times \tau_D} & 3\mathbf{J}_{\tau_D \times \tau_C} \\ \mathbf{0}_{\tau_C \times \tau_P} & \mathbf{0}_{\tau_C \times \tau_N} & \mathbf{0}_{\tau_C \times \tau_D} & \mathbf{0}_{\tau_C \times \tau_C} \end{bmatrix}$$

We will only show a few block matrix notations and will do repeat the full example here.

The full example can be found in Appendix A.2.

$$\begin{aligned}
\boldsymbol{\Sigma}_{\boldsymbol{gr}(H(P)),\boldsymbol{gr}(H(P))} &= \lambda_{H(P)}\mathbf{I}_{\tau_P\times\tau_P} = 1\mathbf{I}_{\tau_P\times\tau_P}\\
\boldsymbol{\Sigma}_{\boldsymbol{gr}(H(P)),\boldsymbol{gr}(T(N))} &= \boldsymbol{\Sigma}_{\boldsymbol{gr}(H(P)),\boldsymbol{gr}(H(P))}\boldsymbol{T}_{\boldsymbol{gr}(H(P)),\boldsymbol{gr}(T(N))} = \mathbf{I}_{\tau_P\times\tau_P}\mathbf{0}_{\tau_P\times\tau_N} = \mathbf{0}_{\tau_P\times\tau_N}\\
\boldsymbol{\Sigma}_{\boldsymbol{gr}(T(N)),\boldsymbol{gr}(T(N))} &= \boldsymbol{\Sigma}_{\boldsymbol{gr}(T(N)),\boldsymbol{gr}(H(P))}\boldsymbol{T}_{\boldsymbol{gr}(H(P)),\boldsymbol{gr}(T(N))} + \lambda_{T(N)}\mathbf{I}_{\tau_N\times\tau_N}\\
&= \mathbf{0}_{\tau_N\times\tau_P}\mathbf{0}_{\tau_N\times\tau_P} + \lambda_{T(N)}\mathbf{I}_{\tau_N\times\tau_N} = 0\mathbf{J}_{\tau_N\times\tau_N} + 2\mathbf{I}_{\tau_N\times\tau_N}\\
\boldsymbol{\Sigma}_{\boldsymbol{gr}(H(P)),\boldsymbol{gr}(W(D))} &= \boldsymbol{\Sigma}_{\boldsymbol{gr}(H(P)),\boldsymbol{gr}(H(P))}\boldsymbol{T}_{\boldsymbol{gr}(H(P)),\boldsymbol{gr}(W(D))}\\
&+ \boldsymbol{\Sigma}_{\boldsymbol{gr}(H(P)),\boldsymbol{gr}(T(N))}\boldsymbol{T}_{\boldsymbol{gr}(T(N)),\boldsymbol{gr}(W(D))}\\
&= 1\mathbf{I}_{\tau_P\times\tau_P}(-2)\mathbf{J}_{\tau_P\times\tau_D} + \mathbf{0}_{\tau_P\times\tau_N}\mathbf{0}_{\tau_N\times\tau_D} = (-2)\mathbf{J}_{\tau_P\times\tau_D}\\
\boldsymbol{\Sigma}_{\boldsymbol{gr}(T(N)),\boldsymbol{gr}(W(D))} &= \boldsymbol{\Sigma}_{\boldsymbol{gr}(T(N)),\boldsymbol{gr}(H(P))}\boldsymbol{T}_{\boldsymbol{gr}(H(P)),\boldsymbol{gr}(W(D))}\\
&+ \boldsymbol{\Sigma}_{\boldsymbol{gr}(T(N)),\boldsymbol{gr}(T(N))}\boldsymbol{T}_{\boldsymbol{gr}(T(N)),\boldsymbol{gr}(W(D))}\\
&= \mathbf{0}_{\tau_N\times\tau_P}(-2)\mathbf{J}_{\tau_P\times\tau_D} + 2\mathbf{I}_{\tau_N\times\tau_N}\mathbf{0}_{\tau_N\times\tau_D} = 0\mathbf{J}_{\tau_N\times\tau_D}\\
\boldsymbol{\Sigma}_{\boldsymbol{gr}(W(D)),\boldsymbol{gr}(W(D))} &= \boldsymbol{\Sigma}_{\boldsymbol{gr}(W(D)),\boldsymbol{gr}(H(P))}\boldsymbol{T}_{\boldsymbol{gr}(H(P)),\boldsymbol{gr}(W(D))}\\
&+ \boldsymbol{\Sigma}_{\boldsymbol{gr}(W(D)),\boldsymbol{gr}(T(N))}\boldsymbol{T}_{\boldsymbol{gr}(T(N)),\boldsymbol{gr}(W(D))}\\
&+ \lambda_{W(D)}\mathbf{I}_{\tau_D\times\tau_D}\\
&= (-2)\mathbf{J}_{\tau_D\times\tau_P}(-2)\mathbf{J}_{\tau_P\times\tau_D} + \mathbf{0}_{\tau_D\times\tau_N}\mathbf{0}_{\tau_N\times\tau_D} + \lambda_{W(D)}\mathbf{I}_{\tau_D\times\tau_D}\\
&= (-2)\cdot(-2)\cdot\tau_P\mathbf{J}_{\tau_D\times\tau_D} + \lambda_{W(D)}\mathbf{I}_{\tau_D\times\tau_D} = 8\mathbf{J}_{\tau_D\times\tau_D} + 3\mathbf{I}_{\tau_D\times\tau_D}
\end{aligned}$$

The example shows how using Lemma 3.3.1 and Lemma 3.3.2 results in the same $\boldsymbol{\rho}$ values as in Example 3.3.1.

We have introduced two approaches for constructing the lifted representation of the covariance matrix of $\boldsymbol{\rho}$ and $\boldsymbol{\lambda}$, allowing us to work with the blue part of our running example in Fig. 3.1. Next, we will focus on generalizing the matrix approach to allow overlaps between logvar sequences.

3.3.3 Allowing for Overlaps

Up until this point, the logvar sequences of connected PRVs are not allowed to have overlaps. However, as the orange box in the running example in Fig. 3.1 shows, overlaps are important to introduce further conditional independencies into the model and thus make the model more expressive. When allowing overlaps between the logvars of a parent and a child PRV, the connections on ground level only exist if the groundings share the same instances of the overlapping logvars. Referring to our previously discussed case, these additional independencies violate the equality assumption used in Expression (3.12) for the non-matrix notation and the structure of the $\boldsymbol{T}$ matrix in Expression (3.30) for the matrix notation. From here on, we will focus on the matrix notation. Therefore, we start with specifying the general form of the transition matrix $\boldsymbol{T}$ given overlaps. Afterwards, we construct a lifted representation of the covariance matrix anew.

General Form of the Transition Matrix

For the new lifted representation, we will need the Kronecker product of matrices.

Definition 3.3.1 (Kronecker product)**.** Given a $m \times n$ matrix $\boldsymbol{A}$ and a $p \times q$ matrix $\boldsymbol{B}$ the Kronecker product is defined as

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} A_{1,1}\boldsymbol{B} & \dots & A_{1,n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ A_{m,1}\boldsymbol{B} & \dots & A_{m,n}\boldsymbol{B} \end{bmatrix}. \tag{3.41}$$

We use the same $M \times M$ dimensional block structure for the transition matrix as introduced in the previous section. Given a parent PRV Y_s and a child PRV Y_t, a full logvar overlap, i.e., $\boldsymbol{L}_s = \boldsymbol{L}_t$, would mean that each randvar in $gr(Y_s)$ has exactly one connection to only one randvar in $gr(Y_t)$, namely, where Y_s and Y_t are grounded with the same constants. Given a global ordering, a full overlap results in a block design where the ζ value is only present on the diagonal:

$$\boldsymbol{T}_{\boldsymbol{gr}(Y_s),\boldsymbol{gr}(Y_t)} = \zeta_{Y_s,Y_t}\mathbf{I}_{|\boldsymbol{D}(\boldsymbol{L}_s)|\times|\boldsymbol{D}(\boldsymbol{L}_t)|}. \tag{3.42}$$

If the full overlap logvar sequence $\boldsymbol{L}_s = \boldsymbol{L}_t$ contains more than one logvar, we can reformulate Expression (3.42) into

$$\boldsymbol{T}_{\boldsymbol{gr}(Y_s),\boldsymbol{gr}(Y_t)} = \zeta_{Y_s,Y_t}\mathbf{I}_{|\boldsymbol{D}(\boldsymbol{L}_s)|\times|\boldsymbol{D}(\boldsymbol{L}_t)|} = \zeta_{Y_s,Y_t} \bigotimes_{L \in \boldsymbol{L}_s} \mathbf{I}_{|\boldsymbol{D}(L)|\times|\boldsymbol{D}(L)|}. \tag{3.43}$$

Adding any non-overlapping logvar L_p in the parent sequence implies that previously one randvar is now replaced by $|\boldsymbol{D}(L_p)|$ randvars, which leads to a Kronecker multiplication of a $\mathbf{J}^{|\boldsymbol{D}(L_p)|\times 1}$ column vector (parent variables correspond to rows). Analogously, a non-overlapping additional logvar L_c in the child PRV results in a Kronecker multiplication of a $\mathbf{J}^{1\times|\boldsymbol{D}(L_c)|}$ row vector. Depending on the global ordering of the logvars, we get the following general formula for $\boldsymbol{T}$, with $\mathbf{J}^0 = \mathbf{I}$:

$$\boldsymbol{T}_{\boldsymbol{gr}(Y_s),\boldsymbol{gr}(Y_t)} = \zeta_{Y_s,Y_t} \bigotimes_{L \in seq(\boldsymbol{L}_s,\boldsymbol{L}_t)} \mathbf{J}^{ov(L,\boldsymbol{L}_s,\boldsymbol{L}_t)}_{dim(L,\boldsymbol{L}_s)\times dim(L,\boldsymbol{L}_t)}. \tag{3.44}$$

The *ov*-function in Expression (3.44) controls if an identity matrix (overlap) or an all-ones matrix (non-overlap) is needed. For the non-overlapping logvars, the Kronecker product contains either a column or row vector dependent on the logvar being in the child or parent logvar sequence.

Example 3.3.3. For the overlapping part (in orange) of our example in Fig. 3.1, the transition matrix $\boldsymbol{T}$ has following the block structure

$$\boldsymbol{T} = \begin{bmatrix} \mathbf{0}_{|\boldsymbol{D}(M)|\times|\boldsymbol{D}(M)|} & \mathbf{0}_{|\boldsymbol{D}(M)|\times|\boldsymbol{D}(P)|} & \boldsymbol{T}_{\boldsymbol{gr}(E(M)),\boldsymbol{gr}(I(M,P)} & \mathbf{0}_{|\boldsymbol{D}(M)|\times|\boldsymbol{D}(P)|} \\ \mathbf{0}_{|\boldsymbol{D}(P)|\times|\boldsymbol{D}(M)|} & \mathbf{0}_{|\boldsymbol{D}(P)|\times|\boldsymbol{D}(P)|} & \boldsymbol{T}_{\boldsymbol{gr}(S(P)),\boldsymbol{gr}(I(M,P)} & \boldsymbol{T}_{\boldsymbol{gr}(S(P)),\boldsymbol{gr}(H(P))} \\ \mathbf{0}_{|\boldsymbol{D}(M,P)|\times|\boldsymbol{D}(M)|} & \mathbf{0}_{|\boldsymbol{D}(M,P)|\times|\boldsymbol{D}(P)|} & \mathbf{0}_{|\boldsymbol{D}(M,P)|\times|\boldsymbol{D}(M,P)|} & \boldsymbol{T}_{\boldsymbol{gr}(I(M,P)),\boldsymbol{gr}(H(P))} \\ \mathbf{0}_{|\boldsymbol{D}(P)|\times|\boldsymbol{D}(M)|} & \mathbf{0}_{|\boldsymbol{D}(P)|\times|\boldsymbol{D}(P)|} & \mathbf{0}_{|\boldsymbol{D}(P)|\times|\boldsymbol{D}(M,P)|} & \mathbf{0}_{|\boldsymbol{D}(P)|\times|\boldsymbol{D}(P)|} \end{bmatrix},$$

with

$$\begin{aligned} \boldsymbol{T}_{\boldsymbol{gr}(E(M)),\boldsymbol{gr}(I(M,P)} &= \zeta_{E(M),I(M,P)}\mathbf{I}_{|\boldsymbol{D}(M)|\times|\boldsymbol{D}(M)|}\mathbf{J}_{1\times|\boldsymbol{D}(P)|}, \\ \boldsymbol{T}_{\boldsymbol{gr}(S(P)),\boldsymbol{gr}(I(M,P)} &= \zeta_{S(P),I(M,P)}\mathbf{J}_{1\times|\boldsymbol{D}(M)|}\mathbf{I}_{|\boldsymbol{D}(P)|\times|\boldsymbol{D}(P)|}, \\ \boldsymbol{T}_{\boldsymbol{gr}(S(P)),\boldsymbol{gr}(H(P))} &= \zeta_{S(P),H(P)}\mathbf{I}_{|\boldsymbol{D}(P)|\times|\boldsymbol{D}(P)|} \text{ and} \\ \boldsymbol{T}_{\boldsymbol{gr}(I(M,P)),\boldsymbol{gr}(H(P))} &= \zeta_{I(M,P),H(P)}\mathbf{J}_{|\boldsymbol{D}(P)|\times 1}\mathbf{I}_{|\boldsymbol{D}(P)|\times|\boldsymbol{D}(P)|}. \end{aligned}$$

Since Expression (3.44) gives us a fixed rule to create a ground version of the transition matrix $\boldsymbol{T}$, we can store all information needed in the cardinality vector $\boldsymbol{\tau}$, containing the domain sizes of the logvars, and the lifted transition matrix $\boldsymbol{\zeta}$.

Constructing the Covariance Matrix

The new general form allowing for the transition matrix $\boldsymbol{T}$ based on a lifted representation $\boldsymbol{\zeta}$ allows for overlaps between logvar sequences. We investigate how this new structure of $\boldsymbol{T}$ influences Expressions (3.31) to (3.34). Analogously to the non-overlapping case, we use an induction-like approach to show that all blocks follow a fixed structure, which enables a lifted storage of the covariance matrix $\boldsymbol{\Sigma}$. For ease of argumentation, we need the concept of a Kronecker sequence and Kronecker factors.

Definition 3.3.2 (Kronecker sequence, Kronecker factor)**.** Given a logvar sequence $\boldsymbol{L}$, we define a *Kronecker sequence* as any Kronecker product sequence

$$\bigotimes_{L\in\boldsymbol{L}} \boldsymbol{H}_{|\boldsymbol{D}(L)|\times|\boldsymbol{D}(L)|}, \tag{3.45}$$

where the matrix $\boldsymbol{H}$ is either an identity matrix $\mathbf{I}$ or an all-ones matrix $\mathbf{J}$. We call every factor $\boldsymbol{H}_{|\boldsymbol{D}(L)|\times|\boldsymbol{D}(L)|}$ in the Kronecker sequence a *Kronecker factor*.

For a logvar sequence $\boldsymbol{L}_s$, there are $2^{|\boldsymbol{L}_s|}$ different Kronecker sequences following Definition 3.3.2, because for each logvar $L \in \boldsymbol{L}_s$, there are two possible Kronecker factors ($\mathbf{I}_{|\boldsymbol{D}(L)|\times|\boldsymbol{D}(L)|}$ and $\mathbf{J}_{|\boldsymbol{D}(L)|\times|\boldsymbol{D}(L)|}$). To identify one of the possible Kronecker sequences, we use a $|\boldsymbol{L}_s|$-dimensional index vector of zeroes and ones, where a zero stands for an identity matrix and one for an all-ones matrix, and $\mathbf{J}$ as basis with $\mathbf{J}^0 = \mathbf{I}$ and $\mathbf{J}^1 = \mathbf{J}$. For example, the vector $\mathbf{0} = (0 \dots 0)$ references the Kronecker sequence of only identity matrices, whereas $\mathbf{1} = (1 \dots 1)$ references the Kronecker sequence of only all-ones matrices. Each position in the vector corresponds to one logvar $L \in \boldsymbol{L}_s$ and thus to one Kronecker factor. The set of possible vectors is referenced by $\boldsymbol{\Phi}_s$. To iterate over all possible Kronecker sequences, we iterate over all possible index vectors, i.e., combinations of zeroes and ones, denoted as $\boldsymbol{q}_s \in \boldsymbol{\Phi}_s$. To iterate over all possible Kronecker sequences of an overlapping logvar sequence $\boldsymbol{L}_s \cap \boldsymbol{L}_t$ in a logvar sequence $\boldsymbol{L}_s \cup \boldsymbol{L}_t$, we write $\boldsymbol{q}_{s,t} \in \boldsymbol{\Phi}_{s,t}$ for short, with $\boldsymbol{q}_{s,t}$ being padded with ones at those positions that do not correspond to overlapping logvars. That means that the set $\boldsymbol{\Phi}_{s,t}$ for two non-overlapping logvar sequences only contains a single vector of ones.

Lemma 3.3.3. *Given a transition matrix $\boldsymbol{T}$ with a structure from Expression* (3.44) *and the inductive application of Expressions* (3.31) *to* (3.34)*, the diagonal blocks of the covariance matrix $\boldsymbol{\Sigma}$ follow the structure*

$$\boldsymbol{\Sigma}_{\boldsymbol{gr}(Y_s,Y_s)} = \sum_{\boldsymbol{q}_s \in \boldsymbol{\Phi}_s} \rho_{s,s}^{\boldsymbol{q}_s} \bigotimes_{L \in \boldsymbol{L}_s} \mathbf{J}_{|\boldsymbol{D}(L)| \times |\boldsymbol{D}(L)|}^{\pi_L(\boldsymbol{q}_s)}, \tag{3.46}$$

and off-diagonal blocks of the covariance matrix $\boldsymbol{\Sigma}$ follow

$$\boldsymbol{\Sigma}_{\boldsymbol{gr}(Y_s,Y_t)} = \sum_{\boldsymbol{q}_{s,t} \in \boldsymbol{\Phi}_{s,t}} \rho_{s,t}^{\boldsymbol{q}_{s,t}} \bigotimes_{L \in \boldsymbol{L}_s \cup \boldsymbol{L}_t} \mathbf{J}_{dim(L,\boldsymbol{L}_s) \times dim(L,\boldsymbol{L}_t)}^{qexp(\boldsymbol{q}_{s,t},L)}, \tag{3.47}$$

with

$$qexp(\boldsymbol{q}_{s,t}, L) = \begin{cases} \pi_L(\boldsymbol{q}_{s,t}) & \text{if } L \in (\boldsymbol{L}_s \cup \boldsymbol{L}_t), \\ 1 & \text{otherwise.} \end{cases} \tag{3.48}$$

The tensor $\boldsymbol{\rho}$ contains $M \cdot M$ vectors, with each vector $\boldsymbol{\rho}_{s,t}$ having a length of $2^{|\boldsymbol{D}(\boldsymbol{L}_s \cap \boldsymbol{L}_t)|}$. Each of the values can be indexed with one vector $\boldsymbol{q}_{s,t}$ referring to the corresponding Kronecker sequence.

Proof. We use an induction proof. First, we assume that Lemma 3.3.3 is true for any current block structure of the covariance matrix. Then we show that all following blocks of the covariance matrix using the condition from Lemma 3.3.3 stay in the format. Last, we show that Expression (3.31) is a valid starting point that fulfils the structure as well.

Expression (3.46) is a special case of Expression (3.47) as both logvar sequences involved are equal, i.e., $\boldsymbol{q}_{s,t} = \boldsymbol{q}_s = \boldsymbol{q}_t$ ($\boldsymbol{L}_s = \boldsymbol{L}_t$), and *dim* and *qexp* simplifying to their first cases.

Inserting the general equation for a block from the transition matrix from Expression (3.44) and the equation for a block from the covariance matrix from Expression (3.47) into the equation for calculating a new block in the covariance matrix from Expression (3.37) for one arbitrary c k results in

$$\boldsymbol{\Sigma}_{\boldsymbol{gr}(Y_s),\boldsymbol{gr}(Y_k)}\boldsymbol{T}_{\boldsymbol{gr}(Y_k),\boldsymbol{gr}(Y_t)} = \sum_{\boldsymbol{q}_{s,k}\in\boldsymbol{\Phi}_{s,k}} \left(\rho_{s,k}^{\boldsymbol{q}_{s,k}} \bigotimes_{L\in\boldsymbol{L}_s\cup\boldsymbol{L}_k} \mathbf{J}_{dim(L,\boldsymbol{L}_s)\times dim(L,\boldsymbol{L}_k)}^{qexp(\boldsymbol{q}_{s,k},L)}\right)\left(\zeta_{Y_k,Y_t} \bigotimes_{L\in\boldsymbol{L}_k\cup\boldsymbol{L}_t} \mathbf{J}_{dim(L,\boldsymbol{L}_k)\times dim(L,\boldsymbol{L}_t)}^{ov(L,\boldsymbol{L}_k,\boldsymbol{L}_t)}\right) \tag{3.49}$$

Multiplying sequences of Kronecker products can be rewritten using the mixed-product property in its general form (Broxson, 2006):

$$(A_1 \otimes A_2 \otimes ... \otimes A_n)(B_1 \otimes B_2 \otimes ... \otimes B_n) = (A_1B_1 \otimes A_2B_2 \otimes ... \otimes A_nB_n). \tag{3.50}$$

We use Expression (3.50) to align the Kronecker factors that correspond to the same logvar. For each logvar L (and thus Kronecker factor), there are thus seven possible Kronecker factor combinations resulting from the occurrence of the logvar in the sequences $\boldsymbol{L}_s$, $\boldsymbol{L}_k$, and $\boldsymbol{L}_t$:

1. $L \in \boldsymbol{L}_s$, $L \notin \boldsymbol{L}_t$ and $L \in \boldsymbol{L}_k$, result: $\mathbf{I}_{|\boldsymbol{D}(L)|\times|\boldsymbol{D}(L)|}\mathbf{J}_{|\boldsymbol{D}(L)|\times 1} = \mathbf{J}_{|\boldsymbol{D}(L)|\times 1}$
2. $L \in \boldsymbol{L}_s$, $L \notin \boldsymbol{L}_t$ and $L \notin \boldsymbol{L}_k$, result: $\mathbf{I}_{|\boldsymbol{D}(L)|\times|\boldsymbol{D}(L)|}\mathbf{J}_{|\boldsymbol{D}(L)|\times 1} = \mathbf{J}_{|\boldsymbol{D}(L)|\times 1}$
3. $L \notin \boldsymbol{L}_s$, $L \in \boldsymbol{L}_t$ and $L \in \boldsymbol{L}_k$, result: $\mathbf{J}_{1\times|\boldsymbol{D}(L)|}\mathbf{I}_{|\boldsymbol{D}(L)|\times|\boldsymbol{D}(L)|} = \mathbf{J}_{1\times|\boldsymbol{D}(L)|}$
4. $L \notin \boldsymbol{L}_s$, $L \in \boldsymbol{L}_t$ and $L \notin \boldsymbol{L}_k$, result: $1 \cdot \mathbf{J}_{1\times|\boldsymbol{D}(L)|} = \mathbf{J}_{1\times|\boldsymbol{D}(L)|}$
5. $L \notin \boldsymbol{L}_s$, $L \notin \boldsymbol{L}_t$ and $L \in \boldsymbol{L}_k$, result: $\mathbf{J}_{1\times|\boldsymbol{D}(L)|}\mathbf{J}_{|\boldsymbol{D}(L)|\times 1} = |\boldsymbol{D}(L)|$
6. $L \in \boldsymbol{L}_s$, $L \in \boldsymbol{L}_t$ and $L \in \boldsymbol{L}_k$, result: $\mathbf{I}_{|\boldsymbol{D}(L)|}\mathbf{I}_{|\boldsymbol{D}(L)|\times|\boldsymbol{D}(L)|} = \mathbf{I}_{|\boldsymbol{D}(L)|\times|\boldsymbol{D}(L)|}$
7. $L \in \boldsymbol{L}_s$, $L \in \boldsymbol{L}_t$ and $L \notin \boldsymbol{L}_k$, result: $\mathbf{J}_{|\boldsymbol{D}(L)|\times 1}\mathbf{J}_{1\times|\boldsymbol{D}(L)|} = \mathbf{J}_{|\boldsymbol{D}(L)|\times|\boldsymbol{D}(L)|}$

The structure in Expression (3.49) defines that only permutations exist for logvars in the overlap between $\boldsymbol{L}_s$ and $\boldsymbol{L}_t$. The seven cases confirm this structure. Cases 1 and 2 result in the same Kronecker factor independently of L_k. The same holds for Cases 3 and 4. Case 5 adds a $|\boldsymbol{D}(L)|$ factor to all summands but does not affect the structure. Cases 6 and 7 are the only cases where L is part of the overlap between $\boldsymbol{L}_S$ and $\boldsymbol{L}_k$. Here, the structure is dependent on L being in $\boldsymbol{L}_k$, but as said L will be also part in the permutation anyway. Each result is stored in the $\boldsymbol{\rho}$ vector at the position corresponding to the Kronecker sequence. Every entry of the new $\boldsymbol{\rho}^{(i,j)}$ vector can be calculated as follows:

$$\rho_{s,t}^{\boldsymbol{q}_{s,t}} = \sum_{n=1}^{t-1} \sum_{\boldsymbol{q}\in\sigma_{\boldsymbol{q_{s,t}}}(\boldsymbol{Q_{s,n}})} \rho_{s,n}^{\boldsymbol{q}}\zeta_{n,t} \prod_{L\in(\boldsymbol{L}_n\setminus\boldsymbol{L}_t)} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})}, \tag{3.51}$$

where the selection in the second sum is controlling that the results are stored in the ρ-entry corresponding to the correct Kronecker sequence.

The transpose calculation in Expression (3.33) stays in the same structure implied by Expression (3.47):

$$\begin{aligned}
&\boldsymbol{\Sigma}^T_{\boldsymbol{gr}(Y_s,Y_t)}\\
&=\left(\sum_{\boldsymbol{q}_{s,t}\in\boldsymbol{\Phi}_{s,t}} \rho_{s,t}^{\boldsymbol{q}_{s,t}} \bigotimes_{L\in seq(\boldsymbol{L}_s,\boldsymbol{L}_t)} \mathbf{J}^{qexp(\boldsymbol{q}_{s,t},L)}_{dim(L,\boldsymbol{L}_s)\times dim(L,\boldsymbol{L}_t)}\right)^T\\
&=\sum_{\boldsymbol{q}_{s,t}\in\boldsymbol{\Phi}_{s,t}} \rho_{s,t}^{\boldsymbol{q}_{s,t}} \bigotimes_{L\in seq(\boldsymbol{L}_s,\boldsymbol{L}_t)} \mathbf{J}^{qexp(\boldsymbol{q}_{s,t},L)}_{dim(L,\boldsymbol{L}_t)\times dim(L,\boldsymbol{L}_s)}\\
&=\boldsymbol{\Sigma}_{\boldsymbol{gr}(Y_t,Y_s)}.
\end{aligned}\tag{3.52}$$

Based on the equation for calculating a new off-diagonal block, we apply Expression (3.34) to calculate a new on-diagonal block. The only difference to the off-diagonal case is that we also add the PRV variance λ_{Y_t} to the diagonal after calculating the $\boldsymbol{\rho}$ vector using Expression (3.51), meaning, we add the PRV variance λ_{Y_t} to the one summand indexed by $\mathbf{0}$:

$$\rho^{\mathbf{0}}_{t,t} \leftarrow \rho^{\mathbf{0}}_{t,t} + \lambda_{Y_t}. \tag{3.53}$$

The last part of the proof is to show that the starting point is in line with our assumptions. Based on Expression (3.31), the vector $\boldsymbol{\rho}_{Y_1,Y_1}$ contains the value λ_{Y_1} at position $\boldsymbol{\rho}^{\mathbf{0}}_{Y_1,Y_1}$. This position is corresponding to the Kronecker sequence full of identity matrices. All other vector entries corresponding to different Kronecker sequences are zero. This structure serves at as a valid starting point. We began the proof by suggesting a closed structure for the blocks in the covariance matrix. We have shown that an inductive creation of the covariance matrix keeps the structure and have formulated a valid starting point. Thus we have a valid lifted representation together with lifted operations for constructing and storing the covariance matrix. □

The proof for Lemma 3.3.3 has also equipped us with an expression to calculate the lifted covariance matrix, stored in the $\boldsymbol{\rho}$ tensor containing $M \cdot M$ lifted vectors, where each vector $\boldsymbol{\rho}_{s,t}$ has a dimensionality of $2^{|\boldsymbol{L}_s\cap\boldsymbol{L}_t|}$. The lemma enables us to calculate the lifted joint distribution for any parameterized GBN and store it in a set of lifted variables. Remark: In the overlapping case, $\boldsymbol{\rho}$ is a tensor, which reduces to a matrix in the non-overlapping case.

Table 3.1 contains an overview of the ground and lifted way to store the joint distribution. The dimensionality of the $\boldsymbol{\rho}$ is the worst case, only occurring if there are logvar sequence overlaps involving all logvars in $\mathcal{L}$.

We have now described an algorithm for constructing a lifted version of the covariance matrix of the joint distribution. The lifted mean vector and the lifted version of the

Table 3.1: Variables in ground in and lifted representation of the joint distribution

	Grounded		**Lifted**	
	Variable	Dimensionality	Variable	Dimensionality
Randvars/PRVs	$\mathcal{X}$	N	$\mathcal{Y}$	M
Covariance	$\boldsymbol{\Sigma}$	$N \times N$	$\boldsymbol{\rho}$	$M \times M \times 2^{\lvert\mathcal{L}\rvert}$
Mean	$\boldsymbol{\mu}$	N	$\boldsymbol{\eta}$	M
Domain sizes		implicit	$\boldsymbol{\tau}$	$\lvert\mathcal{L}\rvert$

covariance matrix together form the *lifted joint distribution.* For better reference of the covariance structure described by Expression (3.47) in later calculations, we define terms to refer to the elements of the structure:

Definition 3.3.3 (Kronecker component block, Kronecker permutation set)**.** We define a *Kronecker component block* as a matrix that follows the structure of Expression (3.47). We call all possible Kronecker sequences occurring in the Kronecker component block the *Kronecker permutation set.*

Before moving on the defining matrix operations for the lifted joint distribution, we look at an example. Example 3.3.4 describes constructing the lifted covariance matrix in the overlapping scenario visualized in the orange box of Fig. 3.1.

Example 3.3.4. We build upon the transition matrix $\boldsymbol{T}$ set up in Example 3.3.3. In the formulas, we assume that the index variables $\boldsymbol{q}$ have an implicit connection to the logvars they represent. When writing the values of for example $\boldsymbol{q}_{s,t}$, the link to $\boldsymbol{L}_s$ and $\boldsymbol{L}_t$ is lost. Therefore, we subscript the 0 and 1 values with the logvar they are connected to. Setting up the first element of the lifted covariance matrix is straight forward:

$$\boldsymbol{\rho}_{E(M),E(M)} = \begin{pmatrix} \rho_{1,1}^{0_M} \\ \rho_{1,1}^{1_M} \end{pmatrix} = \begin{pmatrix} \lambda_{E(M)} \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Next, we calculate our first off-diagonal entry based on Expression (3.51):

$$\begin{aligned} \boldsymbol{\rho}_{E(M),S(P)} = \begin{pmatrix} \rho_{1,2}^{1_M 1_P} \end{pmatrix} &= \left(\sum_{\boldsymbol{q} \in \sigma_{\boldsymbol{q}_{s,t}}(\boldsymbol{Q}_{s,s})} \rho_{s,s}^{\boldsymbol{q}} \zeta_{s,t} \prod_{L \in (\boldsymbol{L}_s \setminus \boldsymbol{L}_t)} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \right) \\ &= \left((\rho_{1,1}^{0_M} \cdot 0) + (\rho_{1,1}^{1_M} \cdot 0) \right) = \begin{pmatrix} 0 \end{pmatrix}. \end{aligned}$$

With $s = t = pos(S(P)) = 2$, the second on-diagonal block is built by

$$
\begin{aligned}
\boldsymbol{\rho}_{S(P),S(P)} = \begin{pmatrix} \rho_{2,2}^{0_P} \\ \rho_{2,2}^{1_P} \end{pmatrix} &= \begin{pmatrix} \sum_{n=1}^{t-1} \sum\limits_{\boldsymbol{q} \in \sigma_{\boldsymbol{q_{s,t}}}(\boldsymbol{Q_{s,n}})} \rho_{s,n}^{\boldsymbol{q}} \zeta_{n,t} \prod\limits_{L \in (\boldsymbol{L}_n \setminus \boldsymbol{L}_t)} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \\ \sum_{n=1}^{t-1} \sum\limits_{\boldsymbol{q} \in \sigma_{\boldsymbol{q_{s,t}}}(\boldsymbol{Q_{s,n}})} \rho_{s,n}^{\boldsymbol{q}} \zeta_{n,t} \prod\limits_{L \in (\boldsymbol{L}_n \setminus \boldsymbol{L}_t)} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \end{pmatrix} + \begin{pmatrix} \lambda_{S(P)} \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \sum\limits_{\boldsymbol{q} \in \sigma_{0_P}(\{1_M 1_P\})} \rho_{2,1}^{\boldsymbol{q}} \zeta_{1,2} \prod\limits_{L \in \{M\}} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \\ \sum\limits_{\boldsymbol{q} \in \sigma_{1_P}(\{1_M 1_P\})} \rho_{2,1}^{\boldsymbol{q}} \zeta_{1,2} \prod\limits_{L \in \{M\}} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \end{pmatrix} + \begin{pmatrix} \lambda_{S(P)} \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 + \lambda_{S(P)} \\ \rho_{1,2}^{1_M 1_P} \cdot \zeta_{1,2} \cdot |\boldsymbol{D}(M)|^{\pi_M(1_M 1_P)} \end{pmatrix} = \begin{pmatrix} \lambda_{S(P)} \\ 0 \cdot 0 \cdot 3 \end{pmatrix} = \begin{pmatrix} \lambda_{S(P)} \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}.
\end{aligned}
$$

This result is not surprising, because $\boldsymbol{\rho}_{1,1}$ has the same structure as $\boldsymbol{\rho}_{2,2}$ and in the topological ordering both could be exchanged, because both have no parents. The covariance between $E(M)$ and $I(M,P)$ is the first non-trivial example. Here, $s = pos(E(M)) = 1$ and $t = pos(I(M,P)) = 3$:

$$
\begin{aligned}
\boldsymbol{\rho}_{1,3} = \begin{pmatrix} \rho_{1,3}^{0_M 1_P} \\ \rho_{1,3}^{1_M 1_P} \end{pmatrix} &= \begin{pmatrix} \sum_{n=1}^{t-1} \sum\limits_{\boldsymbol{q} \in \sigma_{\boldsymbol{q_{s,t}}}(\boldsymbol{Q_{s,n}})} \rho_{s,n}^{\boldsymbol{q}} \zeta_{n,t} \prod\limits_{L \in (\boldsymbol{L}_n \setminus \boldsymbol{L}_t)} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \\ \sum_{n=1}^{t-1} \sum\limits_{\boldsymbol{q} \in \sigma_{\boldsymbol{q_{s,t}}}(\boldsymbol{Q_{s,n}})} \rho_{s,n}^{\boldsymbol{q}} \zeta_{n,t} \prod\limits_{L \in (\boldsymbol{L}_n \setminus \boldsymbol{L}_t)} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \end{pmatrix} \\
&= \begin{pmatrix} \sum\limits_{\boldsymbol{q} \in \sigma_{0_M 1_P}(\{0_M, 1_M\})} \rho_{1,1}^{\boldsymbol{q}} \zeta_{1,3} \prod\limits_{L \in \emptyset} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} + \sum\limits_{\boldsymbol{q} \in \sigma_{0_M 1_P}(\{1_M 1_P\})} \rho_{1,2}^{\boldsymbol{q}} \zeta_{2,3} \prod\limits_{L \in \emptyset} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \\ \sum\limits_{\boldsymbol{q} \in \sigma_{1_M 1_P}(\{0_M, 1_M\})} \rho_{1,1}^{\boldsymbol{q}} \zeta_{1,3} \prod\limits_{L \in \emptyset} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} + \sum\limits_{\boldsymbol{q} \in \sigma_{1_M 1_P}(\{1_M 1_P\})} \rho_{1,2}^{\boldsymbol{q}} \zeta_{2,3} \prod\limits_{L \in \emptyset} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \end{pmatrix} \\
&= \begin{pmatrix} \rho_{1,1}^{0_M} \zeta_{1,3} \\ \rho_{1,1}^{1_M} \zeta_{1,3} + \rho_{1,2}^{1_M 1_P} \zeta_{2,3} \end{pmatrix} = \begin{pmatrix} 1 \cdot 2 \\ 0 \cdot 2 + 0 \cdot 5 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}.
\end{aligned}
$$

For $s = pos(S(P)) = 2$ and $t = pos(I(M,P)) = 3$, the block is

$$\boldsymbol{\rho}_{2,3} = \begin{pmatrix} \rho_{2,3}^{1_M 0_P} \\ \rho_{2,3}^{1_M 1_P} \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^{t-1} \sum\limits_{\boldsymbol{q} \in \sigma_{\boldsymbol{q_{s,t}}}(\boldsymbol{Q_{s,n}})} \rho_{s,n}^{\boldsymbol{q}} \zeta_{n,t} \prod\limits_{L \in (\boldsymbol{L}_n \setminus \boldsymbol{L}_t)} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \\ \sum_{n=1}^{t-1} \sum\limits_{\boldsymbol{q} \in \sigma_{\boldsymbol{q_{s,t}}}(\boldsymbol{Q_{s,n}})} \rho_{s,n}^{\boldsymbol{q}} \zeta_{n,t} \prod\limits_{L \in (\boldsymbol{L}_n \setminus \boldsymbol{L}_t)} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \end{pmatrix}$$

$$= \begin{pmatrix} \sum\limits_{\boldsymbol{q} \in \sigma_{1_M 0_P}(\{1_M 1_P\})} \rho_{2,1}^{\boldsymbol{q}} \zeta_{1,3} \prod\limits_{L \in \emptyset} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} + \sum\limits_{\boldsymbol{q} \in \sigma_{1_M 0_P}(\{0_P, 1_P\})} \rho_{2,2}^{\boldsymbol{q}} \zeta_{2,3} \prod\limits_{L \in \emptyset} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \\ \sum\limits_{\boldsymbol{q} \in \sigma_{1_M 1_P}(\{1_M 1_P\})} \rho_{2,1}^{\boldsymbol{q}} \zeta_{1,3} \prod\limits_{L \in \emptyset} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} + \sum\limits_{\boldsymbol{q} \in \sigma_{1_M 1_P}(\{0_P, 1_P\})} \rho_{2,2}^{\boldsymbol{q}} \zeta_{2,3} \prod\limits_{L \in \emptyset} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \end{pmatrix}$$

$$= \begin{pmatrix} \rho_{2,2}^{0_P} \zeta_{2,3} \\ \rho_{2,1}^{1_M 1_P} \zeta_{1,3} + \rho_{2,2}^{1_P} \zeta_{2,3} \end{pmatrix} = \begin{pmatrix} 2 \cdot 5 \\ 0 \cdot 2 + 0 \cdot 5 \end{pmatrix} = \begin{pmatrix} 10 \\ 0 \end{pmatrix}.$$

For $s = pos(I(M,P)) = 3$ and $t = pos(I(M,P)) = 3$, we omit the product in the second step, because the set $\boldsymbol{L}_n \setminus \boldsymbol{L}_t$ is always empty, and get

$$\boldsymbol{\rho}_{3,3} = \begin{pmatrix} \rho_{3,3}^{0_M 0_P} \\ \rho_{3,3}^{0_M 1_P} \\ \rho_{3,3}^{1_M 0_P} \\ \rho_{3,3}^{1_M 1_P} \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^{t-1} \sum\limits_{\boldsymbol{q} \in \sigma_{\boldsymbol{q_{s,t}}}(\boldsymbol{Q_{s,n}})} \rho_{s,n}^{\boldsymbol{q}} \zeta_{n,t} \prod\limits_{L \in (\boldsymbol{L}_n \setminus \boldsymbol{L}_t)} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \\ \sum_{n=1}^{t-1} \sum\limits_{\boldsymbol{q} \in \sigma_{\boldsymbol{q_{s,t}}}(\boldsymbol{Q_{s,n}})} \rho_{s,n}^{\boldsymbol{q}} \zeta_{n,t} \prod\limits_{L \in (\boldsymbol{L}_n \setminus \boldsymbol{L}_t)} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \\ \sum_{n=1}^{t-1} \sum\limits_{\boldsymbol{q} \in \sigma_{\boldsymbol{q_{s,t}}}(\boldsymbol{Q_{s,n}})} \rho_{s,n}^{\boldsymbol{q}} \zeta_{n,t} \prod\limits_{L \in (\boldsymbol{L}_n \setminus \boldsymbol{L}_t)} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \\ \sum_{n=1}^{t-1} \sum\limits_{\boldsymbol{q} \in \sigma_{\boldsymbol{q_{s,t}}}(\boldsymbol{Q_{s,n}})} \rho_{s,n}^{\boldsymbol{q}} \zeta_{n,t} \prod\limits_{L \in (\boldsymbol{L}_n \setminus \boldsymbol{L}_t)} |\boldsymbol{D}(L)|^{\pi_L(\boldsymbol{q})} \end{pmatrix} + \begin{pmatrix} \lambda_{I(M,P)} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} \sum\limits_{\boldsymbol{q} \in \sigma_{0_M 0_P}(\{0_M 1_P, 1_M 1_P\})} \rho_{3,1}^{\boldsymbol{q}} \zeta_{1,3} + \sum\limits_{\boldsymbol{q} \in \sigma_{0_M 0_P}(\{1_M 0_P, 1_M 1_P\})} \rho_{3,2}^{\boldsymbol{q}} \zeta_{2,3} \\ \sum\limits_{\boldsymbol{q} \in \sigma_{0_M 1_P}(\{0_M 1_P, 1_M 1_P\})} \rho_{3,1}^{\boldsymbol{q}} \zeta_{1,3} + \sum\limits_{\boldsymbol{q} \in \sigma_{0_M 1_P}(\{1_M 0_P, 1_M 1_P\})} \rho_{3,2}^{\boldsymbol{q}} \zeta_{2,3} \\ \sum\limits_{\boldsymbol{q} \in \sigma_{1_M 0_P}(\{0_M 1_P, 1_M 1_P\})} \rho_{3,1}^{\boldsymbol{q}} \zeta_{1,3} + \sum\limits_{\boldsymbol{q} \in \sigma_{1_M 0_P}(\{1_M 0_P, 1_M 1_P\})} \rho_{3,2}^{\boldsymbol{q}} \zeta_{2,3} \\ \sum\limits_{\boldsymbol{q} \in \sigma_{1_M 1_P}(\{0_M 1_P, 1_M 1_P\})} \rho_{3,1}^{\boldsymbol{q}} \zeta_{1,3} + \sum\limits_{\boldsymbol{q} \in \sigma_{1_M 1_P}(\{1_M 0_P, 1_M 1_P\})} \rho_{3,2}^{\boldsymbol{q}} \zeta_{2,3} \end{pmatrix} + \begin{pmatrix} \lambda_{I(M,P)} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ \rho_{3,1}^{0_M 1_P} \zeta_{1,3} \\ \rho_{3,2}^{1_M 0_P} \zeta_{2,3} \\ \rho_{3,1}^{1_M 1_P} \zeta_{1,3} + \rho_{3,2}^{1_M 1_P} \zeta_{2,3} \end{pmatrix} + \begin{pmatrix} \lambda_{I(M,P)} \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \cdot 2 \\ 10 \cdot 5 \\ 0 \cdot 2 + 0 \cdot 5 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 50 \\ 0 \end{pmatrix}.$$

Next, we present to develop operations to work with the lifted joint in order to then develop a lifted query answering algorithm.

3.4 Operators for the Lifted Joint Distribution

In this section, we develop a framework for working with the lifted joint distribution as a prerequisite for answering queries in the next section. For conditional query answering using Expression (2.11) and Expression (2.12), we need operations for addition and subtraction, for multiplication, and for inversion. We have already used some of these operations in the construction of the covariance matrix, but here we define them more formally. We apply these operations to blocks that follow the same structure as the blocks in the covariance matrix.

3.4.1 Addition and Subtraction

Addition and subtraction of Kronecker component blocks is simple. The precondition is that the Kronecker component blocks share the same Kronecker permutation set. The factors belonging to the same Kronecker sequence in the Kronecker component blocks are added or subtracted to get the resulting Kronecker component block.

Example 3.4.1. Two Kronecker component blocks of the same Kronecker permutation set are added:

$$\begin{pmatrix} 5^{0_L} \\ 3^{1_L} \end{pmatrix} + \begin{pmatrix} 2^{0_L} \\ 2^{1_L} \end{pmatrix} = \begin{pmatrix} 7^{0_L} \\ 5^{1_L} \end{pmatrix}$$

3.4.2 Multiplication

Multiplying two Kronecker component blocks $\mathbf{\Sigma}_{Y_s,Y_k}$ and $\mathbf{\Sigma}_{Y_k,Y_t}$ comes down to a repeated application of Expression (3.49) because every summand of $\mathbf{\Sigma}_{Y_s,Y_k}$ is multiplied with every summand of $\mathbf{\Sigma}_{Y_k,Y_t}$, with each summand following the same structure as the $\boldsymbol{T}$ matrix, meaning we can calculate the values in the resulting Kronecker component block in a lifted way as follows:

$$\rho^{\boldsymbol{q}_{s,t}} = \sum_{\boldsymbol{q}_{s,k}=\mathbf{0}}^{\mathbf{1}} \sum_{\boldsymbol{q}_{k,t}=\mathbf{0}}^{\mathbf{1}} id(\boldsymbol{q}_{s,t}, or(\boldsymbol{q}_{s,k}, \boldsymbol{q}_{k,t}))\rho_{s,k}^{\boldsymbol{q}_{s,k}} \rho_{k,t}^{\boldsymbol{q}_{k,t}} \prod_{L \in \boldsymbol{L}_k} |\boldsymbol{D}(L)|^{lexp(L,\boldsymbol{q}^{(s,k)},\boldsymbol{q}^{(k,t)})}, \tag{3.54}$$

with $or(\boldsymbol{q}_{s,k}, \boldsymbol{q}_{k,t})$ as the bitwise or-operation, a function id that returns 1 if the resulting logvar sequence of the or-operation is identical to the values in $\boldsymbol{q}_{(s,t)}$ at the logvar positions referenced in $\boldsymbol{q}_{(s,t)}$, i.e.,

$$id(\boldsymbol{q}_{s,t}, \boldsymbol{q}) = \begin{cases} 1 & \text{if } \pi_{\boldsymbol{q}_{s,t}}(\boldsymbol{q}) = \boldsymbol{q}_{s,t}, \\ 0 & \text{otherwise,} \end{cases} \tag{3.55}$$

and a function *lexp*

$$lexp(L, \boldsymbol{q}_{s,k}, \boldsymbol{q}_{k,t}) = \begin{cases} 1 & (L \in \boldsymbol{L}_k \setminus \boldsymbol{L}_s \vee \pi_L(\boldsymbol{q}_{s,k}) = \mathbf{1}) \wedge (L \in \boldsymbol{L}_k \setminus \boldsymbol{L}_t \vee \pi_L(\boldsymbol{q}_{k,t}) = \mathbf{1}), \\ 0 & \text{otherwise}, \end{cases} \tag{3.56}$$

which returns 1 if the referenced logvar L fulfils both parts of a conjunction. The first part asks that L occurs only in $\boldsymbol{L}_k$ and not $\boldsymbol{L}_s$ or that the value of L in $\boldsymbol{q}_{s,k}$ is 1, which can only happen if the first disjunct is false. The second part asks the same regarding t instead of s. The returned value as an exponent ensures that the factor of the domain size $|\boldsymbol{D}(L)|$ occurs whenever two all-ones matrices meet.

The main idea of Expression (3.54) is to take into account all possible combinations of $\mathbf{J}$ and $\mathbf{I}$ matrices that occur in the summations of both blocks. Expression (3.54) shows that the Kronecker component block structure is closed under multiplication, which is necessary to combine it with other operations or to multiply several Kronecker component blocks without leaving the structure. Block matrix equation rules allow us to use Expression (3.54) also for multiplying rows or matrices of structured blocks as long as the dimensions match.

Example 3.4.2. We take the resulting $\boldsymbol{\rho}_{2,3}$ and $\boldsymbol{\rho}_{3,3}$ Kronecker component blocks from Example 3.3.4:

$$\boldsymbol{\rho}_{res} = \begin{pmatrix} \boldsymbol{\rho}_{res}^{1_M 0_P} \\ \boldsymbol{\rho}_{res}^{1_M 1_P} \end{pmatrix}$$

We focus on calculating $\boldsymbol{\rho}_{res}^{1_M 0_P}$, the other entry is calculated analogously. In the iteration, we calculate the *id* operation for all combinations of elements of $\mathbf{\Phi}_{2,3}$ and $\mathbf{\Phi}_{3,3}$. For better readability, we calculate them separately:

$$\begin{pmatrix} id(1_M0_P, or(1_M0_P, 0_M0_P)) \\ id(1_M0_P, or(1_M0_P, 0_M1_P)) \\ id(1_M0_P, or(1_M0_P, 1_M0_P)) \\ id(1_M0_P, or(1_M0_P, 1_M1_P)) \\ id(1_M0_P, or(1_M1_P, 0_M0_P)) \\ id(1_M0_P, or(1_M1_P, 0_M1_P)) \\ id(1_M0_P, or(1_M1_P, 1_M0_P)) \\ id(1_M0_P, or(1_M1_P, 1_M1_P)) \end{pmatrix} = \begin{pmatrix} id(1_M0_P, 1_M0_P) \\ id(1_M0_P, 1_M1_P) \\ id(1_M0_P, 1_M0_P) \\ id(1_M0_P, 1_M1_P) \\ id(1_M0_P, 1_M1_P) \\ id(1_M0_P, 1_M1_P) \\ id(1_M0_P, 1_M1_P) \\ id(1_M0_P, 1_M1_P) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

We only write down the elements of the sum that have a result of 1 for *id* function. Then, we have four different calls of *lexp* function, which we also write down separately for better readability:

$$\begin{pmatrix} lexp(M, 1_M0_P, 0_M0_P) \\ lexp(P, 1_M0_P, 0_M0_P) \\ lexp(M, 1_M0_P, 1_M0_P) \\ lexp(P, 1_M0_P, 1_M0_P) \end{pmatrix} = \begin{pmatrix} (True \lor True) \land (False \lor False) \\ (False \lor False) \land (False \lor False) \\ (True \lor True) \land (False \lor True) \\ (False \lor False) \land (False \lor False) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Then, the resulting ρ entry is calculated as follows:

$$\begin{aligned} \boldsymbol{\rho}_{res}^{1_M0_P} = & \sum_{\boldsymbol{q}_{2,3} \in \boldsymbol{\Phi}_{2,3}} \sum_{\boldsymbol{q}_{3,3} \in \boldsymbol{\Phi}_{3,3}} id(1_M0_P, or(\boldsymbol{q}_{2,3}, \boldsymbol{q}_{3,3}))\rho_{2,3}^{\boldsymbol{q}_{2,3}} \rho_{3,3}^{\boldsymbol{q}_{3,3}} \prod_{L \in \{M,P\}} |\boldsymbol{D}(L)|^{lexp(L, \boldsymbol{q}^{(2,3)}, \boldsymbol{q}^{(3,3)})} \\ = & \rho_{2,3}^{1_M0_P} \rho_{3,3}^{0_M0_P} \prod_{L \in \{M,P\}} |\boldsymbol{D}(L)|^{lexp(L, 1_M0_P, 0_M0_P)} \\ & + \rho_{2,3}^{1_M0_P} \rho_{3,3}^{1_M0_P} \prod_{L \in \{M,P\}} |\boldsymbol{D}(L)|^{lexp(L, 1_M0_P, 1_M0_P)} \\ = & \rho_{2,3}^{1_M0_P} \rho_{3,3}^{0_M0_P} \cdot 1 + \rho_{2,3}^{1_M0_P} \rho_{3,3}^{1_M0_P} |\boldsymbol{D}(M)| = 10 \cdot 3 \cdot 1 + 10 \cdot 50 \cdot 3 = 1530. \end{aligned}$$

The other entries of the resulting $\boldsymbol{\rho}$ vector are calculated analogously.

3.4.3 Lifted Matrix Inversion

In this section, we look at how to invert a matrix following the structure from Lemma 3.3.3. We start by understanding how to invert individual blocks of the matrix and then apply the block matrix inversion algorithm.

Inverting a Diagonal Kronecker Component Block

The Kronecker component blocks on the diagonal follow a specific form researched by Searle and Henderson (1979). For this inversion, we use Lemma 3.4.1 by Searle and Henderson (1979) for which they provide a detailed deduction and proof in their work.

Lemma 3.4.1. *Given*

$$\boldsymbol{V}_p = \sum_{\boldsymbol{q}=\boldsymbol{0}}^{\boldsymbol{1}} \theta_{\boldsymbol{q}} (\mathbf{J}_{n_p}^{q_p} \otimes ... \otimes \mathbf{J}_{n_p}^{q_1}), \tag{3.57}$$

of order $N_p = \prod_{i=1}^{p} n_i$ where q_i refers to the entries in $\boldsymbol{q}$ and n_i to each dimension, the inverse is given by

$$\boldsymbol{V}_p^{-1} = \sum_{\boldsymbol{q}=\boldsymbol{0}}^{\boldsymbol{1}} \kappa_{\boldsymbol{q}} (\mathbf{J}_{n_p}^{q_p} \otimes ... \otimes \mathbf{J}_{n_p}^{q_1}), \tag{3.58}$$

where

$$\boldsymbol{R}_p = \begin{bmatrix} 1 & 0 \\ 1 & n_p \end{bmatrix} \otimes ... \otimes \begin{bmatrix} 1 & 0 \\ 1 & n_1 \end{bmatrix} \tag{3.59}$$

and

$$\boldsymbol{\kappa} = \boldsymbol{R}^{-1}\frac{1}{\boldsymbol{R\theta}}. \tag{3.60}$$

There are two main benefits of Lemma 3.4.1. First, it works fully in a lifted way because it does not involve any matrix operations dependent on the cardinality of the all-ones and identity matrices. Second, it stays within the Kronecker component block structure, making it possible to use the result in all upcoming calculations analogously to working with other blocks in the covariance matrix. The individual n_i dimension values correspond to the values in the cardinality vector $\boldsymbol{\tau}$ of the lifted joint distribution.

Definition 3.4.1 (Inv)**.** Let $\boldsymbol{\rho}_{s,s}$ be a vector of the lifted covariance tensor and $\boldsymbol{\tau}$ be the cardinality vector that can be filtered for the cardinalities corresponding to the PRV Y_s. We call the function that is returning the inverse of Kronecker component block structured matrix corresponding PRV Y_s based on Lemma 3.4.1 $INV(\boldsymbol{\rho}_{s,s}, \boldsymbol{\tau})$.

Interestingly, the approach used in the non-overlapping scenario (Hartwig and Möller, 2020b) is a special case of Lemma 3.4.1, where the Kronecker sequences consist of only one term. Lemma 3.4.1 simplifies into Lemma 3.4.2 (Searle and Henderson, 1979).

Lemma 3.4.2. *Let $\boldsymbol{L}$ be an $G \times G$ matrix of the form $\boldsymbol{L} = a\mathbf{J} + b\mathbf{I}$. Then the inverse of L can be calculated analytically by $\boldsymbol{L}^{-1} = x\mathbf{J} + y\mathbf{I}$, where*

$$x = -\frac{a}{b(aG+b)} \text{ and } y = \frac{1}{b}. \tag{3.61}$$

Proof. To prove this lemma, we follow the definition of an inverse $\boldsymbol{L}\boldsymbol{L}^{-1} = \mathbf{I}$. Solving the equation $\boldsymbol{L}\boldsymbol{L}^{-}1 = (a\mathbf{J} + b\mathbf{I})(x\mathbf{J} + y\mathbf{I}) = \mathbf{I}$ is equivalent to solving the linear equation system (LES)

$$(a+b)(x+y) + (G-1)ax = 1, \tag{3.62}$$

$$(a+b)x + a(x+y) + (G-2)ax = 0. \tag{3.63}$$

Solving the LES for x and y results in Equation 3.61. □

Now that we can invert individual blocks on the diagonal and multiply them with other blocks from the covariance matrix without changing the structure, we can apply the block matrix inversion algorithm.

Block Matrix Inversion

To break down the inversion, we use the block matrix inversion formula (Bernstein, 2009):

$$\begin{bmatrix} \tilde{\boldsymbol{A}} & \tilde{\boldsymbol{B}} \\ \tilde{\boldsymbol{C}} & \tilde{\boldsymbol{D}} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{O} & \boldsymbol{P} \\ \boldsymbol{Q} & \boldsymbol{R} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{A}}^{-1} + \tilde{\boldsymbol{A}}^{-1}\tilde{\boldsymbol{B}}\tilde{\boldsymbol{F}}^{-1}\tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^{-1} & -\tilde{\boldsymbol{A}}^{-1}\tilde{\boldsymbol{B}}\tilde{\boldsymbol{F}}^{-1} \\ -\tilde{\boldsymbol{F}}^{-1}\tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^{-1} & \tilde{\boldsymbol{F}}^{-1} \end{bmatrix}, \tag{3.64}$$

where $\tilde{\boldsymbol{F}} = \tilde{\boldsymbol{D}} - \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^{-1}\tilde{\boldsymbol{B}}$.

To invert a full matrix $\boldsymbol{Z}$ that follows the structure of the covariance matrix, we use the lifted inversion of individual blocks together with lifted matrix multiplication recursively. The steps within the recursive approach are defined as follows. Pseudocode can be found in Alg. 2.

Algorithm 2 Lifted recursive block matrix inversion

```
1: function LiftedInversion($\boldsymbol{\rho}, \boldsymbol{\tau}$)          ▷ lifted representation as an input
2:     if $\boldsymbol{\tau}.size = 1$ then
3:         $\rho_{inv} \leftarrow$ Inv($\boldsymbol{\rho}_{1,1}, \boldsymbol{\tau}$)          ▷ based on Definition 3.4.1
4:         return $\rho_{inv}$
5:     $\boldsymbol{\rho}_{\tilde{\boldsymbol{A}}} \leftarrow \boldsymbol{\rho}_{1,1}$
6:     $\boldsymbol{\rho}_{\tilde{\boldsymbol{B}}}, \boldsymbol{\rho}_{\tilde{\boldsymbol{C}}} \leftarrow \boldsymbol{\rho}_{1,2:K}, \boldsymbol{\rho}_{2:K,1}$
7:     $\boldsymbol{\rho}_{\tilde{\boldsymbol{D}}} \leftarrow \boldsymbol{\rho}_{2:K,2:K}$
8:     $\boldsymbol{\rho}_{\tilde{\boldsymbol{A}}^{-1}} \leftarrow$ LiftedInversion($\boldsymbol{\rho}_{\tilde{\boldsymbol{A}}}, \boldsymbol{\tau}$)          ▷ recursive call
9:     $\boldsymbol{\rho}_{\tilde{\boldsymbol{F}}} \leftarrow \boldsymbol{\rho}_{\tilde{\boldsymbol{D}}} -$ multi($\boldsymbol{\rho}_{\tilde{\boldsymbol{C}}}, \boldsymbol{\rho}_{\tilde{\boldsymbol{A}}^{-1}}, \boldsymbol{\rho}_{\tilde{\boldsymbol{B}}}, \boldsymbol{\tau}$)          ▷ based on Expression (3.54)
10:    $\boldsymbol{\rho}_{\tilde{\boldsymbol{F}}^{-1}} \leftarrow$ LiftedInversion($\boldsymbol{\rho}_{\tilde{\boldsymbol{F}}}, \boldsymbol{\tau}$)          ▷ recursive call
11:    $\boldsymbol{\rho}_{\boldsymbol{O}}, \boldsymbol{\rho}_{\boldsymbol{P}}, \boldsymbol{\rho}_{\boldsymbol{Q}}, \boldsymbol{\rho}_{\boldsymbol{R}} \leftarrow$ multi($\boldsymbol{\rho}_{\tilde{\boldsymbol{A}}}, \boldsymbol{\rho}_{\tilde{\boldsymbol{B}}}\boldsymbol{\rho}_{\tilde{\boldsymbol{C}}}, \boldsymbol{\rho}_{\tilde{\boldsymbol{F}}}$)          ▷ Expressions (3.54) and (3.64)
12:    $\boldsymbol{\rho}_{inv} \leftarrow$ Stack($\boldsymbol{\rho}_{\boldsymbol{O}}, \boldsymbol{\rho}_{\boldsymbol{P}}, \boldsymbol{\rho}_{\boldsymbol{Q}}, \boldsymbol{\rho}_{\boldsymbol{R}}$)          ▷ based on Expression (3.64)
13:    return $\boldsymbol{\rho}_{inv}$          ▷ lifted representation of the inverse as an output
```

Step 1: The input for the function is the lifted representation $\boldsymbol{\rho}$ and $\boldsymbol{\tau}$ of the matrix $\boldsymbol{Z}$. If the matrix $\boldsymbol{\rho}$ has only one element, $\boldsymbol{Z}$ would consist of only one block and we can use Lemma 3.4.1 to calculate the lifted representation of $\boldsymbol{Z}^{-1}$. The resulting $\boldsymbol{\rho}_{res}$ is returned and the function call terminates. If $\boldsymbol{\rho}$ has more than one element the algorithm continues.

Step 2: The grounded matrix $\boldsymbol{Z}$ would be split into four blocks based on Equation 3.64. These four blocks are constructed of the $K \times K$ blocks forming the matrix $\boldsymbol{Z}$ (where K is decreased by one in each recursion step). In our fully lifted algorithm, we analogously split the tensor $\boldsymbol{\rho}$:

- $\tilde{\boldsymbol{A}} = \boldsymbol{B}_{1,1}$, $\rho_{\tilde{\boldsymbol{A}}} = \rho_{1,1}$,
- $\tilde{\boldsymbol{B}} = \boldsymbol{B}_{1,2:K}$, $\boldsymbol{\rho}_{\tilde{\boldsymbol{B}}} = \boldsymbol{\rho}_{1,2:K}$
- $\tilde{\boldsymbol{C}} = \boldsymbol{B}_{1,1}$, $\boldsymbol{\rho}_{\tilde{\boldsymbol{C}}} = \boldsymbol{\rho}_{2:K,1}$
- $\tilde{\boldsymbol{D}} = \boldsymbol{B}_{2:K,2:K}$, $\boldsymbol{\rho}_{\tilde{\boldsymbol{D}}} = \boldsymbol{\rho}_{2:K,2:K}$

Step 3: The inversion function is called for the lifted representation of $\tilde{\boldsymbol{A}}$ and Step 1 will directly return the lifted inverse. The lifted version $\boldsymbol{\rho}_{\tilde{\boldsymbol{F}}}$ of matrix $\tilde{\boldsymbol{F}}$ can be calculated using the lifted multiplication rules from Expression (3.54). The inversion algorithm is recursively called for $\boldsymbol{\rho}_{\tilde{\boldsymbol{F}}}$.

Step 4: Once the recursive call returns the lifted inverse for $\boldsymbol{\rho}_{\tilde{\boldsymbol{F}}}$, $\boldsymbol{\lambda}_{\tilde{\boldsymbol{F}}}$ and $\boldsymbol{\tau}_{\tilde{\boldsymbol{F}}}$, the blocks the lifted representations for $\boldsymbol{O}$, $\boldsymbol{P}$, $\boldsymbol{Q}$ and $\boldsymbol{R}$ can be calculated using Expression (3.54). Based on Expression (3.64), the four lifted representations of $\boldsymbol{O}$, $\boldsymbol{P}$, $\boldsymbol{Q}$ and $\boldsymbol{R}$ are combined into $\boldsymbol{\rho}_{\boldsymbol{Z}^{-1}}$.

Having now the ability to work with the lifted version of the covariance matrix, we look into using this lifted joint representation and the lifted operations to answer queries.

3.5 Lifted Query Answering

This section covers query answering using the lifted joint distribution. As described in the preliminaries, a query $P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e})$ with respect to the ground model is defined by its query variables $\boldsymbol{Q} \subset \mathcal{X}$ and the evidence $\boldsymbol{E}=\boldsymbol{e}$ with $\boldsymbol{E} \subset \mathcal{X}$. The query answer is a conditional probability distribution over the variables $\boldsymbol{Q}$ given the evidence. Of course we could also query individual events $\boldsymbol{Q}=\boldsymbol{q}$ but because we can use the query answer for the full distribution of $\boldsymbol{Q}$ to single out events in a second step we focus on querying the distribution over $\boldsymbol{Q}$.

To calculate the conditional distribution we use the special algorithm for Gaussian distributions introduced in Expression (2.11) and Expression (2.12), which we repeat here for better readability:

$$\boldsymbol{\mu}^* = \boldsymbol{\mu}_{\boldsymbol{Q}} + \boldsymbol{\Sigma}_{\boldsymbol{QE}}\boldsymbol{\Sigma}_{\boldsymbol{EE}}^{-1}(\boldsymbol{e}-\boldsymbol{\mu}_{\boldsymbol{E}}), \tag{3.65}$$

$$\boldsymbol{\Sigma}^* = \boldsymbol{\Sigma}_{\boldsymbol{QQ}} - \boldsymbol{\Sigma}_{\boldsymbol{QE}}\boldsymbol{\Sigma}_{\boldsymbol{EE}}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{EQ}}. \tag{3.66}$$

The special properties of the Gaussian distribution allow us to avoid solving integrals in the process of marginalization. In the upcoming subsections, we develop algorithms based Expression (3.65) and Expression (3.66) that work with the lifted joint distribution instead of the ground distribution.

3.5.1 Lifted Answering of a Marginal Query

The algorithm for obtaining a marginal distribution of a multivariate normal distribution is trivial, because all matrix calculations in Expression (3.65) and Expression (3.66) can be omitted when having an empty matrix $\boldsymbol{\Sigma}_{\boldsymbol{EE}}$. One can simply select the means $\boldsymbol{\mu}_{\boldsymbol{Q}}$ and covariance sub-matrix $\boldsymbol{\Sigma}_{\boldsymbol{QQ}}$ corresponding to the queried randvars and insert them into the probability distribution

$$P(\boldsymbol{Q}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{Q}}; \boldsymbol{\Sigma}_{\boldsymbol{QQ}}). \tag{3.67}$$

For the lifted case, we select the means by $\boldsymbol{\mu}_{\boldsymbol{Q}} = \eta_{\boldsymbol{lif}(\boldsymbol{Q})}$, ground the covariance matrix $\boldsymbol{\Sigma}$ with Expression (3.47), and then select the elements with $\boldsymbol{\Sigma}_{\boldsymbol{QQ}}$.

3.5.2 Lifted Answering of a Conditional Query

We define a few helper functions:

Definition 3.5.1 (const, perm, gr)**.** Given a set of grounded randvars $\boldsymbol{X}$, a PRV Y with a randvar name ξ, a sequence of constants $\boldsymbol{l}$ and a set of sequences of constants $\boldsymbol{\iota}$, we define the following three helper functions:

$$const_{\boldsymbol{L}}(\boldsymbol{X}) = \{\sigma_{\boldsymbol{L}}(\boldsymbol{l}) | \xi(\boldsymbol{l}) \in \boldsymbol{X}\}, \tag{3.68}$$

$$perm(\boldsymbol{X}) = \{\boldsymbol{l} | \boldsymbol{l} \in \times_{L \in lv(lif(\boldsymbol{X}))} const_L(\boldsymbol{X})\}, \tag{3.69}$$

$$gr(Y, \boldsymbol{\iota}) = \{\xi(\boldsymbol{l}') | \xi(\boldsymbol{l}') \in \boldsymbol{gr}(Y) \wedge \pi_{\boldsymbol{L}_Y}(\boldsymbol{l}) = \boldsymbol{l}' \wedge \boldsymbol{l} \in \boldsymbol{\iota}\}. \tag{3.70}$$

When answering a conditional query based on Expression (3.65) and Expression (3.66), we need to apply the following operations:

1. lifted addition when adding the lifted version of $\boldsymbol{\Sigma_{QQ}}$ to $\boldsymbol{\Sigma_{QE}}\boldsymbol{\Sigma_{EE}^{-1}}\boldsymbol{\Sigma_{EQ}}$,
2. lifted multiplication in $\boldsymbol{\Sigma_{QE}}\boldsymbol{\Sigma_{EE}^{-1}}\boldsymbol{\Sigma_{EQ}}$ and in $\boldsymbol{\Sigma_{QE}}\boldsymbol{\Sigma_{EE}^{-1}}$,
3. lifted inversion in $\boldsymbol{\Sigma_{EE}^{-1}}$, and
4. lifted handling of the ground evidence values $\boldsymbol{e} - \boldsymbol{\mu_E}$ and the combination with the lifted result of $\boldsymbol{\Sigma_{QE}}\boldsymbol{\Sigma_{EE}^{-1}}$.

The covariance matrix $\boldsymbol{\Sigma}$ follows a structure that allows addition, multiplication, and inversion. However, this structure is not necessarily given for $\boldsymbol{\Sigma_{QQ}}, \boldsymbol{\Sigma_{QE}}, \boldsymbol{\Sigma_{EQ}}$, and $\boldsymbol{\Sigma_{EE}}$. For inversion and matrix multiplications, all blocks in these matrices need to follow the Kronecker block structure with matching dimensions. The Kronecker block structure is given when all ground level observation randvars $\boldsymbol{E}$ and query randvars $\boldsymbol{Q}$ have the same set of instantiated logvars. We call the query to be liftable if $\boldsymbol{E}$ and $\boldsymbol{Q}$ fulfil this condition with respect to the lifted joint distribution $\boldsymbol{\rho}$ and $\boldsymbol{\tau}$. More formally liftability is defined as:

Definition 3.5.2 (Liftable query)**.** Given a set of PRVs $\mathcal{Y}$ and a corresponding set of logvars $\mathcal{L}$, a *query* $P(\boldsymbol{Q}|\boldsymbol{E} = \boldsymbol{e})$, with $\boldsymbol{E} \subset \boldsymbol{gr}(\mathcal{Y})$ and $\boldsymbol{Q} \subset \boldsymbol{gr}(\mathcal{Y})$, is *liftable* if it fulfils

$$(\forall Y \in \boldsymbol{lif}(\boldsymbol{E}) : \boldsymbol{gr}(Y, perm(\boldsymbol{E} \cup \boldsymbol{Q})) \in \boldsymbol{E}) \wedge (\forall Y \in \boldsymbol{lif}(\boldsymbol{Q}) : \boldsymbol{gr}(Y, perm(\boldsymbol{E} \cup \boldsymbol{Q})) \in \boldsymbol{E}). \tag{3.71}$$

Definition 3.5.2 ensures that matrices $\boldsymbol{\Sigma_{QQ}}, \boldsymbol{\Sigma_{QE}}, \boldsymbol{\Sigma_{EQ}}$, and $\boldsymbol{\Sigma_{EE}}$ of a liftable query follow the Kronecker block structure needed to perform matrix multiplications and inversions in a lifted way. The restriction excludes all queries that do not follow the Kronecker block structure from being calculated in a lifted way. One example for a non-liftable query would be a query having ground randvars of the same PRV in both evidence set and query set. In the non-overlapping logvar case, Expression (3.71) is always true as long as randvars belonging to the same PRV are not present in both query and evidence set. The matrix operations enable lifted calculations of Operations 1-3 in above's list, resulting in the conditional covariance $\mathbf{\Sigma}^*$. The last operation (4 in above's list) means calculating the conditional mean $\boldsymbol{\mu}^*$, which involves handling evidence. The easiest way to handle evidence is to do the calculations on ground level which means using the ground level $\boldsymbol{\mu}_{\boldsymbol{Q}}$ and ground result of $\mathbf{\Sigma}_{\boldsymbol{QE}}\mathbf{\Sigma}_{\boldsymbol{EE}}^{-1}$ to do a grounded matrix multiplication with the ground level vector $\boldsymbol{e} - \boldsymbol{\mu}_{\boldsymbol{E}}$ (corresponding to operation 4 in the list above). Even with this grounding, performing the other steps in a lifted way already brings major time savings because more complex matrix operations are still avoided, but we are not satisfied yet. Therefore, we look into cases next where we can group the ground level evidence to further reduce the time complexity. Section 3.6 contains the detailed complexity discussion for all steps and approaches discussed here.

3.5.3 Grouping the Evidence

In general, the difficulty of handling evidence in a lifted way is that evidence itself occurs on a ground level, i.e., evidence contains individual randvars including their values. In the continuous setting, it is very unlikely that the same evidence value will occurs for two different randvars. If we assume perfect measure accuracy and a fully continuous setting, the chance for measuring the same value twice is zero (or infinitely close to zero). In the non-overlapping case, different evidence values are not a big deal because every observed randvar of a PRV has the same effect on all randvars belonging to another PRV. Thus, we can group the evidence within each observed PRV by summing up all ground values for the PRV in $\boldsymbol{e} - \boldsymbol{\mu}_{\boldsymbol{E}}$ and multiplying it with the corresponding term from the lifted result of $\mathbf{\Sigma}_{\boldsymbol{QE}}\mathbf{\Sigma}_{\boldsymbol{EE}}^{-1}$. The resulting vector along the K PRVs for which we have evidence is given by

$$\begin{bmatrix} \sum_{E_h \in \boldsymbol{gr}(Y_1^{\boldsymbol{E}})} (e_h - \mu_h) \\ \vdots \\ \sum_{E_h \in \boldsymbol{gr}(Y_K^{\boldsymbol{E}})} (e_h - \mu_h) \end{bmatrix}. \tag{3.72}$$

In the case of overlapping logvar sequences, summing up the evidence for the whole PRV does not work because every overlap introduces independencies, i.e., not all randvars of one PRV have the same effect on all randvars of another PRV. Next, we look at how we can still lift part of the calculations involved in the overlapping case.

Two-PRV Case

We begin with a scenario of only two PRVs before generalizing it to multiple PRVs. We have one PRV $Y_Q = lif(\boldsymbol{Q})$ whose instances are (partially) queried and one PRV $Y_E = lif(\boldsymbol{E})$ whose instances are (partially) observed in the query $P(\boldsymbol{Q}|\boldsymbol{E} = \boldsymbol{e})$. Evidence randvars only influence query randvars that share the same constants. In a liftable query based on Definition 3.5.2, the number of evidence randvars is equal for each queried variable but the actual observed evidence values can be different. In the ground case, the operation is a multiplication of a Kronecker component block $\boldsymbol{\Sigma_{QE}\Sigma_{EE}^{-1}}$ and the evidence vector $(\boldsymbol{e} - \boldsymbol{\mu_E})$. Every row of the Kronecker component block is multiplied with the same vector $(\boldsymbol{e} - \boldsymbol{\mu_E})$. Non-overlapping logvar sequences between the observed Y_E and the queried PRV Y_Q result in duplicates in the Kronecker component block, in a way that logvars only present in Y_E lead to duplicate columns and logvars only present in Y_Q lead to duplicate rows. Duplicates in the rows will lead to the same conditional mean value for the corresponding ground query randvars and duplicates in the columns will lead to the same influence of evidence randvars onto ground randvars. To combine the randvars with the same influence and same result, we group along the overlapping logvars. We denote the overlaps in the logvar sequences as $\boldsymbol{L}_O = \boldsymbol{L}_{lif(\boldsymbol{Q})} \cap \boldsymbol{L}_{lif(\boldsymbol{E})}$ and group the evidence along the constants $\boldsymbol{l}_O$ of the overlapping logvars, with $\boldsymbol{l}_O \in \boldsymbol{D}(\boldsymbol{L}_O)$, and add up the evidence values:

$$sumev_{\boldsymbol{E}}(Y_E, \boldsymbol{l}_O) = \sum e - \mu_E \mid E = e \wedge E \in \boldsymbol{gr}(Y_E, \{\boldsymbol{l}_O\}) \tag{3.73}$$

The conditional mean vector is grouped along the same set of instance sequences

$$\mu_{Y_Q} = \begin{pmatrix} \mu_{Y_Q, \boldsymbol{l}_O^1} \\ \vdots \\ \mu_{Y_Q, \boldsymbol{l}_O^G} \end{pmatrix} = \begin{pmatrix} \eta_{Y_Q} \\ \vdots \\ \eta_{Y_Q} \end{pmatrix} + \begin{pmatrix} \nu_{Y_Q, \boldsymbol{l}_O^1} \\ \vdots \\ \nu_{Y_Q, \boldsymbol{l}_O^G} \end{pmatrix}, \tag{3.74}$$

where G is the number of groups and given by $G = |const_{\boldsymbol{L}_O}(\boldsymbol{Q})|$. The entries $\nu_{Y_Q, \boldsymbol{l}_O^g}$, with $g = 1, ..., G$ are given by

$$\nu_{Y_Q, \boldsymbol{l}_O^g} = \nu_{Y_Q, \boldsymbol{l}_O^g, Y_E}, \tag{3.75}$$

with

$$\nu_{Y_Q, \boldsymbol{l}_O^g, Y_E} = \sum_{\boldsymbol{l}_O \in inst_{\boldsymbol{L}_O}(\boldsymbol{E})} \sum_{\boldsymbol{q} \in \boldsymbol{Q}_{\boldsymbol{L}_Q, \boldsymbol{L}_E}} sumev_{\boldsymbol{E}}(Y_E, \boldsymbol{l}_O^g) \rho^{\boldsymbol{q}}_{Y_Q, Y_E} filter(\boldsymbol{l}_O^g, \boldsymbol{l}_O, \boldsymbol{q}), \tag{3.76}$$

and

$$filter(\boldsymbol{l}_O^g, \boldsymbol{l}_O, \boldsymbol{q}) = \begin{cases} 1 & \pi_{\sigma_{q=0}(\boldsymbol{q})}(\boldsymbol{l}_O^g) = \pi_{\sigma_{q=0}(\boldsymbol{q})}(\boldsymbol{l}_O), \\ 0 & \text{otherwise}, \end{cases} \tag{3.77}$$

where q in the selection is a placeholder for all positions in $\boldsymbol{q}$. The filter function ensures that independencies between randvars in $\boldsymbol{Q}$ and $\boldsymbol{E}$ are taken into account. In a full overlap, the sum simplifies to Expression (3.72).

Multi-PRV Case

Having solved the case for one evidence and one query PRV, we now relax the restriction on the number of PRVs and allow for multiple PRVs. The liftable query rule from Definition 3.5.2 still applies, but even with this rule there might be different overlapping sets for each evidence PRV to the queried PRVs. The sequence of all overlapping logvars between any PRV in $\boldsymbol{Q}$ and any PRV in $\boldsymbol{E}$ is still given by $\boldsymbol{L}_O = \boldsymbol{L}_{lif(\boldsymbol{Q})} \cap \boldsymbol{L}_{lif(\boldsymbol{E})}$. For each query PRV Y_Q, the groups for the final mean $\mu_{Y_Q,\boldsymbol{l}_O^g}$ and for the result of the Kronecker component block multiplication $\nu_{Y_Q,\boldsymbol{l}_O^g}$ are given by $inst_{\boldsymbol{L}_O}(\boldsymbol{Q})$, such that $\boldsymbol{l}_O^g \in inst_{\boldsymbol{L}_O}(\boldsymbol{Q})$. If we have more than one evidence PRV, the equality of $\nu_{Y_Q,\boldsymbol{l}_O^g} = \nu_{Y_Q,\boldsymbol{l}_O^g,Y_E}$ in Expression (3.75) does not hold anymore, because all evidence randvars might influence the mean values. Expression (3.76) still works for all individual evidence PRVs and query PRVs but can contain different groupings that need to be combined in a way that the values in $\nu_{Y_Q,\boldsymbol{l}_O^g,Y_E}$ are sorted into the correct group of $\nu_{Y_Q,\boldsymbol{l}_O^g}$. The grouping is done by

$$\nu_{Y_Q,\boldsymbol{l}_O} = \sum_{Y_E \in \boldsymbol{lif}(\boldsymbol{E})} \nu_{Y_Q,\boldsymbol{l}'_O,Y_E} match(\boldsymbol{l}'_O, \boldsymbol{l}_O), \tag{3.78}$$

with

$$match(\boldsymbol{l}'_O, \boldsymbol{l}_O) = \begin{cases} 1 & \pi_{\boldsymbol{L}'_O}(\boldsymbol{l}_O) = \boldsymbol{l}'_O, \\ 0 & \text{otherwise}, \end{cases} \tag{3.79}$$

where $\boldsymbol{L}'_O$ is referring to the logvar sequence corresponding to the constants in $\boldsymbol{l}'_O$. Values calculated in Expression (3.78) can then be put into Expression (3.74) for the final query answer.

3.5.4 Example Queries and Calculations

In this section, we cover a few example queries to illustrate the liftability characteristic and the calculations performed in query answering. All queries use our running example from Fig. 3.1.

Example 3.5.1. Given the lifted joint distribution in Appendix A.3, we define the following queries, where we omit set parentheses and value assignments for better readability.

1. $P(H(P1)|E(M1), E(M2), S(P1))$
2. $P(H(P1), H(P2)|S(P1), S(P2), I(M1, P1), I(M1, P2), I(M2, P1), I(M2, P2))$
3. $P(U(C1), U(C2)|H(P1), H(P2))$

4. $P(H(P1)|I(M1,P1),S(P2))$
5. $P(H(P1)|H(P2))$

The entries of the evidence vector $\boldsymbol{e}$ for every query are the natural numbers beginning with one.

Query 1: Query 1 is a liftable query and we have only one constant defining the groups for the conditional mean. The query answer is given by:

$$\boldsymbol{\rho}^*_{H(P)} = \begin{pmatrix} \rho^{0_P}_{H(P)} \\ \rho^{1_P}_{H(P)} \end{pmatrix} = \begin{pmatrix} 1 \\ 6642 \end{pmatrix}; \eta^*_{H(P)} = 48; \Sigma^*_{H(P1)} = 6643; \mu^*_{H(P1)} = 48$$

Query 2: Query 2 is a liftable query and we have two constants defining the groups for the conditional mean. The query answer is given by:

$$\boldsymbol{\rho}^*_{H(P)} = \begin{pmatrix} \rho^{0_P}_{H(P)} \\ \rho^{1_P}_{H(P)} \end{pmatrix} = \begin{pmatrix} 49 \\ 64 \end{pmatrix}; \begin{pmatrix} \eta^*_{H(P),P1} \\ \eta^*_{H(P),P2} \end{pmatrix} = \begin{pmatrix} 50 \\ 70 \end{pmatrix};$$

$$\boldsymbol{\Sigma}^* = \begin{bmatrix} 113 & 64 \\ 64 & 113 \end{bmatrix}; \begin{pmatrix} \mu^*_{H(P1)} \\ \mu^*_{H(P2)} \end{pmatrix} = \begin{pmatrix} 50 \\ 70 \end{pmatrix}$$

Query 3: Query 3 is a liftable query. We have only one single group because there is no overlap between query and evidence. The query answer is given by:

$$\boldsymbol{\rho}^*_{U(C)} = \begin{pmatrix} \rho^{0_P}_{U(C)} \\ \rho^{1_P}_{U(C)} \end{pmatrix} = \begin{pmatrix} 3 \\ 152430 \end{pmatrix}; \eta^*_{U(C)} = -2048;$$

$$\boldsymbol{\Sigma}^* = \begin{bmatrix} 152434 & 152430 \\ 152430 & 152434 \end{bmatrix}; \begin{pmatrix} \mu^*_{U(C1)} \\ \mu^*_{U(C2)} \end{pmatrix} = \begin{pmatrix} -2048 \\ -2048 \end{pmatrix}$$

Query 4: Query 4 is a non-liftable query because P1 and P2 occur in the query but we only have a measurement for I(M1,P1). The mismatch between the constants in the query set and the evidence set for the for *Patient* logvar (P1 and P2 versus only P1) contradicts Definition 3.5.2.

Query 5: Query 5 is a non-liftable query because randvars of the same PRV occur in the evidence and the query set. Whenever the evidence set and the query set PRVs are not disjoint the symmetry condition in Definition 3.5.2 is contradicted.

The example shows, that there are queries that we currently cannot calculate in a lifted way. Query answering is still possible using ground level operations. Of course it would be beneficial to allow for partial lifted algorithms which we will discuss later in the outlook.

We have lifted approaches for constructing the joint distribution and lifted query answering algorithms for working with the lifted joint. We have also shown in a running example how all operations can be used and how queries can be answered in a lifted way. Next, we evaluate the complexity gains by a theoretical complexity analysis.

3.6 Complexity Analysis

This section covers the runtime and space complexity of creating the joint distribution and the runtime complexity of answering conditional probability queries. Let us specify the parameters used for this complexity analysis. We have N randvars in the GBN combined into M PRVs. The terms N_E and N_Q denote the number of randvars in the evidence and the query set respectively, whereas the terms M_E and M_Q denote the number of PRVs referenced in the evidence and query set respectively. The term Λ denotes the number of logvars and S the longest logvar sequence. In general, we focus on the upper bound of the runtime complexity. There are matrix inversion and multiplication algorithms that have a runtime complexity of less than $O(N^3)$, but for simplicity and without changing the overall argumentation, we take $O(N^3)$ as an upper bound (Davie and Stothers, 2013; Le Gall, 2014).

3.6.1 Complexity for Constructing the Joint Distribution

The only time-consuming part when creating the lifted joint is to generate the joint covariance matrix, since the mean values of the nodes just need to be stored into a vector (for the lifted and grounded approach).

Ground Complexity

Constructing the ground version of the joint probability distribution calculates the $N \times N$-dimensional covariance matrix by iterating in two loops over the randvars. In each step, the covariance between two randvars X_i and X_j, with $j > i$, is calculated by taking into account all randvars V_k that occur earlier than X_j, i.e., $k < j$, in the topological ordering, resulting in an upper bound of $O(N^3)$. The space complexity of storing the covariance matrix $\mathbf{\Sigma}$ and the mean vector $\boldsymbol{\mu}$ is $O(N^2 + N) = O(N^2)$.

Lifted Complexity

The lifted algorithm to construct the lifted joint distribution as presented in Section 3.3 is fully independent on the number of randvars N, no matter if the network contains overlaps or not. To construct the $M \cdot M$ $\boldsymbol{\rho}$ vectors, two iterations over all M PRVs are necessary. To calculate one individual new $\boldsymbol{\rho}_{s,t}$ vector, we need to iterate over the PRVs Y_u prior in the topological ordering to Y_t, i.e., $u < t$, and for each of these PRVs Y_u, we iterate at most over 2^S possible Kronecker sequences. Combining these iterations results in an overall runtime complexity of $O(M^3 2^S)$. In a parameterized GBN, it is typically $M \ll N$ and $2^S \ll N$, which results in significant speed up. The space complexity to store all values in $\boldsymbol{\rho}$ is $O(M^2 2^S)$. In the non-overlapping scenario, 2^S reduces to a constant because even if a PRV has a logvar sequence with more than one logvar, the logvars can be combined into a single logvar without influencing the model. When combining logvars, the domain size increases for the combined logvar but this has no effect on the construction runtime as visible in the complexity classes.

3.6.2 Complexity for Conditional Query Answering

For query answering, we need to look at the four operations listed in Section 3.5.2.

Ground Complexity

Matrix addition has a complexity of $O(N_Q^2)$ because every element in the matrix needs to be visited at least once. The calculation of the evidence vector $\boldsymbol{e} - \boldsymbol{\mu_E}$ has a complexity of $O(N_E)$. The inversion of evidence covariance matrix has a complexity of $O(N_E^3)$. The involved matrix multiplications have either a complexity of $O(N_Q^2 N_E)$ and $O(N_Q N_E^2)$. Combining all operations results in a complexity for query answering of $O(N_E^3 + N_Q N_E^2 + N_Q^2 N_E)$. Depending on the partitioning of evidence and query set, either the term N_E^3 or the term $N_Q^2 N_E$ dominates the complexity. Matrix addition and calculation of the evidence vector are out-weighted by the other terms.

Lifted Complexity

Multiplying two Kronecker component blocks has a complexity of $O(2^{2S})$ because in the worst case we need to iterate fully through all permutations of possible Kronecker sequences. Thus, the matrix multiplications (Items 2 and 3 in the list in Section 3.5.2) have complexity of $O(M_E^2 M_Q 2^{2S})$ and $O(M_E M_Q^2 2^{2S})$.

The inversion of one individual Kronecker component block from the diagonal of the covariance matrix involves the multiplication of two at most $2^S \times 2^S$ matrices resulting in a complexity of $O(2^{3S})$. In the block matrix inversion we have at most M_E recursive function calls and Kronecker component block multiplications of at most M_E blocks, resulting in a combined inversion complexity of $O(M_E 2^{3S} + M_E^2 2^S)$.

To calculate the mean vector we still have ground operations. Grounding the result of $\boldsymbol{\Sigma_{QE}} \boldsymbol{\Sigma_{EE}^{-1}}$ takes $O(N_Q N_E)$ time and multiplying it with the ground level evidence vector $\boldsymbol{e} - \boldsymbol{\mu_E}$ is in $O(N_Q N_E)$ as well. This is still a linear relationship to both N_Q and N_E. Overall, this results in a complexity of $O(M_E^2 M_Q 2^{2S} + M_E M_Q^2 2^{2S} + M_E 2^{3S} + N_Q N_E)$.

With the grouping of the evidence along the overlapping logvar instances, we reduce N_Q to G_Q where $G_Q = |\boldsymbol{D}(\boldsymbol{L}_Q \cap \boldsymbol{L}_E)|$ refers to the number of instances that occur both in $\boldsymbol{Q}$ and $\boldsymbol{E}$. The biggest time-savings occur if there is a relatively even split between queried variables and evidence variables, when they have no or little overlap. When there is a full overlap between evidence PRVs and query PRVs, then $G_Q = N_Q$ and the grouping will not lead to efficiency gains. In the non-overlapping case, there is only one group, i.e., $G_Q = 1$, which is consistent with the linear dependency on N_E reported in Hartwig and Möller (2020b).

Table 3.2: Experiment setups

Experiment	PRVs	Domain sizes	Query
Overlap	$E(M), S(P),$ $I(M,P), H(P)$	$\tau_M, \tau_P = 2^1, ..., 2^K$	$\boldsymbol{Q} = \boldsymbol{gr}(H(P)),$ $\boldsymbol{E} = \boldsymbol{gr}(S(P))$
Mixed	$E(M), S(P), I(M,P),$ $H(P), W(D)$	$\tau_M, \tau_D = 2^1, ..., 2^K,$ $\tau_P = 2$	$\boldsymbol{Q} = \boldsymbol{gr}(W(D)),$ $\boldsymbol{E} = \boldsymbol{gr}(E(M))$
No Overlap	$H(P), T(N),$ $W(D), U(C)$	$\tau_P, \tau_T, \tau_N = 2^1, ..., 2^K,$ $\tau_C = 2$	$\boldsymbol{Q} = \boldsymbol{gr}(U(C)),$ $\boldsymbol{E} = \boldsymbol{gr}(H(P), T(N))$

Theorem 3.6.1. *The complexity of query answering as described in Section 3.5 including evidence grouping is given by*

$$O(M_E^2 M_Q 2^{2S} + M_E M_Q^2 2^{2S} + M_E 2^{3S} + G_Q N_E). \tag{3.80}$$

3.7 Empirical results

We show in this empirical study that the complexity gains translate into practice in an implementation in Python code. We verify the theoretical complexity analysis in the experiments. First, we evaluate the lifted construction of the joint distribution described in Section 3.3. Second, we evaluate the conditional query answering. We use three different sub-graphs of our running example along with three different queries focusing on different preconditions for the query answering algorithm. The first experiment (overlap) uses the sub-graph in the orange box, focusing on the full overlap between logvar sequences in query and evidence set. The second experiment (mixed) uses the orange box with the added *workload of doctors* PRV $W(D)$, focusing on a query where grouping plays an important role because of balanced query and evidence set. The third example (no overlap) uses the sub-graph in the blue-box, focusing on a case where there is no overlap between any evidence and query randvars (also not along the path) but where the query-set is hold constant while the evidence set is increased. In all three experiments, we increase certain domain sizes to see the influence on the runtime in both the ground and the lifted approach. Table 3.2 contains the PRVs, domain sizes, and queries used in the experiments. Figure 3.3 shows the results, which we discuss next.

3.7.1 Experimental Constructions

We run the lifted and grounded construction algorithms for all three experiments in Table 3.2. As described in the theoretical analysis, the runtime should be independent of the logvar domain size (cardinality), which is validated in all three experiments as can be seen in Fig. 3.3 (a-c). Overall, the runtime in the construction of the overlap experiment is lowest, because the number of PRVs and the number of logvars involved in the PRVs is lowest. The slight up and downs in the lifted scenarios are due to the very low runtime (milliseconds) and comparably high effect of slight shifts in background processes.

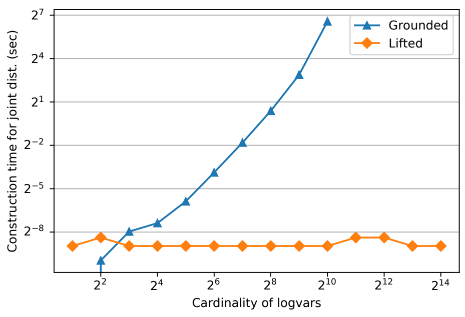
3.7.2 Experimental Query Answering

For query answering, we also use the three experiments from Table 3.2. In all cases, we observe both lifted variants to be outperforming the grounded algorithm from a certain logvar domain size onwards (see Fig. 3.3). In the non-overlapping case (f), where we have a constant number of queried randvars, both lifted algorithms outperform the ground algorithm significantly and show only a linear with respect to the number of evidence randvars increase as expected. When comparing the orange and green lines in Fig. 3.3 (f), we see no speed-ups between the lifted query answering using evidence grouping and the lifted query answering without evidence grouping because the linear factor of N_Q is constant and small. In the overlap experiment in Fig. 3.3 (d), where we also have an overlap between the logvars in $\boldsymbol{Q}$ and $\boldsymbol{E}$, we see that the lifting algorithm with grouping is worse than the algorithm without grouping. Because of the full overlap between $\boldsymbol{Q}$ and $\boldsymbol{E}$ all groups will have exactly one evidence entry resulting in no efficiency gains. With higher domain sizes, the difference between grouping and non-grouping gets smaller because the constant overhead of handling the groups has relatively less effect. In the mixed scenario in Fig. 3.3 (e), we have high speed-ups for the lifted algorithm with grouping compared to the lifted algorithm without grouping. This speed-up occurs, because we have no overlaps between $\boldsymbol{Q}$ and $\boldsymbol{E}$, resulting in one big group for which the evidence can be summed up, resulting in the elimination of the large N_Q factor.

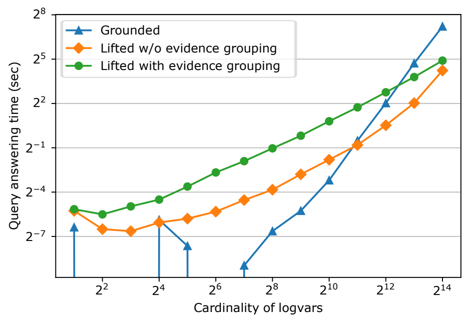
3.8 Intermediate Discussion

We have presented lifting for GBNs. We have developed algorithms for constructing a lifted joint distribution for the non-overlapping case and generalized the representation and construction algorithm to allow for overlaps between the logvars sequences (Contribution **1**). We have developed operations to work with the lifted joint distribution in a fully lifted way including addition, multiplication, and inversion (Contribution **2**). The operations provide the basis for the lifted query answering algorithms developed (Contribution **3**). We have shown with a theoretical complexity analysis and an experimental evaluation that we can achieve significant performance improvements compared to the ground level implementations of the algorithms. A running example has been used to illustrate a possible application and all theoretical operations and algorithms for better comprehensibility.

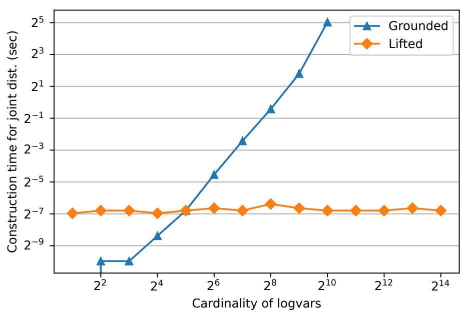
For future research on lifting GBNs, there are mainly three open issues that could trigger further research. First, we have have made some restrictions for liftable queries.



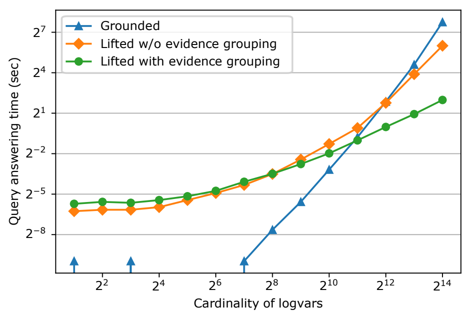
(a) Overlap construction



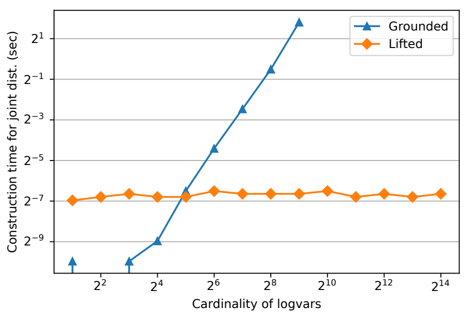
(d) Overlap query answering



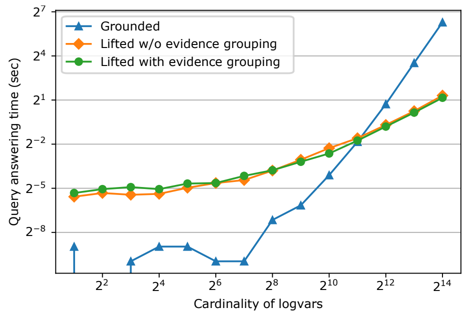
(b) Mixed construction



(e) Mixed query answering



(c) No overlap construction



(f) No overlap query answering

Figure 3.3: Evaluation of experiments

Adding mechanisms to allow for a broader class of queries through combination of lifted and ground treatment would broaden the possible areas for applications. Second, for lifting in general, we assume indistinguishable randvar instances. One interesting question is how to treat not indistinguishable but very similar randvar instances. It would be interesting to understand in which circumstances approximations can fulfil error bounds and be thus a valid alternative for exact query answering. There are existing ideas for understanding error bounds when dealing with changed preconditions by Van den Broeck and Niepert (2015) and Gehrke *et al.* (2020) in discrete environments. Third, current approaches are either lifting discrete PGMs or continuous. Understanding, how to work with hybrid algorithms, e.g., the hybrid version of the junction tree algorithm by Lauritzen and Jensen (2001), could be an interesting path forward.

The upcoming chapter looks into a different cause for complexity in a Gaussian PGM, namely adding a dynamic dimension. We will investigate the relationship between different Gaussian Dynamic PGMs and GPs.

Chapter 4

Gaussian Probabilistic Graphical Models and Gaussian Processes

Relatively recently, Gaussian processes (GPs) have been brought into focus of the machine learning community (Rasmussen, 2006) and have triggered a lot of research (Quinonero-Candela and Rasmussen, 2005; Matthews *et al.*, 2018; Liu *et al.*, 2020). Anologously to Gaussian PGMs, GPs also use Gaussian distributions for modelling the behavior of randvars. The applications of GPs are quite vast including, regression problems, classification problems, multi-output modelling, time series modelling, etc. (Rasmussen, 2006; Roberts *et al.*, 2013; Frigola-Alcalde, 2016; Álvarez *et al.*, 2012).

GPs have a very general structure as we will see in the upcoming sections. Because of this general structure other models have been restated using the GP notation. Examples for approaches or models that have been reformulated as GPs are Bayesian regression, Support Vector Machines, neural networks, and certain types of Kalman filters (Schulz *et al.*, 2018; Seeger, 199; Lee *et al.*, 2018; Reece and Roberts, 2010b). In this chapter, we focus on the relationship between dynamic Gaussian Bayesian network (DGBNs) and GPs for time series modelling. Both models — DGBNs and GPs — use Gaussian distributions for randvars at specific time points. So it is intuitive that if we restate the rules for constructing the probability distribution of a DGBNs in GP notation, we could get an equivalent probability distribution (just with a different form of representation). In contrast to DGBNs, GPs are typically defined on a continuous time dimension and in addition allow direct inference without propagation of evidence through a network. Additionally, an existing GP that models a certain behavior can be easily extended or adapted by making changes to its covariance function, e.g., by combining it with other covariance functions specialized on other characteristics, as we will see later in this chapter. Drawbacks of GPs are that modeling multiple outputs is challenging (Álvarez *et al.*, 2012; Liu *et al.*, 2018) and that modeling multi-outputs with a detailed interpretable (in)dependence structure as it is done in a GBN is currently not possible. Typical multi-output examples come from the field of geostatistics, where the concentration of toxic metals and the pH-value might be modeled as two separate outcomes and influence each other (Álvarez *et al.*, 2019).

This chapter is based on the following publications:

Mattis Hartwig, Marisa Mohr, and Ralf Möller. Constructing Gaussian Processes for Probabilistic Graphical Models. In *FLAIRS-20 Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference.* AAAI Press, 2020

Mattis Hartwig and Ralf Möller. How to Encode Dynamic Gaussian Bayesian Networks as Gaussian Processes? In *AJCAI-20 Proceedings of the Australasian Joint Conference on Artificial Intelligence*, pages 371–382. Springer, 2020

The remainder of this chapter has the following structure. We start by explaining the preliminaries about DGBN and GPs, followed by a discussion of their respective benefits. Afterwards, we discuss related work that draws connections between relation based models and GPs. Then, we construct GPs for three types of DGBNs. We conclude with a discussion of benefits of encoding DGBNs into GPs, and present potential future research.

4.1 Gaussian Process Specific Preliminaries

In this section, we introduce DGBNs, GPs, and kernel functions for GPs. Afterwards, we discuss the advantages of the two models, which also motivates combining them.

4.1.1 Dynamic Gaussian Bayesian Networks

We have introduced GBNs in Section 2.4. DGBNs model the behavior of randvars over time. For specifying time steps, we use the index t. In DGBNs, time is discrete with $t = 0, ..., \tilde{T}$. We use $\tilde{T}$ to avoid confusion with the matrix transpose operation. In this chapter, we assume a Markov property of order 1 for the DGBNs.

Definition 4.1.1 (Order-1 Markov property)**.** A DGBN has an order-1 Markov property if the set of randvars $\mathcal{X}_{t+1}$ at time step $t+1$ is only dependent on the set of randvars $\mathcal{X}_t$ at time step t.

The term *randvar* could stand for each individual node $X_d^{(t)}$ in the DGBN or for an X_d which develops over time. To be consistent with the terminology used in the previous chapters, we use the term randvar for each individual node $X_d^{(t)}$ at a specific time and the term *time-dependent randvar (tdvar)* for variable X_d that develops over time t. So in total, we have a number of D tdvars, a number of $\tilde{T}+1$ time steps, and a number of $N = D \cdot (\tilde{T}+1)$ randvars in the network. We specify three types of DGBN: A one-dimensional Gaussian Markov chain, a Gaussian hidden Markov model, and a two-timeslice DGBN. Illustrations of the three models can be found in Fig. 4.1.

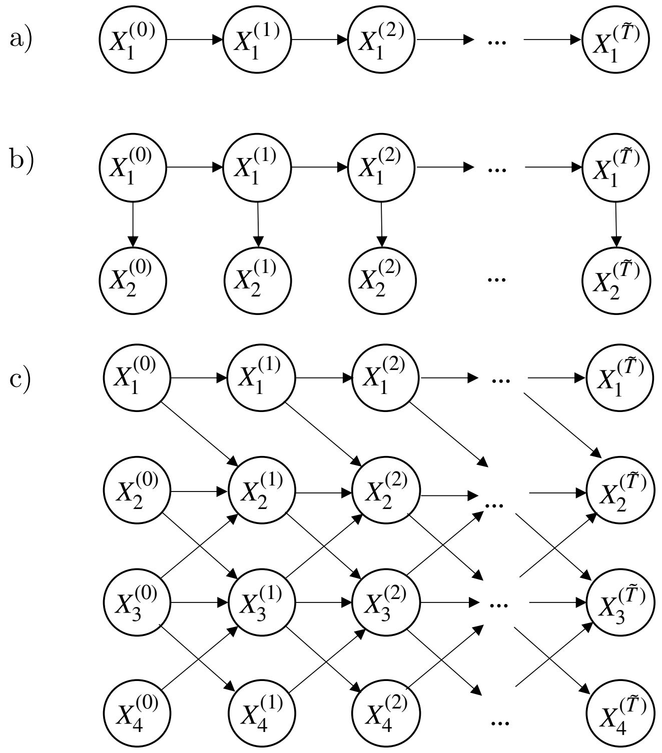


Figure 4.1: Three examples for different types of DGBNs: a) Gaussian Markov chain b) Gaussian hidden Markov model c) two-timeslice DGBN

One-Dimensional Gaussian Markov Chain

A one-dimensional Gaussian Markov chain describes the behavior of one tdvar X_1. It is defined by a starting distribution $P(X^{(0)})$ and a linear transition $\beta_{t,t+1} = \beta_{X_1}$. The conditional distributions of all further time steps are based on Definition 2.4.1 given by

$$P(X^{(t+1)}|X^{(t)}) \sim \mathcal{N}\left(\mu_{X_1} + \beta_{X_1}(x^{(t)} - \mu_{X_1}), \sigma^2_{X_1}\right). \tag{4.1}$$

Gaussian Hidden Markov Model

A two-dimensional Gaussian hidden Markov model describes the behavior of tdvar X_1, that usually cannot directly be measured, but introduces a tdvar X_2, which can be measured. The Gaussian hidden Markov model is defined by a starting distribution $P(X_1^{(0)})$, a linear time transition $\beta_{X_1^{(t)},X_1^{(t+1)}} = \beta_{X_1}$, and a linear dependency β_{X_1,X_2} between $X_1^{(t)}$ and $X_2^{(t)}$. The conditional distributions are based on Definition 2.4.1 and, analogously to Expression (4.1), given by

$$P(X_1^{(t+1)}|X_1^{(t)}) \sim \mathcal{N}\left(\mu_{X_1} + \beta_{X_1}(x_1^{(t)} - \mu_{X_1}), \sigma_{X_1}^2\right) \tag{4.2}$$

and

$$P(X_2^{(t)}|X^{(t)}) \sim \mathcal{N}\left(\mu_{X_2} + \beta_{X_1,X_2}(x_1^{(t)} - \mu_{X_1}), \sigma_{X_2}^2\right). \tag{4.3}$$

Dynamic Gaussian Bayesian Network

The most general case for our description is a DGBN. We follow Murphy (2002) and use a two-timeslice notation, where we have one starting point $P(\mathcal{X}^{(0)})$ defined by the node means and node variances of all tdvars in $\mathcal{X}$, and a transition matrix defining the edges between two time steps. Fig. 4.2 contains an illustration of the two-timeslice notation for the example DGBN in Fig. 4.1c. The conditional distributions based on Definition 2.4.1 and analogously to Expression (4.1) are given by

$$P(X_d^{(t+1)}|\mathcal{X}^{(t)}) \sim \mathcal{N}\left(\mu_{X_d} + \sum_{X_k^{(t)} \in \boldsymbol{Pa}(X_d^{(t+1)})} \beta_{k,d}(x_k^{(t)} - \mu_{X_k}), \sigma_{X_d}^2\right). \tag{4.4}$$

Since the linear relationships do not change over the time, we can store them in a transition matrix $\boldsymbol{M}$, where a non-zero entry at position (d, d') stands for a linear relationship between $X_d^{(t)}$ and $X_{d'}^{(t+1)}$. The transition matrix for the example in Fig. 4.2 is given by

$$\mathbf{M} = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & 0 & 0 \\ 0 & \beta_{2,2} & \beta_{2,3} & 0 \\ 0 & \beta_{3,2} & \beta_{3,3} & \beta_{3,4} \\ 0 & 0 & \beta_{4,3} & 0 \end{bmatrix}. \tag{4.5}$$

For further details on (Gaussian) dynamic BNs, we refer to Murphy (2002).

4.1.2 Gaussian Processes

We define GPs according to Rasmussen (2006).

Definition 4.1.2 (Gaussian Process)**.** A *Gaussian Process* is defined by a mean function $m : \mathbb{T} \to \mathbb{R}$, a kernel function $k : \mathbb{T} \times \mathbb{T} \to \mathbb{R}$, and a spatial dimension $\mathbb{T}$. A *Gaussian Process* describes a collection of randvars, any finite number of which have a joint Gaussian distribution.

Similar to DGBNs, GPs can result in a multivariate Gaussian distribution if they are sampled for a specific set of random variables. In classic GPs, we have one tdvar whose behavior over time t is described. Each element of the randvar set mentioned

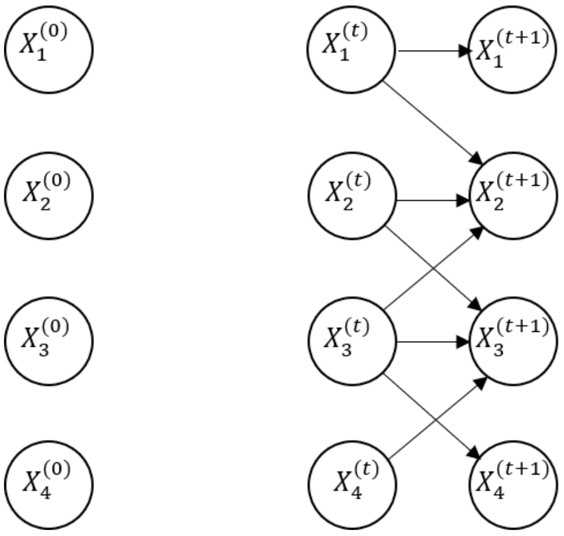


Figure 4.2: A DGBN represented in two-timeslice notation with a transition matrix

in Definition 4.1.2 is thus associated with a time step. One difference between DGBNs and GPs is, that the GP not only defines a specific distribution for a randvar at a specific number of time steps but an infinite-dimensional distribution for randvars at all possible time steps in the feature space of the kernel function (see Definition 4.1.3). By evaluating the mean and covariance function of a GP for multiple time steps $\boldsymbol{t}$, we get the distribution of the randvars at the time steps in $\boldsymbol{t}$:

$$P(\boldsymbol{X_t}) = \mathcal{N}(\boldsymbol{m}(\boldsymbol{t}), \boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t})), \tag{4.6}$$

where $\boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t})$ is a covariance matrix generated by plugging into the covariance function $k(t, t')$ all values in $\boldsymbol{t}$. The result of $\boldsymbol{m}(\boldsymbol{t})$ is a mean vector, whereas the result $\boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t})$ is a covariance function (if k is a valid kernel, see Definition 4.1.3).

Instead of describing the relationships as parent child relations as in DGBNs, we use the covariance function (kernel) to describe the similarity of randvars (here at different points in time). The kernel influences the design of the distributions sampled from a GP. Remark: A GP can also be interpreted as a distribution over functions. The spatial dimension is the input for all functions in the distribution. Thus when evaluating all functions in the distribution for a potential input we get a distribution over outputs instead of a single value (see right side of Fig. 4.4).

The syntax for query answering in a GP is equal to the conditional probability queries in multivariate Gaussian distributions as introduced in Section 2.3. We use a query set $\boldsymbol{Q}$ and an evidence set $\boldsymbol{E}$ containing the queried randvars and evidence randvars respectively. The evidence randvars $\boldsymbol{E}$ and the observed values $\boldsymbol{e}$ form events $\boldsymbol{E} = \boldsymbol{e}$. Additionally, we use $\boldsymbol{t}_Q$ to denote the set of query time points and $\boldsymbol{t}_E$ to denote the set of evidence time points. The evidence time points $\boldsymbol{t}_E$ and the evidence randvars directly correspond to each other in such a way that a randvar in this dynamic model is always associated with a time point. For $\boldsymbol{Q}$ and $\boldsymbol{t}_Q$ it is the same.

The semantic of query answering is that the response should contain the posterior distribution $P(\boldsymbol{Q}|\boldsymbol{E} = \boldsymbol{e})$ over randvars in $\boldsymbol{Q}$ conditioned on the observations $\boldsymbol{E} = \boldsymbol{e}$. To calculate the posterior distribution, we first form the joint probability distribution of the evidence and query randvars using Expression (4.6):

$$P(\boldsymbol{Q}, \boldsymbol{E}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{m}(\boldsymbol{t}_E)\\ \boldsymbol{m}(\boldsymbol{t}_Q)\end{bmatrix}, \begin{bmatrix}\boldsymbol{K}(\boldsymbol{t}_E, \boldsymbol{t}_E) & \boldsymbol{K}(\boldsymbol{t}_E, \boldsymbol{t}_Q)\\ \boldsymbol{K}(\boldsymbol{t}_Q, \boldsymbol{t}_E) & \boldsymbol{K}(\boldsymbol{t}_Q, \boldsymbol{t}_Q)\end{bmatrix}\right). \tag{4.7}$$

where $\boldsymbol{K}(\boldsymbol{t}_E, \boldsymbol{t}_Q)$ is again a covariance matrix generated by evaluating the covariance function k for all combinations of elements in $\boldsymbol{t}_E$ and $\boldsymbol{t}_Q$. Having the joint distribution defined we can use the same algorithm avoiding solving integrals by applying Expressions (2.11) and (2.12) from Section 2.3. The posterior distribution is then given by

$$P(\boldsymbol{Q}|\boldsymbol{E} = \boldsymbol{e}) = N(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*), \tag{4.8}$$

where

$$\boldsymbol{\mu}^* = \boldsymbol{m}(\boldsymbol{t}_Q) + \boldsymbol{K}(\boldsymbol{t}_Q, \boldsymbol{t}_E)\boldsymbol{K}(\boldsymbol{t}_E, \boldsymbol{t}_E)^{-1}(\mathrm{e} - \boldsymbol{m}(\boldsymbol{t}_E)) \tag{4.9}$$

and

$$\boldsymbol{\Sigma}^* = \boldsymbol{K}(\boldsymbol{t}_Q, \boldsymbol{t}_Q) - \boldsymbol{K}(\boldsymbol{t}_Q, \boldsymbol{t}_E)\boldsymbol{K}(\boldsymbol{t}_E, \boldsymbol{t}_E)^{-1}\boldsymbol{K}(\boldsymbol{t}_E, \boldsymbol{t}_Q). \tag{4.10}$$

Because we use functions to generate the covariance matrix and the mean vector, we can use any number of points on the spatial axis in the query. The posterior distribution is always based on a discrete number of points, but the number of points can be chosen freely. Fig. 4.3, inspired by Roberts *et al.* (2013), illustrates how the posterior as of a GP with three evidence points can be generated over a) a relatively sparse query set ($\boldsymbol{t}_Q = 1, 3, 4, 5, 7, 9, 10$) or over b) a very a dense query set (1000 steps in the interval between 0 and 10), looking like continuity. The grey area in Fig. 4.3 is corresponding to the confidence interval of two standard deviations and the line (or dots) in the middle of the confidence interval are the mean values in that respective time step. The red-dots refer to observed values. When looking closely even observed values can have a uncertainty, which depends on the modelling choices in the covariance function.

4.1.3 Kernel Functions

As mentioned above, kernel functions are used to construct the covariance matrix between any features (in our case timepoints) in a GP. We define valid kernels based on Rasmussen (2006).

Definition 4.1.3 (Valid kernel functions)**.** A *valid kernel function* (or just *kernel*) $k : \mathbb{T} \times \mathbb{T} \to \mathbb{R}$ for a GP, where $\mathbb{T}$ is the feature space, needs to fulfil two characteristics:

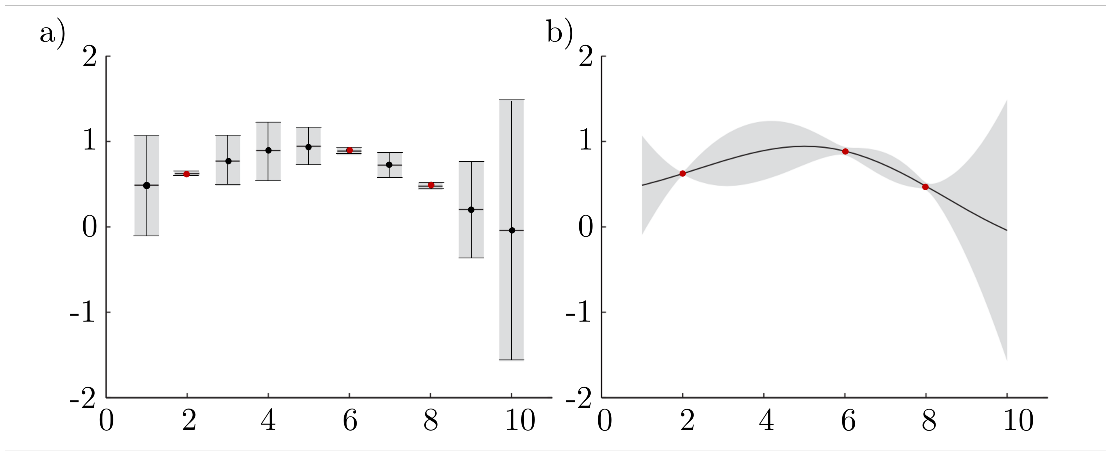


Figure 4.3: A GP with 3 evidence points at timepoints 2, 6, and 8, where a) contains a sparse query set ($\boldsymbol{t}_Q = 1, 3, 4, 5, 7, 9, 10$) of the posterior and b) a relatively continuous one (1000 steps in the interval between 0 and 10) (Roberts *et al.*, 2013)

- any resulting matrix needs to be symmetric, i.e., $k(t, t') = k(t', t)$ for all t and t',
- any resulting matrix needs to be positive semi-definite, i.e., symmetric and inquality $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(t_i, t_j) \geq 0$ for $n \in \mathbb{N}$, $t_1, ..., t_n \in \mathbb{T}$, $c_1, ..., c_n \in \mathbb{R}$ must be true.

In literature, a kernel is also called symmetric or semi-definite if it fulfils the conditions.

Kernels define how the values at different points influence each other and thus define how the posterior distributions look like. For GPs, the feature space is the spatial dimension of the GP. The most-used kernel in GP use-cases is the squared exponential kernel

$$k_{se}(t, t') = \sigma^2 \exp\left(-\frac{(t - t')^2}{2\ell^2}\right), \tag{4.11}$$

where σ^2 and ℓ are hyperparameters for signal noise and lengthscale (roughly speaking how strongly the feature's associated value changes) respectively. Another kernel that is often used is the so-called linear kernel

$$k_{lin}(t, t') = \sigma_b^2 + \sigma^2(t - c) \cdot (t' - c), \tag{4.12}$$

where c, σ_b^2, and σ^2 are hyperparameters for the offset (roughly speaking the x-coordinate that all the lines in the posterior cross), the uncertainty of the offset, and the variance respectively. A kernel for modeling periodic behavior is the periodic kernel

$$k_{per}(t, t') = \sigma^2 \exp\left(-\frac{2}{\ell^2} \sin^2\left(\pi \frac{|t - t'|}{p}\right)\right), \tag{4.13}$$

where σ^2, ℓ, and p are hyperparameters for signal noise, lengthscale, and period length (the distance after which values are repeated) respectively. Example plots of how kernels influence the shape of the posterior distribution for all three kernels can be found in Fig. 4.4.

Valid kernels can be constructed using other kernels. Bishop (2006) lists valid operations for constructing kernels. We use the following subset of these operations in later sections.

Lemma 4.1.1. *Given valid kernels $k_1(t,t')$, $k_2(t,t')$, and a constant $c \in \mathbb{R}$, the following constructed kernels are also valid:*

$$k(t,t') = ck_1(t,t'), \tag{4.14}$$
$$k(t,t') = k_1(t,t') + k_2(t,t'), \tag{4.15}$$
$$k(t,t') = \exp(k_1(t,t')), \tag{4.16}$$
$$k(t,t') = k_1(t,t')k_2(t,t'). \tag{4.17}$$

The feature space $\mathbb{T}$ in this dissertation is the temporal space and thus continuous $\mathbb{T} = \mathbb{R}$ if not stated otherwise. In the following sections, we will assume to have a temporal space even though the rules would also apply for other feature spaces.

4.1.4 Multi-output Kernels

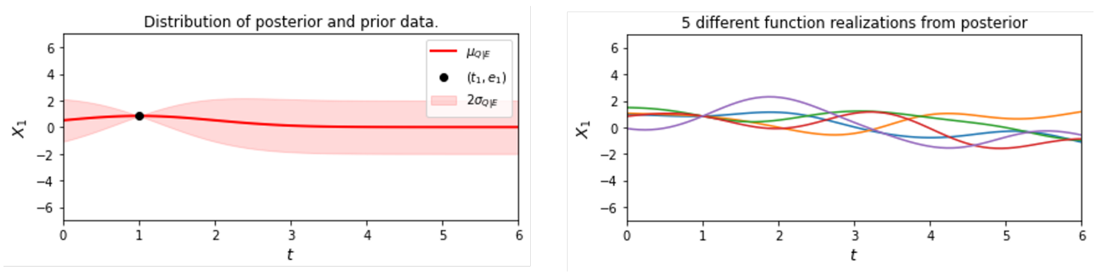
As mentioned above, classic GPs model the behavior of the values of a single tdvar that develops time. Multi-output GPs model more than one tdvar over time . In our case, we would like to model an arbitrary number of tdvars in a DGBN over the time dimension. To model a number of D tdvars, a multi-output kernel $k : \mathbb{T} \times \mathbb{T} \to \mathbb{R}^D$, which can construct covariance matrices for multi-outputs, is needed. If we look at $\tilde{T}$ different points, the covariance matrix in the one-dimensional setup is $\tilde{T} \times \tilde{T}$-dimensional.With D output dimensions (tdvars), the covariance matrix is $D\tilde{T} \times D\tilde{T}$-dimensional. An illustration of the covariance matrix design can be found in Fig. 4.5.

A general notation for a multi-output kernel is

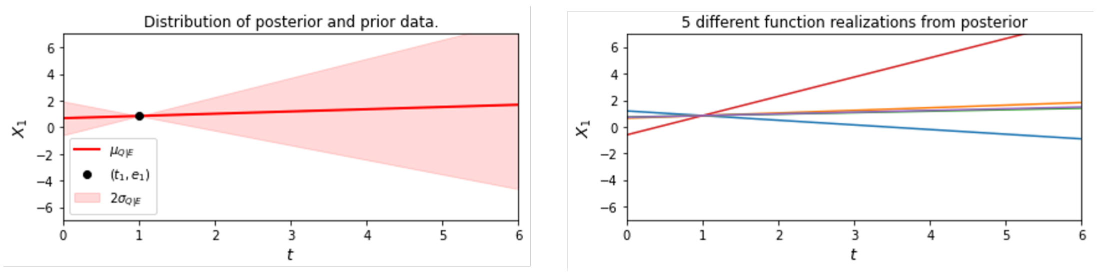
$$k((d,t),(d',t')), \tag{4.18}$$

where d is referring to a tdvar X_d and t to a point in time. The kernel allows us to calculate the covariance between any tdvar X_d at any time t with any other variable $X_{d'}$ at time t'.

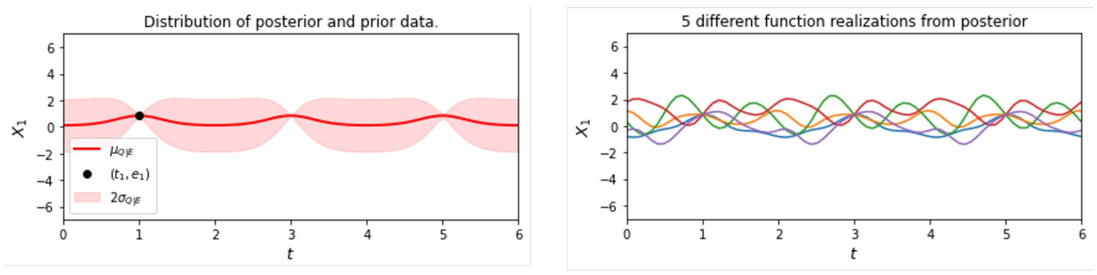
In general, multi-output kernels can have any kind of structure but need to fulfil the conditions in Definition 4.1.3. Álvarez *et al.* (2012) has defined two special types of multi-output kernels that fulfil the conditions Definition 4.1.3, namely separable kernels and sum of separable kernels. We will use those types later to prove the validity of the constructed kernel for the Gaussian hidden Markov model.



(a) Squared exponential kernel



(b) Linear kernel



(c) Periodic kernel

Figure 4.4: Illustration of the influence of kernels on the posterior distribution generated by a GP with one point of evidence at $t_E = 1$. The red area corresponds to the confidence interval of two standard deviations and the read line corresponds to the most probable assignment (the means at each timepoint in the posterior).

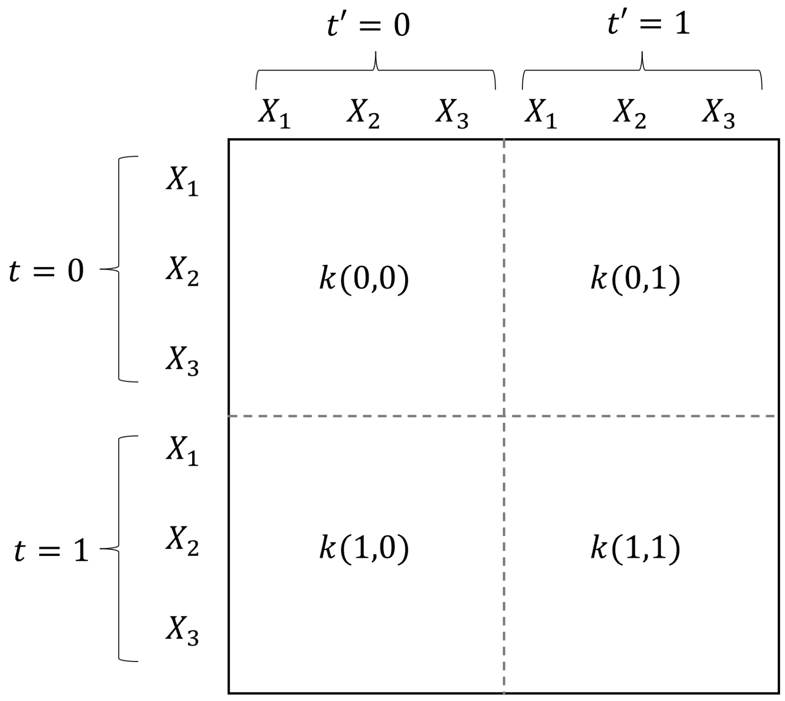


Figure 4.5: Illustration of multi-output covariance matrices for two example timesteps

Definition 4.1.4 (Separable kernel)**.** A multi-output kernel with output dimension d and spatial dimension t is *separable* if it can be separated in one kernel being dependent on d and one being dependent on t, i.e.,

$$k((d,t),(d',t')), = k(t,t')k(d,d'). \tag{4.19}$$

Definition 4.1.5 (Sum of separable kernels)**.** A kernel that is constructed by a *sum of separable kernels* has a structure given by

$$k(t,t') = \sum_{r=1}^{R} k_r(t,t')k_r(d,d') = \sum_{r=1}^{R} k_r(t,t')\boldsymbol{B}_r, \tag{4.20}$$

where $\boldsymbol{B}_r$ is a $D \times D$-dimensional symmetric and positive semi-definite matrix and $R \in \mathbb{N}$ is the number of summands.

The purpose of the $\boldsymbol{B}$ matrices is to describe the relationships between the different dimensions (tdvars). The simplest setting is if $\boldsymbol{B}_r$ is a diagonal matrix. Then, all outputs are treated in a completely independent way. In our DGBN, this would be the case if there were no connections between any of the different randvars $X_d^{(t)}$ and $X_{d'}^{(t+1)}$ for $d \neq d'$, i.e. no connection between randvars from one time slice to the next one.

4.1.5 Comparing Probabilistic Graphical Models to Gaussian Processes

DGBNs have several benefits that make them straightforward to use. One benefit is that they can capture (conditional) dependencies and independencies of the randvars very intuitively (Koller and Friedman, 2009), i.e. DGBNs can incorporate expert knowledge. It is possible to construct a network entirely by expert knowledge but it is also possible to use expert knowledge as a prior for a probability distribution (Flores *et al.*, 2011). Gaussian Hidden Markov models and DGBNs can model a probability distribution over multiple randvars on a discrete time dimension. Last but not least, PGMs have already been used in many applications. Therefore a wide range of inference and learning algorithms have been developed (Koller and Friedman, 2009).

Using GPs also has benefits. GPs have a continuous spatial dimension which allows for modeling continuous changes directly and without the need of discretization. GPs are nonparametric in a sense that they are not fixed in the numbers of parameters they learn but rather use a flexible number of observations. Additionally, GPs directly incorporate a quantification of uncertainty. Because of their joint Gaussian characteristics, calculating posterior distributions is straightforward and relatively efficient (Roberts *et al.*, 2013).

Converting PGMs to GPs while retaining the benefits described above is a promising approach that we pursue in this chapter to exploit the benefits of both approaches.

4.2 Related Work on the Relationship Between PGMs and GPs

There have been three different streams to bring PGMs together with GPs. One research stream known as relation learning uses multiple GPs to identify probabilistic relations or links within sets of entities (Xu *et al.*, 2009; Yu *et al.*, 2006). A second research stream uses GPs for transition functions in state space models. Frigola-Alcalde (2016) has investigated different techniques for learning state space models that have GP priors over their transition functions. Turner (2012) has explored change point detection in state space models using GPs. A third research stream focuses on constructing covariance functions for GPs to mimic certain behaviors from other models. Reece and Roberts (2010b) have shown that they can convert a specific Kalman filter model for the near constant acceleration model into a kernel function for a GP and then Reece and Roberts (2010a) combine that kernel function with other known kernels to get better results for predictions in target tracking. Rasmussen (2006) has done a lot of work to describe the relationship between stationary auto-regressive models and also described the kernel for a non-stationary Wiener process (the min kernel) that we reuse in this chapter as well. The results of this chapter contribute to the third research stream by providing a novel approach to converting three types of DGBNs into GPs, namely the one-dimensional Gaussian Markov chain, the Gaussian hidden Markov model, and the two-timeslice DGBN.

4.3 Gaussian Processes for Gaussian Markov Chains

In this section, we construct a GP for the Gaussian Markov chain model. We start by constructing a kernel function, continue with proving its validity, and end by describing the full GP.

4.3.1 Constructing the Kernel

We reuse a contribution in the paper by Shachter and Kenley (1989), which we have already used in the previous chapter as a reference for constructing covariance matrices of GBNs. To prove correctness of their inductive approach for constructing the covariance matrix, Shachter and Kenley (1989) have used the following lemma that we reuse for constructing the kernel function. Since the GP starts at $t = 0$, we also use zero as a starting index in the GBN (in contrast to the previous section where we used a starting index of one).

Lemma 4.3.1. *For $N+1$ topologically ordered randvars $X_i \in \mathcal{X}$, $i = 0, \ldots, N$ in a GBN, let σ_i^2 be the node variance of X_i and $\boldsymbol{T} \in \mathbb{R}^{N+1 \times N+1}$ be a matrix, where the entries $\beta_{i,k}$, describe the linear relationship between a parent X_i and its child X_k. If X_i is no parent of X_k, the entry is zero. For a fixed $j \in \{0, \ldots, N\}$, let $\boldsymbol{\Sigma}_{0:j,0:j}$ be the covariance matrix between all randvars X_0 to X_j, and $\boldsymbol{T}_{\boldsymbol{s},j} \in \mathbb{R}^{j-1 \times 1}$, $\boldsymbol{s} = (0, \ldots, j-1)$, the corresponding part of $\boldsymbol{T}$. We define the following two matrices*

$$\boldsymbol{S}_j := \begin{bmatrix} \boldsymbol{\Sigma}_{0:j,0:j} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \sigma_{j+1}^2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & 0 & \ldots & \sigma_N^2 \end{bmatrix}, \tag{4.21}$$

$$\boldsymbol{U}_j := \begin{bmatrix} \mathbf{I}_{j \times j} & \boldsymbol{T}_{\boldsymbol{s},j} & \mathbf{0} \\ \mathbf{0} & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{N-j \times N-j} \end{bmatrix}. \tag{4.22}$$

Let $\boldsymbol{D}$ be the diagonal matrix containing the node variances. Then it holds that

$$\boldsymbol{\Sigma} = \boldsymbol{S}_N = \boldsymbol{U}_N^T ... \boldsymbol{U}_0^T \boldsymbol{D} \boldsymbol{U}_0 ... \boldsymbol{U}_N, \tag{4.23}$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{N+1 \times N+1}$ is the covariance matrix of the equivalent multivariate Gaussian distribution for the above defined GBN.

To define a GP, a kernel function must be constructed that maps arbitrary time points t and t' to a covariance value. Therefore, we convert the inductive multiplication of the matrices in Expression (4.23) into a kernel function.

In the case of a Markov chain, each value on the diagonal of $\boldsymbol{D}$ is $\sigma^2_{X_1}$ and $N = \tilde{T}$. The $(\tilde{T}+1) \times (\tilde{T}+1)$ dimensional matrix $\boldsymbol{T}$ has entries β at the positions $(s, s+1)$ for $s = 0, ..., \tilde{T}$, which describe the linear relationship along the time dimension of X_1, and zeros everywhere else:

$$\boldsymbol{T} = \begin{bmatrix} 0 & \beta_{X_1} & 0 & \dots & 0 \\ 0 & 0 & \beta_{X_1} & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \beta_{X_1} \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}. \tag{4.24}$$

Consequently, the matrix $\boldsymbol{U}_i$ is the identity matrix of size $\tilde{T}+1 \times \tilde{T}+1$ plus the β_{X_1} at position $(i, i+1)$. By multiplying all $\boldsymbol{U}$-matrices as indicated above, we obtain the diagonal matrix

$$\prod_{i=0}^{\tilde{T}} \boldsymbol{U}_i = \begin{bmatrix} 1 & \beta_{X_1} & \beta^2_{X_1} & \dots & \beta^{\tilde{T}}_{X_1} \\ 0 & 1 & \beta_{X_1} & \dots & \beta^{(\tilde{T}-1)}_{X_1} \\ 0 & 0 & 1 & \dots & \beta^{(\tilde{T}-2)}_{X_1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}. \tag{4.25}$$

The same applies for the left part of the Expression (4.23):

$$\prod_{i=0}^{\tilde{T}} \boldsymbol{U}_i^T = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ \beta_{X_1} & 1 & 0 & \dots & 0 \\ \beta^2_{X_1} & \beta_{X_1} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta^{\tilde{T}}_{X_1} & \beta^{\tilde{T}-1}_{X_1} & \beta^{\tilde{T}-2}_{X_1} & \dots & 1 \end{bmatrix}. \tag{4.26}$$

Since all values on the diagonal-matrix $\boldsymbol{D}$ have the same scalar value $\sigma^2_{X_1}$, we can move $\sigma^2_{X_1}$ at the beginning of the equation, resulting in

$$\boldsymbol{\Sigma} = \boldsymbol{U}_{\tilde{T}}^T ... \boldsymbol{U}_0^T \boldsymbol{D} \boldsymbol{U}_0 ... \boldsymbol{U}_{\tilde{T}} = \sigma^2_{X_1} \prod_{i=0}^{\tilde{T}} \boldsymbol{U}_i^T \prod_{i=0}^{\tilde{T}} \boldsymbol{U}_i. \tag{4.27}$$

To come up with a kernel, we need a function that results in the correct element in the covariance matrix. Based on matrix algebra the item at position (i, j) can be calculated by multiplying row i from Expression (4.26) with column j from Expression (4.25). For $i = j$ (the diagonal of the resulting covariance matrix), we calculate the covariance with

$$\begin{bmatrix} \beta_{X_1}^{i} \\ \beta_{X_1}^{i-1} \\ \beta_{X_1}^{i-2} \\ \beta_{X_1}^{i-3} \\ \vdots \\ \beta_{X_1}^{0} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \cdot \begin{bmatrix} \beta_{X_1}^{i} \\ \beta_{X_1}^{i-1} \\ \beta_{X_1}^{i-2} \\ \beta_{X_1}^{i-3} \\ \vdots \\ \beta_{X_1}^{0} \\ 0 \\ \vdots \\ 0 \end{bmatrix}^T = \sum_{k=0}^{i} (\beta_{X_1}^{k})^2. \tag{4.28}$$

For $i \neq j$, we denote the difference as $\Delta = |i-j|$. Because of the symmetry of matrices in Expressions (4.25) and (4.26) there is no difference between the cases $j > i$ and $j < i$. Let $j > i$, then the j-th vector contains more elements and in addition elements with higher exponents. These changing values are only relevant in the first i entries because all other entries are multiplied with zero. Substituting j by $i + \Delta$ results in:

$$\begin{bmatrix} \beta_{X_1}^{i} \\ \beta_{X_1}^{i-1} \\ \beta_{X_1}^{i-2} \\ \vdots \\ \beta_{X_1}^{0} \end{bmatrix} \cdot \begin{bmatrix} \beta_{X_1}^{j} \\ \beta_{X_1}^{j-1} \\ \beta_{X_1}^{j-2} \\ \vdots \\ \beta_{X_1}^{j-i} \end{bmatrix}^T = \begin{bmatrix} \beta_{X_1}^{i} \\ \beta_{X_1}^{i-1} \\ \beta_{X_1}^{i-2} \\ \vdots \\ \beta_{X_1}^{0} \end{bmatrix} \cdot \begin{bmatrix} \beta_{X_1}^{i+\Delta} \\ \beta_{X_1}^{i+\Delta-1} \\ \beta_{X_1}^{i+\Delta-2} \\ \vdots \\ \beta_{X_1}^{\Delta} \end{bmatrix}^T \tag{4.29}$$
$$= \sum_{k=0}^{i} \beta_{X_1}^{k} \beta_{X_1}^{k+\Delta} = \beta_{X_1}^{\Delta} \sum_{k=0}^{i} (\beta_{X_1}^{k})^2.$$

To ensure symmetry of the kernel, we also need to include the case $i > j$, which leads to replacing i by $\min(i, j)$ and Δ by $|i - j|$ in Expression (4.29), resulting in

$$\beta_{X_1}^{|i-j|} \sum_{k=0}^{min(i,j)} (\beta_{X_1}^{k})^2. \tag{4.30}$$

Expression (4.30) has the structure of a partial sum of a geometric series.

Definition 4.3.1 (Geometric series, partial sum). A *geometric series* $\sum_k a_k$ is a series for which the ratio of each two consecutive terms a_{k+1}/a_k is a constant function of the summation index k. A *partial sum* of the first $n + 1$ elements of a geometric series can be calculated in closed form by

$$\sum_{k=0}^{n} br^k = b\left(\frac{1 - r^{n+1}}{1 - r}\right). \tag{4.31}$$

We reformulate Expression (4.30) using the formula for the partial sum of a geometric series from Expression (4.31) and replace i and j with t and t' respectively:

$$k(t,t') = \sigma_{X_1}^2 \beta_{X_1}^{|t-t'|} \frac{1-\beta_{X_1}^{2\min(t,t')+2}}{1-\beta_{X_1}^2}. \tag{4.32}$$

By using the partial sum of a geometric series we can only work with $\beta \neq 1$. That is not an issue because a value of $\beta = 1$ the sum in Expression (4.30) would simplify to the min-kernel, which is a known kernel for the Wiener process Rasmussen (2006). By converting the approach of Shachter and Kenley (1989) into a closed formula, we have shown that Expression (4.32) can be used to construct a covariance matrix that encodes the characteristics of the one-dimensional Gaussian Markov chain. Next, we show that the Expression (4.32) is indeed a valid kernel.

4.3.2 Kernel Validity

To prove the validity of Expression (4.32) as a kernel function for a GP, we use the characteristics that kernels can be constructed of other kernels using the equations from Lemma 4.1.1. This additional step is necessary because we broaden the feature space from $\mathbb{T} = \mathbb{N}$ to $\mathbb{T} = \mathbb{R}$. We show that the two factors

$$k_a(t,t') = \sigma_{X_1}^2 \beta_{X_1}^{|t-t'|} \text{and} \tag{4.33}$$

$$k_b(t,t') = \frac{1-\beta_{X_1}^{2\min(t,t')+2}}{1-\beta_{X_1}^2} \tag{4.34}$$

of the Expression (4.32) are valid kernels. Then, based on Expression (4.17), our constructed kernel being the product of the two factors is a valid kernel as well. The exponent $|t-t|'$ is the one dimensional case of the Euclidean distance kernel (Bishop, 2006). With Expression (4.14) and the exponential rule from Expression (4.16) $k_a(t,t')$ is a valid kernel.

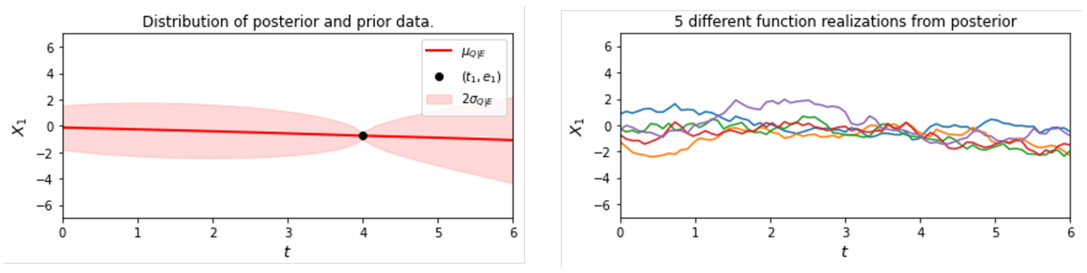
To show that the function $k_b(t,t')$ is a valid kernel, we use the summation in Expression (4.29). Expression (4.29) converts the fraction back into a sum of exponential functions that have a positive numbers $n = 1, ..., \min(t,t')$ as the exponents. Based on Expression (4.15), Expression (4.16), and the fact that $\min(t,t')$ is a valid kernel (Rasmussen, 2006), the function $k_b(t,t')$ is a valid kernel. Consequently, Expression (4.32) is a valid kernel for a GP.

4.3.3 Defining the Gaussian Process

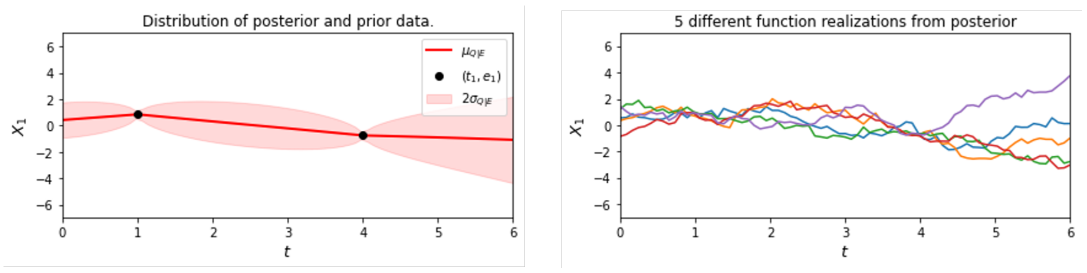
Having constructed a valid kernel function, we also need a mean function for the GP. Since the mean stays the same in a DGBN, we have a constant mean function

$$m(t) = \mu_{X_1}. \tag{4.35}$$

With the covariance function from Expression (4.32) and the mean function from Expression (4.35), we have constructed the desired GP for a Gaussian Markov chain. The hyperparameters for the constructed GP are β_{X_1}, $\sigma^2_{X_1}$, and μ_{X_1}. Example plots with hyperparameters $\beta_{X_1} = 1.2$, $\sigma^2_{X_1} = 1.2$, and $\mu_{X_1} = 0$ can be found in Fig. 4.6. Fig. 4.6 also shows the Markov property of the kernel, because an additional evidence point at $t = 1$ in Fig. 4.6 (b) does not change the distribution after the point $t = 4$, when comparing it to Fig. 4.6 (a).



(a) Evidence for $t_E = 4$



(b) Evidence for $\boldsymbol{t}_E = \{1, 4\}$

Figure 4.6: Illustration of the constructed GP for one-dimensional Gaussian Markov chains

We have generalized the feature space from $\mathbb{N}$ to $\mathbb{R}$ in a way, that the probability distribution of a one-dimensional Gaussian Markov chain can be also sampled using the constructed GP. Furthermore, we can now sample also distributions including randvars at fractional or real time points. Of course we have made the assumption that the observed behavior formalized in Expression (4.32) also happens in between discrete time points. This assumption effects in no way the distributions that we get by putting in discrete time points and thus allows for generalizing without making compromises on the original one-dimensional Gaussian Markov chain. Next, we look at how to describe the Gaussian hidden Markov model as a GP.

4.4 Gaussian Processes for Hidden Markov Models

In a Gaussian hidden Markov model, there are two tdvars X_1 and X_2. We have Expressions (4.2) and (4.3) for the distributions over X_1 and X_2. Similar to the previous section, we start with constructing the kernel function, continue with proving its validity and finish with defining the full resulting GP including its hyperparameters.

4.4.1 Constructing the Kernel

We generalize the kernel from the previous section to support the two randvars as outputs. We use the general form of the multi-output kernel from Expression (4.18), where d can be 1 or 2. We know that based on Expression (2.15) that the covariance is always influenced by parent-child relationships. Since the introduction of X_2 has no influence on the parent structure of X_1, the covariance of X_1 developing over t is unchanged. We can write:

$$k((1,t),(1,t')) = k(t,t') = \sigma^2_{X_1}\beta^{|t-t'|}_{X_1}\frac{1-\beta^{2\min(t,t')+2}_{X_1}}{1-\beta^2_{X_1}}. \tag{4.36}$$

For the kernel $k((1,t),(2,t'))$, we know that each $X_2^{t'}$ has a parent of $X_1^{t'}$ which has the same linear relationship β_{X_1,X_2}. We can thus construct $k((1,t),(2,t'))$ by using Expressions (2.15) and (4.36):

$$k((2,t'),(1,t)) = k((1,t),(2,t')) = k((1,t)(1,t'))\beta_{X_1,X_2}. \tag{4.37}$$

The last part that needs to be constructed is the covariance $k((2,t),(2,t'))$ between two X_2 randvars. With Expression (2.15) and with the knowledge that the only parent of X_2^t is X_1^t the covariance is given by

$$\begin{aligned} k(2,t),(2,t')) &= k((2,t)(1,t'))\beta_{X_1,X_2} + \delta_{t,t'}\sigma^2_X \\ &= k((1,t)(1,t'))\beta^2_{X_1,X_2} + \delta_{t,t'}\sigma^2_X. \end{aligned} \tag{4.38}$$

Combining Expressions (4.36) to (4.38) results in the final kernel

$$k((d,t),(d',t')) = \begin{cases} k((1,t)(1,t')), & \text{if } d,d' = 1, \\ k((1,t)(1,t'))\beta^2_{X_1,X_2}, & \text{if } d \neq d', \\ k((1,t)(1,t'))\beta^2_{X_1,X_2} + \delta_{tt'}\sigma^2_{X_2}, & \text{if } d,d' = 2. \end{cases} \tag{4.39}$$

4.4.2 Kernel Validity

To prove validity, we convert the kernel from Expression (4.39) into the structure of a sum of separable kernels following Definition 4.1.5. We rewrite the cases as individual kernels such that the kernels are returning zero, when the case condition is not met:

$$k_1(d,d') = \epsilon_{d,d',1}, \tag{4.40}$$

$$k_2(d,d') = 1 - \delta_{d,d'}, \text{ and} \tag{4.41}$$

$$k_3(d,d') = \epsilon_{d,d',2}, \tag{4.42}$$

where ϵ_{d,d',d^*} is equal to 1 if $d = d' = d^*$ and otherwise zero. Expressions (4.40) to (4.42) are valid kernels themselves because they are symmetric and their eigenvalues are $\{1,0\}$, $\{0,1\}$, and $\{1,1\}$ respectively, making them positive semi-definite.

With Expressions (4.40) to (4.42), Expression (4.39) can be rewritten to

$$\begin{aligned} k((d,t),(d',t')) = {} & k_1(d,d')k((1,t)(1,t')) \\ & + k_2(d,d')k((1,t)(1,t'))\beta_{X_1,X_2} \\ & + k_3(d,d')(k((1,t)(1,t'))\beta^2_{X_1,X_2} + \delta_{t,t'}\sigma^2_{X_2}). \end{aligned} \tag{4.43}$$

The sum of separable kernels structure in Expression (4.43) proves that the constructed kernel is a valid kernel to be used in a GP.

4.4.3 Defining the Gaussian Process

As in the previous case, constructing the mean function is fairly simple. Analogously to the single-variable case, the mean functions for X_1 and X_2 are constant resulting in

$$m(d,t) = \begin{cases} \mu_{X_1}, & \text{if } d = 1, \\ \mu_{X_1}, & \text{if } d = 2. \end{cases} \tag{4.44}$$

For the constructed GP, we have a set of hyperparameters $\theta = \{\mu_{X_1}, \mu_{X_2}, \beta_{X_1}, \beta_{X_1,X_2}\}$. Again we have generalized the feature space from $\mathbb{N}$ to $\mathbb{R}$ and made the assumption that the behavior described by the kernel in Expression (4.39) is also happening in fractional and real time steps. Next, we look at how to described DGBNs as GPs.

4.5 Gaussian Processes for Dynamic Gaussian Bayesian Networks

In a DGBN, we have tdvars $X_1, ..., X_D$ evolving over time $t = 0, ..., \tilde{T}$. Again, we start by constructing a kernel. In contrast to the two previous sections the constructed kernel includes operations that cannot be easily generalized to a continuous time dimension. Therefore, we have a discussion on needed next steps to allow for continuity. We follow-up by a quick validity discussion which is not problematic here, because we do not change the feature space and conclude with defining the resulting GP.

4.5.1 Constructing the Kernel

Again, we use Lemma 4.3.1 to find a general formula for the multi-output kernel. Since the σ^2-values for all tdvars stay constant over time, the matrix $\boldsymbol{D}$ has repeating diagonal entries every D entries. We denote $\mathbf{A}$ for all $D \times D$-dimensional blocks that are on the diagonal of $\boldsymbol{D}$. The block $\mathbf{A}$ itself is constructed by

$$\mathbf{A} = diag(\sigma^2_{X_1}, ..., \sigma^2_{X_D}). \tag{4.45}$$

The $N+1 \times N+1$-dimensional matrix $\boldsymbol{T}$ from Lemma 4.3.1, containing all linear relationships in the DGBN, has the $\tilde{T}+1 \times \tilde{T}+1$-dimensional block structure visualized in Fig. 4.5. Using the transition matrix $\mathbf{M}$ for transitions of tdvars from one time step to the next results in

$$\boldsymbol{T} = \begin{bmatrix} \mathbf{0} & \mathbf{M} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{M} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \vdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{M} \end{bmatrix}. \tag{4.46}$$

Given structure for the matrix $\boldsymbol{T}$, we can group the multiplications of matrices $\boldsymbol{U}$ from Expression (4.23) in Lemma 4.3.1. With D dimensions, we can multiply D consecutive matrices from $\boldsymbol{U}_t$ to $\boldsymbol{U}_{t+D-1}$ that would belong to the randvars within one time step:

$$\mathbf{O}_t = \prod_{i=tD}^{tD+(D-1)} \mathbf{U}_i = \begin{bmatrix} \mathbf{I}_{tD} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_D & \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_D & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{(\tilde{T}-t-1)D} \end{bmatrix}. \tag{4.47}$$

With block matrix multiplication and the construction of matrices $\mathbf{O}_t$, we can reformulate the multiplication from Lemma 4.3.1 into

$$\prod_{i=0}^{N} \boldsymbol{U}_i = \prod_{t=0}^{\tilde{T}} \boldsymbol{O}_t = \begin{bmatrix} \mathbf{I} & \mathbf{M} & \mathbf{M}^2 & \dots & \mathbf{M}^{\tilde{T}} \\ \mathbf{0} & \mathbf{I} & \mathbf{M} & \dots & \mathbf{M}^{\tilde{T}-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \dots & \mathbf{M}^{\tilde{T}-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} \end{bmatrix}. \tag{4.48}$$

The full $N+1 \times N+1$ covariance matrix would be calculated by using Expression (4.23). In our kernel function, we only want to calculate the $D \times D$ block containing the covariances between two time steps t and t'. We would get this matrix by multiplying the t-th row of blocks from $\mathbf{U}_N^T ... \mathbf{U}_0^T$, with the t'-th column of $\boldsymbol{D}\mathbf{U}_0 ... \mathbf{U}_N$. If $t = t'$, we have

$$\begin{bmatrix} \mathbf{M}^t \\ \mathbf{M}^{t-1} \\ \mathbf{M}^{t-2} \\ \vdots \\ \mathbf{M}^0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{AM}^t \\ \mathbf{AM}^{t-1} \\ \mathbf{AM}^{t-2} \\ \vdots \\ \mathbf{AM}^0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} = \sum_{k=0}^{t} (\mathbf{M}^T)^k \mathbf{AM}^k, \tag{4.49}$$

where $\mathbf{M}^0 = \mathbf{I}$.

In the case $t \neq t'$, we start with looking at $t < t'$. The t-th row of blocks contains less non-zero blocks than the t'-th. With block matrix multiplication, only the first t blocks are multiplied with non-zero matrices, making the other blocks irrelevant. The blocks in the t'-th column also have higher exponents, where the difference is given by $t' - t$. Thus, the multiplication of the t-th row of blocks with the t'-th column of blocks results in

$$\begin{bmatrix} \mathbf{M}^t \\ \mathbf{M}^{t-1} \\ \vdots \\ \mathbf{M}^0 \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{AM}^{t'} \\ \mathbf{AM}^{t'-1} \\ \vdots \\ \mathbf{AM}^{t'-t} \end{bmatrix} = \sum_{k=0}^{t} (\mathbf{M}^T)^k \mathbf{AM}^{k+(t'-t)} = \left(\sum_{k=0}^{t} (\mathbf{M}^T)^k \mathbf{AM}^k \right) \mathbf{M}^{t'-t} \tag{4.50}$$

For the case $t > t'$, the t-th row contains more non-zero blocks than the t'-column. Analogously to Expression (4.50) the multiplication results in

$$\begin{bmatrix} \mathbf{M}^t \\ \mathbf{M}^{t-1} \\ \vdots \\ \mathbf{M}^{t-t'} \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{AM}^{t'} \\ \mathbf{AM}^{t'-1} \\ \vdots \\ \mathbf{AM}^0 \end{bmatrix} = \sum_{k=0}^{t'} (\mathbf{M}^T)^{k+(t-t')} \mathbf{AM}^k = \mathbf{M}^{T^{t-t'}} \sum_{k=0}^{t'} (\mathbf{M}^T)^k \mathbf{AM}^k. \tag{4.51}$$

With the transpose multiplication rule

$$(\boldsymbol{U} \cdot \boldsymbol{V})^T = \boldsymbol{V}^T \cdot \boldsymbol{U}^T \tag{4.52}$$

it is clear that Expression (4.51) is the transpose of Expression (4.50), which is also expected based on the symmetry conditions on the covariance matrix.

Combining Expression (4.50) and Expression (4.51) results in a kernel function of

$$k(t, t') = \begin{cases} \left(\sum_{k=0}^{min(t,t')} (\mathbf{M}^T)^k \mathbf{AM}^k \right) \mathbf{M}^{|t-t'|}, & t \leq t', \\ \left(\left(\sum_{k=0}^{min(t,t')} (\mathbf{M}^T)^k \mathbf{AM}^k \right) \mathbf{M}^{|t-t'|} \right)^T, & t > t'. \end{cases} \tag{4.53}$$

4.5.2 Continuity Discussion

In general, a GP is defined over a continuous scale. Having a continuous scale would be a benefit compared to the discrete DGBN. For a GP to be continuous, the kernel needs to be defined for $t, t' \in \mathbb{R}$. In the previous two cases we did find a closed form of the kernel, that could be easily generalized to the feature space of $\mathbb{R}$ using the continuously defined partial sum of a geometric series. In the case for the DGBN, we have not such a formula. The two key issues are that neither the summation term nor the exponential of the matrix $\mathbf{M}$ are necessarily uniquely defined for $t \in \mathbb{R}$.

For the DGBN, we keep the time-scale discrete but we discuss a few ideas how to generalize the kernel for a continuous case. The summation is dependent on the minimum value of t and t', meaning that if the smaller of the two values is a natural number, the summation is defined. When setting up the covariance matrix for timesteps $\boldsymbol{t}$ we feed in the Cartesian product of $\boldsymbol{t}$ and $\boldsymbol{t}$ into the kernel, so if one of those time-points is not an integer value there will be at least one entry, where the minimum function of Expression (4.53) returns a non integer value. Müller and Schleicher (2005) have discussed specific fractional sums but a full mathematical consideration is not in the scope of this section. The exponent $|t-t|'$ is real-valued if t or t' are real numbers. For rational exponents, the result is defined by the n-th root

$$\mathbf{M}^{(\frac{q}{d})} = \sqrt[d]{\mathbf{M}^q}. \tag{4.54}$$

The n-th root can have none, exactly one, or multiple solutions, depending on the structure of $\mathbf{M}$. A full continuous definition of the GP would need to handle the cases where there is no exact solution or put further restrictions on the transition matrix $\mathbf{M}$.

4.5.3 Kernel Validity

Since we stay in the discrete space and directly follow the proven approach from Shachter and Kenley (1989) for constructing a valid covariance matrix, the kernel is resulting in a symmetric and positive semi-definite covariance matrix and thus a valid kernel. When generalizing to the continuous setting by solving the issues discussed in the previous section, a further kernel validation is needed.

4.5.4 Defining the Gaussian Process

We have defined the covariance function in Expression (4.53). Again, the mean function stays constant and is given by

$$m(d,t) = \mu_{X_d}. \tag{4.55}$$

The hyper parameters for the GP are the mean values and all entries of the time transition matrix $\boldsymbol{M}$.

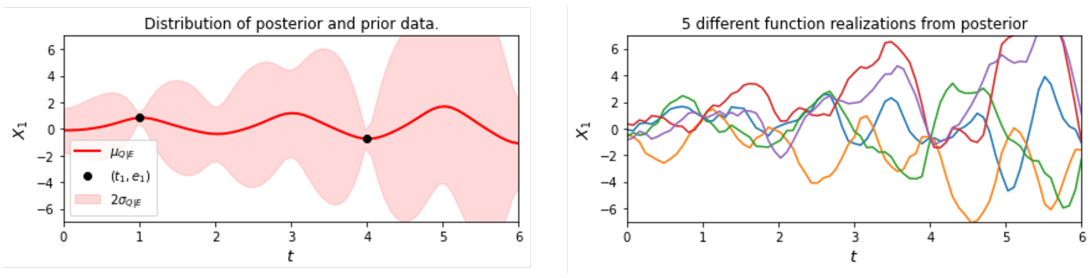


Figure 4.7: Combination of a periodic kernel with the one-dimensional Markov chain kernel from Expression (4.32)

Overall, we have now constructed three kernels for three different types of DGBNs. For two of them we have been able to generalize to a continuous time scale and for one we had to stay on a discrete one. Next, we conclude the chapter by a discussion of the results.

4.6 Intermediate Discussion

In this chapter, we have encoded one-dimensional Gaussian Markov chains, Gaussian hidden Markov models, and two-timeslice DGBNs as GPs, demonstrating the generalizing power of GPs (Contribution **4**). For the first two models, the constructed kernel function allows for a continuous treatment of the time dimension, which is a useful add-on to the classic dynamic PGM approach with a discrete time-scale allowing for varying intervals of observations. A continuous time scale avoids defining a sample rate when introducing evidence into the model. Additionally, tdvars can be queried at any real-valued point in time. The two-timeslice DGBN kernel is so far only defined on a discrete time scale but might be further generalized to a continuous scale in future research. Here, we have identified the challenges that need to be solved to allow for continuous treatment. One challenge is to handle the fractional sum and the other is to handle real valued matrix exponents in the kernel function.

For all constructed kernels, we have shown that the kernel is valid, resulting in a symmetric and positive semi-definite covariance matrix. The Markov-property of the kernel is preserved and the kernel function can be combined with other kernels to allow for different covariance structures. Fig. 4.7 contains an illustration of a combination of a periodic kernel with the one-dimensional Markov chain kernel from Expression (4.32). Introducing periodic behavior into a Gaussian Markov chain is not trivial in the PGM notation. In the new GP notation, we can simply combine kernels to combine their characteristics.

The kernels also allow for more efficient query answering. Instead of propagating evidence through the network, we can construct the evidence covariance matrices and directly calculate the posterior distribution for any chosen query time-point. In this chapter we have also impacted the theoretical research in the areas of PGMs and GPs Bringing together different perspectives and the underlying concepts can benefit both research areas. Existing methods from one area can be possibly transferred to the other research stream and enhance existing applications and vice versa. Also, further research can be better directed because scholars in different research groups can work more closely together.

Further research could focus on evaluating whether the newly defined kernels and their characteristics can enhance existing use cases of GPs for time series modeling. Additionally, it would be interesting to see if existing use cases of DGBNs can be enhanced by either having less run-time for inference, in the presence of a continuous time scale, or by combining the new kernels with existing ones.

Chapter 5

Approximate Query Answering in Complex Gaussian Mixture Models

The models we discussed in the previous chapters work only with Gaussian distributed randvars. Of course there are real-world examples that do not follow a fully Gaussian distributed setup. In such situation, hybrid PGMs, i.e. PGMs with Gaussian and discrete randvars, can be used for flexibility. Hybrid PGMs can be transformed into Gaussian mixture models (GMMs). Thus query answering in hybrid PGMs is connected to query answering in GMMs, which is focus of this chapter.

GMMs are a set of weighted multivariate Gaussian distributions, whereas each element of the set is called a component. GMMs have been used very widely for modeling complex probabilistic distributions across diverse fields including agriculture, medicine, and bioinformatics (McLachlan *et al.*, 2019). If the dimensions of the probability distribution and number of mixture components are large, the computational costs for calculating posterior distributions grow. In nonparametric approaches, the number of components is not limited before learning the model (Rasmussen, 2000) allowing the mixture to grow in its number of components dependent on the data that is fed into the model. Examples for these models are infinite GMMs (Rasmussen, 2000) and nonparametric BNs (Hanea *et al.*, 2015; Ickstadt *et al.*, 2011). These nonparametric models have been used more widely recently and several use cases from different fields have been identified (Hanea *et al.*, 2015). Existing exact query answering algorithms that work along the semantics of the query language perform calculations for all mixture components separately, which works very well for models with low dimensionality and a low number of components (Petersen, 2011). If the model complexity grows the computational costs increase rapidly.

In this chapter, we describe an approach to speed up the query answering in GMMs with focus on calculating conditional probability distributions. The approach uses the fact that the posterior weights of mixture components can be calculated with relatively little cost and combined with a modified version of a quick-select algorithm to identify the most important mixture components for the posterior distribution. The relatively costly step of calculating a posterior distribution for each mixture is only performed on the priority components, which significantly reduces the overall runtime. In situations with a high number of components and dimensions, the approximate approach is significantly faster while maintaining a low error bound.

This chapter is based on the publication:

> Mattis Hartwig, Marcel Gehrke, and Ralf Möller. Approximate Query Answering in Complex Gaussian Mixture Models. In *ICBK-19 Proceedings of the 2019 IEEE International Conference on Big Knowledge*, pages 81–86. IEEE, 2019

The remainder of this chapter has the following structure. We begin by introducing GMMs, explain how highly complex mixture models can emerge, and introduce how query answering works in GMMs. Afterwards, we introduce our approximate approach followed by a detailed complexity comparison and an evaluation. The chapter is concluded by a discussion of the intermediate results.

5.1 GMM Preliminaries

This section introduces GMMs and explains how highly complex GMMs can emerge. For further information on mixture models or Gaussian distribution, we refer to Bishop (2006).

5.1.1 Gaussian Mixture Models

In Section 2.3, we describe the characteristics of Gaussian distributions, allowing for relatively easy calculations. Unfortunately, not all real-world phenomena can be represented well enough with a pure Gaussian distribution. To keep some of the nice features of Gaussian distributions and gain flexibility in the representation, mixtures of Gaussians have been developed, which use a set of weighted multivariate Gaussian distributions. A mixture of multivariate Gaussians can approximate any density function up to an arbitrary accuracy (Alspach and Sorenson, 1972).

A mixture model over a set of randvars $\mathcal{X}$ is defined as a set of K probability distributions $P(\mathcal{X}|\boldsymbol{\theta}_k)$ and K corresponding weights $w(k)$:

$$\left(\{P(X|\boldsymbol{\theta}_k)\}_1^K, \{w(k)\}_1^K\right), \tag{5.1}$$

where $\boldsymbol{\theta}_k$ includes the parameters of the corresponding probability distribution, with $w(k) \geq 0$ and

$$\sum_{k=1}^{K} w(k) = 1. \tag{5.2}$$

Each pair of a probability distribution and a weight is called a mixture component or just component. A mixture model describes a joint probability distribution of the following form:

$$P(X) = \sum_{k=1}^{K} w(k) P(X|\boldsymbol{\theta}_k) \tag{5.3}$$

Each component can be interpreted as an expert whose weight is a measure for his relative “expertise” (Gormley and Frühwirth-Schnatter, 2018). Since the weights sum-up to one the weights basically form a discrete probability distribution over the parameters. A mixture model is called a GMM if all distributions $P(X|\boldsymbol{\theta}_k)$ are Gaussians. In a GMM, each parameter set $\boldsymbol{\theta}_k$ contains a mean $\boldsymbol{\mu}_k$ and a covariance matrix $\boldsymbol{\Sigma}_k$.

5.1.2 Motivating High-Complexity Mixtures

The complexity of GMMs is driven by the number of dimensions and the number of components. The number of dimensions is equivalent to the number of randvars in $\mathcal{X}$ (in this chapter we refer to it with dimensions because this term is often used in connection with mixture models). The number of components in a mixture model is often fixed but nonparametric models are used to make the number of components dependent on the amount of data that is fed into the model, resulting in a possibly very high number of components (Rasmussen, 2000). One example for high dimensions in models are medical diagnosis networks. The famous “Computer-based Patient Case Simulation probabilistic model” (CPCS-PM) has 422 dimensions and the “Quick Medical Reference Bayesian Network” (QMR-BN) even has 4040 dimensions (Pradhan *et al.*, 1994; Shwe *et al.*, 1991). Hybrid BNs containing discrete and Gaussian randvars can be restated into GMM notation, which makes our work applicable in these fields as well (Shachter and Kenley, 1989). Nonparametric BNs are a special type of BNs, where both the number of dimensions and the number of components can get arbitrary large depending on the data that has been used to learn the model (Ickstadt *et al.*, 2011).

In general, a high number of components, which can be interpreted as a high number of experts, allows for very specific “expert opinions” in the model. Query answering in these mixtures can be interpreted as asking all experts based on a certain evidence. Since the knowledge of the experts depends on the evidence, we should ask different experts for different sets of evidence (having a broken leg, no one would ask a dentist). An approximate algorithm that does not wait for the full answer of every expert but can anticipate which expert’s opinions are needed for the specific evidence is a contribution to make highly complex GMMs more usable.

5.1.3 Query Answering

Answering a conditional probability query $P(\boldsymbol{Q}|\boldsymbol{E} = \boldsymbol{e})$ as defined in Definition 2.2.5 requires calculating the posterior distribution for each component weighted with the posterior weight of the component:

$$P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e}) = \sum_{k=1}^{K} w(k|\boldsymbol{E}=\boldsymbol{e}) P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e}, \boldsymbol{\theta}_k). \tag{5.4}$$

The conditional distribution $P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e}, \boldsymbol{\theta}_k)$ in each component is computed using Expressions (2.11) and (2.12) from Section 2.3.

According to the Bayes rule (see Expression (2.3)), to calculate the posterior weight $w(k|\boldsymbol{E}=\boldsymbol{e})$, we calculate the likelihood given the observation multiplied with the prior and divided by the overall likelihood of observing $\boldsymbol{E}=\boldsymbol{e}$:

$$w(k|\boldsymbol{E}=\boldsymbol{e}) = \frac{w(k)P(\boldsymbol{E}=\boldsymbol{e}|k)}{P(\boldsymbol{E}=\boldsymbol{e})}. \tag{5.5}$$

The fact that not only the component posterior distributions change but also the weights of posterior mixture change can be counter-intuitive. The intuition behind is that each component has an area where it is knowledgeable. It is also called a mixture of experts (Gormley and Frühwirth-Schnatter, 2018). The evidence therefore has an impact on the probability of a component to be an "expert" for that case. Remark: In classic mixture of experts models, combining the expert opinions into one query answer is known as pooling (Zhang and Ma, 2012). Approaches can be majority votes or linear combination. The classic query answering approach in Expression (5.4) is a weighted linear combination using the posterior component weights.

As apparent in Expressions (2.11) and (2.12), the calculation of the conditional covariance matrix involves matrix inversion and matrix multiplication. Both computations are polynomial and thus are driving the complexity of the query answering algorithm (Boyd and Vandenberghe, 2018). A classic query answering approach repeats the inference steps for each mixture component. The result is that execution time grows linearly with the number of components and polynomially with the number of dimensions which causes a need for faster approximate approaches. We will revisit the runtime complexity of the classic approach as a baseline in the complexity analysis in Section 5.3.

5.2 Approximation Approach

To improve the query answering runtime, the idea of our approximation approach is to divide the calculation in Expression (5.4) into two steps. First, we calculate the posterior weights for all components based on Expression (5.5). Second, we use these posterior weights to prioritize components based on an allowed error bound and only calculate the conditional distribution based on Expressions (2.11) and (2.12) for the prioritized components. As an intuition, we can think of our first step as an approximation of the expert knowledge given some evidence. In the second step, we only ask experts and wait for their answers if they contribute enough to the final answer. In this section, we start with detailing our concrete algorithm and then show that a predefined error bound is met.

5.2.1 Detailed Algorithm

Algorithm 3 Recursive select top-L elements function

```
1: procedure SelectComponents($list, errorBound, sC$)  ▷ sC list used to store the selected components
2:     $size \leftarrow list.size$
3:     $pivot \leftarrow$ Partition($list, 0, size - 1$)  ▷ partitions the list and returns the pivot element
4:     $partSum \leftarrow$Sum($list[pivot : size]$)  ▷ sums the right part of the partitioned list
5:     if $partSum < errorBound$ then
6:         Add($sC, list[pivot : size]$)  ▷ adds the right part to the selected components
7:         $newBound \leftarrow errorBound - partSum$
8:         SelectComponents($list[0 : pivot], newBound, sC$)  ▷ recursively calls on left partition with new bound
9:     else
10:        if $pivot \neq 0$ then
11:            SelectComponents($list[pivot : size], errorBound, sC$)  ▷ recursively calls on right partition with old bound
12:        else
13:            Add($sC, list[pivot : size]$)  ▷ adds the final component and terminates
```

The algorithm is given a GMM with K components and N dimensions (number of randvars). The randvars are split into queried randvars $\boldsymbol{Q}$ and measured randvars $\boldsymbol{E}$. A given error bound ϵ specifies the desired accuracy of the approximation.

Step 1: Calculate Posterior Weights

To calculate the posterior weights, the evidence $\boldsymbol{E} = \boldsymbol{e}$ and the priors $w(k)$ are put in Expression (5.5). This calculation is performed for all components and the result is an unsorted list that contains all posterior weights. Because of the conjugate nature of the weight distribution, all the posterior weights still sum up to one, see Expression (5.2).

Step 2: Component Selection

The goal is to select a subset $\boldsymbol{\Upsilon} \subseteq \{1, \ldots, K\}$ of all mixture components such that the difference between the conditional probability distribution of the subset $\boldsymbol{\Upsilon}$ (the approximation) and the exact conditional probability distribution of the whole mixture as in Expression (5.4) is smaller than the error bound ϵ in any point, i.e.,

$$\sum_{k\in\{1,\ldots,K\}} w(k|\boldsymbol{E}=\boldsymbol{e})P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e},\boldsymbol{\theta}_k) - \sum_{k\in\boldsymbol{\Upsilon}} w(k|\boldsymbol{E}=\boldsymbol{e})P(\boldsymbol{E}=\boldsymbol{e},\boldsymbol{\theta}_k) \leq \epsilon. \tag{5.6}$$

To save runtime, the size of subset should be as small as possible. Unfortunately, a direct minimization of the size of $\boldsymbol{\Upsilon}$ would involve the computation of the conditional probability distributions of all components, which would result in the exact solution anyhow. We have developed a method that selects a subset $\boldsymbol{\Upsilon}$ that fulfills Expression (5.6) without calculating the full posterior for every component. Our algorithm selects the top-L components whose sum of posterior weights is at least $1-\epsilon$ (in Expression (5.6) we describe why the error bound holds). A naive approach would sort the whole list and would then select the top-L components until their sum reaches $1-\epsilon$ but would need on average $O(Klog(K))$ time. Our approach is similar to the Quick-Select algorithm (Hoare, 1961) and like Quick-Select also has $O(K)$ average runtime (for pseudocode of the recursive procedure, see Alg. 3). One element of the unsorted list is selected as a pivot element that is used to partition the list. Afterwards, the right (bigger) part is summed up. If this sum is smaller than the needed accuracy, the complete right part is selected and only the left part needs to be checked further. For the left part, the same partitioning is done again and the selection function is called recursively with a new accuracy decreased by the sum of the right part. If the sum of the right part is bigger, the left part can be ignored and the right part needs to be checked further. The recursive process starts again but this time with an unchanged needed accuracy because no elements have been selected yet. The quick-select algorithm has an average runtime of $O(K)$ as long as the pivot is not systematically chosen badly because then each partition step reduces the list size by a fraction which results in linear runtime (Hoare, 1961). The number of element swaps in the partitioning process is smaller or equal to the number of components in the larger partition. The same is true for the additional summing-up that is needed by our selection algorithm. Since both have the same upper bound, we stay in the complexity of $O(K)$.

Step 3: Completing the Posterior

In the last step, the posterior probability distribution of each of the L selected mixture components is calculated based on Expressions (2.11), (2.12) and (5.4). The new posterior mixture with L components represents the answer of the conditional probability query.

The next section discusses the error bound in more detail and explains why it holds even without calculating the exact posterior distributions for all components.

5.2.2 Error Bound

We need to show that our approximation fulfills the defined error bound in Expression (5.6).

Theorem 5.2.1. *The error bound*

$$\sum_{k \in \{1,\dots,K\}} w(k|\boldsymbol{E}=\boldsymbol{e})P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e},\boldsymbol{\theta}_k) - \sum_{k \in \boldsymbol{\Upsilon}} w(k|\boldsymbol{E}=\boldsymbol{e})P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e},\boldsymbol{\theta}_k) \leq \epsilon \tag{5.7}$$

described in Expression (5.6) *holds when using the approximation approach to select the top-L components in* $\boldsymbol{\Upsilon}$*.*

Proof. We can shorten Expression (5.6) by subtracting all components in the selected set $\boldsymbol{\Upsilon}$ which leaves us with components that are not selected:

$$\sum_{k \in \{1,\dots,K\} \setminus \boldsymbol{\Upsilon}} w(k|\boldsymbol{E}=\boldsymbol{e})P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e},\boldsymbol{\theta}_k) \leq \epsilon. \tag{5.8}$$

We know that the sum of all the component weights is equal to 1 (see Expression (5.2) and Expression (5.5)). The sum of the weights of the selected components is bigger than $1-\epsilon$, resulting in the sum of the ignored component weights being lower than ϵ:

$$\sum_{k \in \{1,\dots,K\} \setminus \boldsymbol{\Upsilon}} w(k|\boldsymbol{E}=\boldsymbol{e}) \leq \epsilon. \tag{5.9}$$

Because $P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e},\boldsymbol{\theta}_k)$ is a probability function, its value is always lower or equal to 1 resulting in

$$w(k|\boldsymbol{E}=\boldsymbol{e})P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e},\boldsymbol{\theta}_k) \leq w(k|\boldsymbol{E}=\boldsymbol{e}). \tag{5.10}$$

Combining Expressions (5.8) to (5.10) results in:

$$\sum_{k \in \{1,\dots,K\} \setminus \boldsymbol{\Upsilon}} w(k|\boldsymbol{E}=\boldsymbol{e})P(\boldsymbol{Q}|\boldsymbol{E}=\boldsymbol{e},\boldsymbol{\theta}_k) \leq \sum_{k \in \{1,\dots,K\} \setminus \boldsymbol{\Upsilon}} w(k|\boldsymbol{E}=\boldsymbol{e}) \leq \epsilon. \tag{5.11}$$

The inequality in Expression (5.11) shows us that Expression (5.6) and Expression (5.8) are always true if we select the components according to our algorithm, which proves the theorem. □

Thm. 5.2.1 shows that we stay in the error bound when using the approximation approach. Next, we analyse the run-time complexity of the approximation.

5.3 Complexity Analysis

We define

- N as the number of randvars (dimensions) in $\mathcal{X}$,
- C as the number of measured variables in $\boldsymbol{E}$,
- K as the number of mixture components,
- L as the number of selected components by the approximation algorithm, and
- $\frac{L}{K} = \Gamma$ as the component utilization factor ($\Gamma = 1$ means that all components need to be used in the posterior to meet the error bound)

In the classic implementation, we have to calculate K times a posterior weight and K times a posterior probability. For the calculation of the posterior weights, two multiplications of two matrices with the dimensions $C \times C$ and an inversion of a $C \times C$ matrix have to be performed (see Expressions (2.11), (2.12) and (5.5)). The runtime of both operations grows polynomial and the function that describes all necessary operations is in $\Omega(C^2)$ and $O(C^3)$ resulting in a $O(KC^3)$ complexity for calculating the posterior weights for all components (Boyd and Vandenberghe, 2018). To calculate the posterior probability distribution for each component based on Expressions (2.11) and (2.12), we need to perform a $C \times C$ matrix inversion and a matrix multiplication involving two $(N-C) \times C$ matrices (one being transposed) which results in a function in $O(K(N-C)^3)$ to describe all necessary operations. In a realistic setup, it often holds that $N \gg C$, resulting in a complexity of $O(KN^3)$.

In the approximate algorithm, the first step is identical and therefore has a complexity of $O(KC^3)$. The second step spends additional runtime to save runtime later in step 3. As described in Section 5.2 selecting the top-L components that meet our error bound can be done in $O(K)$. The function describing the last step is in $O(\Gamma K(N-C)^3)$ because only L of the K components need to be used. Comparing the two approaches shows that we have runtime savings if:

$$\begin{aligned}
& K + \Gamma K(N-C)^3 \leq K(N-C)^3 && (5.12)\\
\Leftrightarrow \quad & 1 + \Gamma(N-C)^3 \leq (N-C)^3 && (5.13)\\
\Leftrightarrow \quad & \frac{1}{(N-C)^3} + \Gamma \leq 1 && (5.14)\\
\Leftrightarrow \quad & \Gamma \leq 1 - \frac{1}{(N-C)^3} && (5.15)
\end{aligned}$$

Expression (5.15) shows that the utilization factor Γ can be nearly one for high $N-C$, which means that we even get a runtime improvement if we use most of the components and the number of dimensions is not too small. For complex mixture structures used in practice, the condition should be always true resulting in a significant speed up. In the next section, we look at the actual time savings of the approximation approach.

5.4 Experimental Evaluation

We look at two evaluations. Evaluation 1 focuses on the linear runtime of the top-L component selection and Evaluation 2 focuses on the time savings of the whole approximation algorithm compared to the exact implementation.

5.4.1 Evaluation 1: Top-L Selection

We evaluate the top-L algorithm with weight lists of different sizes. To get realistic test data, the lists are generated by a Dirichlet process, a process that is also often used to generate mixture models (Görür and Rasmussen, 2010). The evaluation is performed with different values for the α-parameter of the Dirichlet process. The α-parameter is responsible for the layout of the distribution. A high α-parameter results in a more peaky distribution, which means that a few components have high weights and others small weights. For each combination of number of components and α-parameter, we perform 200 runs and take the average runtime.

Fig. 5.1 shows the linear behavior of our selection algorithm independently of the chosen α-parameter of the Dirichlet process.

5.4.2 Evaluation 2: Time Savings

To evaluate the actual time savings compared to the exact implementation, we look at the time savings dependent on the number of mixture components and number of dimensions. Our starting mixture has an uniform weight distribution where the mean vectors and covariance matrices are randomly generated.

Table 5.1 shows the average runtime measurements for the exact and approximate algorithm using an error bound of $\epsilon = 0.01$. The measurements show that when the complexity get really high (1250 components and 1250 dimensions) that the run-time of the base algorithm explodes. Simulations for even more complex GMMs were not possible anymore. Even more important than the absolute savings are the relative improvements. Table 5.2 shows that the runtime saving factor ($\frac{\text{runtime exact}}{\text{runtime approx.}}$) increases with the number of components and the number of dimensions, which we also assumed based on the complexity analysis. Interestingly, the time savings of the approximation always outweigh the additional cost for selecting the top-L components, even in relatively low complex situations. In the high complexity case in our evaluation, the approximate algorithm performs up to 40 times than the exact implementation. That factor is high enough to make mixture models applicable in situations where they otherwise would lead to

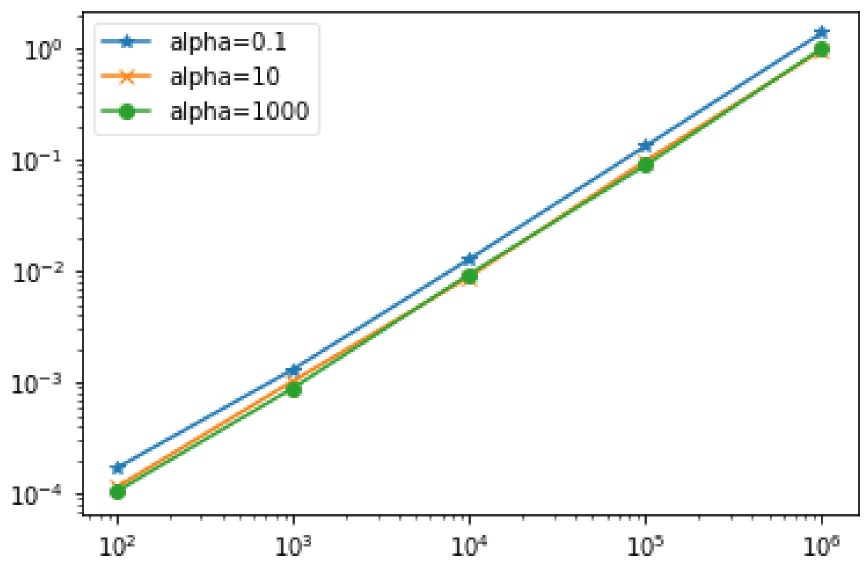


Figure 5.1: Runtime [s] (y-axis) for different list sizes [#] (x-axis)

unreasonably high computational costs. We can also observe that the error bound has not a huge effect on the time savings. With some irregularities (potentially connected to processes running in the background), we can see a tendency that a more relaxed error bound leads to more runtime savings.

Table 5.1: Average exact and approximate query answering runtime in seconds ($\epsilon = 0.01$)

Comp.	**Exact**				**Approximate**			
	Dimensions				**Dimensions**			
	10	50	250	1250	10	50	250	1250
10	0.003	0.003	0.008	0.369	0.002	0.002	0.003	0.066
50	0.014	0.013	0.035	1.778	0.007	0.007	0.010	0.120
250	0.064	0.066	0.166	8.847	0.032	0.033	0.047	0.619
1250	0.315	0.337	0.836	869.411	0.161	0.165	0.231	20.740

5.4.3 Evaluation 3: Used Components

The third evaluation investigates the number of components used in the approximate approach and the results should be connected to the results in the time savings evaluation because the run-time is driven by the number of components for which the posterior distribution needs to be calculated. Table 5.3 contains the average number of components

Table 5.2: Average time saving factor of approximate vs exact algorithm for three different error bounds

Comp.	$\epsilon = 0.01$				$\epsilon = 0.05$				$\epsilon = 0.1$			
	Dimensions				**Dimensions**				**Dimensions**			
	10	50	250	1250	10	50	250	1250	10	50	250	1250
10	1.7	1.4	2.4	5.6	2.1	1.2	1.9	6.7	1.5	1.6	2.5	7.3
50	2.1	1.9	3.4	14.9	1.9	2.0	3.3	12.8	1.9	2.0	3.5	13.9
250	2.0	2.0	3.5	14.3	1.9	2.1	3.5	19.6	1.9	2.1	3.7	18.0
1250	2.0	2.0	3.6	41.9	1.9	2.1	3.7	34.6	2.0	2.1	3.7	36.2

used for different error bounds and show that for more restrictive error bounds the number of used components gets larger. It is reasonable that the approximation also needs more components if the underlying model has more components. Similar to the time-savings, the number of dimensions also has a positive effect on the number of components but much less than the number of components. In general, the number of used components for the restrictive error bound in Table 5.3 is around 5%.

Table 5.3: Average number of used components for different error bounds

Comp.	$\epsilon = 0.1$				$\epsilon = 0.01$				$\epsilon = 0.001$			
	Dimensions				**Dimensions**				**Dimensions**			
	10	50	250	1250	10	50	250	1250	10	50	250	1250
10	1.2	2.0	1.0	1.1	1.1	2.2	2.3	1.2	2.2	2.2	2.8	2.4
50	1.8	1.7	1.9	1.6	2.8	2.6	2.4	2.8	2.5	2.5	2.5	3.3
250	5.3	4.2	5.1	3.1	7.8	7.1	7.4	9.1	9.4	10.8	11.8	12
1250	15.6	16.7	16.8	19.3	32	36.3	35.2	34	43.4	49.8	54.2	53.2

5.5 Intermediate Discussion

This chapter contributes a new approximate query answering algorithm for conditional probability distributions in highly complex GMMs. In our theoretical complexity analysis, we have proven that the additional costs of selecting the top-L mixture components are easily outweighed by the time-savings of calculating fewer conditional probability distributions. Using randomly generated test mixtures, we have shown that we can realize significant time savings in high-complexity setups. These high complexity setups will become more frequent when concepts such as nonparametric BNs or infinite mixture models are used more widely. At the same time, our approximate approach offers runtime savings that can make high complex GMMs applicable in situations where the calculation of the exact solution of a conditional probability query is currently leading to unacceptable runtime.

Chapter 6

Final Discussion

Efficiently answering queries is the backbone of many artificial intelligence applications. Especially when agents based on PGMs have to make decisions or give recommendations in a time sensitive environment, run-time efficiency is a key factor of success. We summarise the contributions of this dissertation towards efficient query answering in Gaussian PGMs and discuss the effect on artificial intelligence research in general. Afterwards, we discuss some directions for future research in general.

6.1 Contributions

This dissertation contains three paths whose common denominator is a connection to efficient query answering in Gaussian PGMs. We summarise the contributions of the three paths as follows.

Lifting GBNs We introduce a lifted representation for the joint multivariate Gaussian joint distribution and present algorithms to construct the lifted joint distribution from a parametized GBN. The algorithms presented for constructing the lifted join cover a more restricted GBN architecture, where logvar sequences are disjoint, and a more general GBN architecture allowing for overlapping logvar sequences. We develop algebraic operations for addition, multiplication, and inversion to work with the lifted joint. The algebraic operations are used in lifted query answering. The lifting of the algorithm takes two steps. The first step includes ground level handling of all evidence, whereas the second version of the algorithm allows for grouping evidence that has an identical effect on the queried randvars. We evaluate construction and query answering in a thorough complexity analysis, proving significant speed-ups. The construction of the lifted joint is completely decoupled from the logvar domain sizes, whereas the query answering still needs some operations to be performed on ground level; but with reduced coupling to the logvar domain sizes (from cubic to linear). An experimental evaluation shows that the lifted algorithms achieve significant performance improvements compared to the ground level implementations of the algorithms.

Enconding DGBNs as GPs We show how to encode three different DGBNs (one-dimensional Gaussian Markov chains, Gaussian hidden Markov models, and two-timeslice DGBNs) as GPs, demonstrating the generalizing power of GPs. We have developed the needed kernel functions and show that kernels for one-dimensional Gaussian Markov chains and Gaussian hidden Markov models allow for continuous treatment of the time dimension, which is a useful add-on to the classic DGBN approach with a discrete time-scale. All constructed kernels are valid and result in symmetric and positive semi-definite covariance matrices. The Markov property of the kernel is preserved and the kernel function can be combined with other kernels to allow for different covariance structures.

Approximate query answering in GMMs We present an algorithm for approximate query answering in highly complex GMMs meeting a predefined error bound. The approximation approach efficiently selects the most important components for the posterior probability distribution using the relative time-efficient calculation of the posterior component weights and a quick-select inspired selection algorithm. A thorough complexity analysis proves the speed-up while staying in a defined error bound. Additionally, experimental evaluations measure the runtime improvements in an implementation of the designed algorithms.

6.2 Conclusion

Looking at the described contributions it is fair to say that our work has an impact on the research of query answering in PGMs. We have developed new forms of query answering along three types of Gaussian PGMs, allowing for better understanding and implementation of use cases involving PGMs and continuous variables. More specifically, we have impact along three directions:

Our work on lifting closed the gap in existing research for continuous GBNs, which were not liftable before. Being able to connect different levels of reasoning and inference — in our case first order inference and propositional inference — is key to enable complex models in scenarios where the elements in the model also have different hierarchical levels. This allows us to connect the modelling of individuals, e.g. the patient bob, with their group or class, e.g. patients in general. With our research we have broadened the potential applications of the lifting concept to use cases involving continuous random variables with parent child relationships.

The detailed relationships between DGBNs and GPs will enable researchers from both fields to benefit each-other's ideas, methods and approaches. Additionally, inference techniques from both fields can lead to performance increases when applied in the right context.

The work on approximate query answering in GMMs has two effects. On the one hand, it might be an enabler for query answering in PGMs that have been learned using nonparametric approaches. On the other hand, it is a step to further work in hybrid PGMs.

6.3 Future Research

Throughout this dissertation, we have identified many interesting paths for further research. We focus on the ones that are most promising from our perspective.

Lifted query answering in hybrid models Currently existing lifting approaches as well as lifting approaches presented in this dissertation cover lifting for either discrete models or Gaussian models. Understanding how to apply lifting in the hybrid context would be very interesting for future research. Hybrid BNs are a good place to start for developing lifted hybrid representations. The hybrid version of the junction tree algorithm by Lauritzen and Jensen (2001) could be an interesting algorithm to lift.

Handling non-liftable queries When being exposed to non-liftable queries, the presented algorithms fall back into a full grounding of the needed covariance matrices and mean vectors. The full grounding is also necessary when only a few observed randvars break out of the needed symmetry. We have two ideas to handle these slight deviations from the needed structure. One idea is to handle parts of the operations in a lifted way and other parts in a grounded way. One would need to develop rules for splitting and combining the query operations in a way that exploits the structure that is present in the evidence and query set but can also handle the deviations without grounding the whole model. The second idea is to make additional assumptions or fake observations (i.e. evidence that is not observed but used as an approximation to achieve symmetry in the query), to ensure full liftability of the query. Here, theoretical work on error bounds would be needed to ensure that the query answer is meaningful for the application.

Time-continuity for DGBN kernels The designed DGBN kernel is so far only defined on a discrete time-scale but might be further generalized to a continuous scale in future research. Challenges that need to be solved are a fractional sum and a real matrix exponent in the kernel function. Additionally, a kernel validation on the new generalized feature space is needed.

Appendix A

Further Lifting Examples

A.1 Ground Level Covariance Construction

We use Expression (2.15) to inductively calculate the ground covariance matrix. We start with writing out the whole sum.

$$\begin{aligned}
\Sigma_{H(Patient1),H(Patient1)} &= 0 + \sigma^2_{H(Patient1)} = 1, \\
\Sigma_{H(Patient1),H(Patient2)} &= 0, \\
\Sigma_{H(Patient2),H(Patient2)} &= 0 + \sigma^2_{H(Patient2)} = 1, \\
\Sigma_{H(Patient1),T(Nurse1)} &= 0, \\
\Sigma_{H(Patient2),T(Nurse1)} &= 0, \\
\Sigma_{H(Nurse1),T(Nurse1)} &= 0 + \sigma^2_{H(Nurse1)} = 2, \\
\Sigma_{H(Patient1),T(Nurse2)} &= 0, \\
\Sigma_{H(Patient2),T(Nurse2)} &= 0, \\
\Sigma_{H(Nurse1),T(Nurse2)} &= 0, \\
\Sigma_{H(Nurse2),T(Nurse2)} &= 0 + \sigma^2_{H(Nurse2)} = 2, \\
\Sigma_{H(Patient1),T(Nurse3)} &= 0, \\
\Sigma_{H(Patient2),T(Nurse3)} &= 0, \\
\Sigma_{H(Nurse1),T(Nurse3)} &= 0, \\
\Sigma_{H(Nurse2),T(Nurse3)} &= 0, \\
\Sigma_{H(Nurse3),T(Nurse3)} &= 0 + \sigma^2_{H(Nurse3)} = 2, \\
\Sigma_{H(Patient1),W(Doctor1)} &= \Sigma_{H(Patient1),H(Patient1)}\beta_{H(Patient1),W(Doctor1)} \\
&+ \Sigma_{H(Patient1),H(Patient2)}\beta_{H(Patient2),W(Doctor1)} \\
&= 1 \cdot (-2) + 0 \cdot (-2) = -2 \\
\Sigma_{H(Patient2),W(Doctor1)} &= \Sigma_{H(Patient2),H(Patient1)}\beta_{H(Patient1),W(Doctor1)} \\
&+ \Sigma_{H(Patient2),H(Patient2)}\beta_{H(Patient2),W(Doctor1)}
\end{aligned}$$

$$
\begin{aligned}
&= 0 \cdot (-2) + 1 \cdot (-2) = -2 \\
\Sigma_{T(Nurse1),W(Doctor1)} &= \Sigma_{T(Nurse1),H(Patient1)}\beta_{H(Patient1),W(Doctor1)} \\
&+ \Sigma_{T(Nurse1),H(Patient2)}\beta_{H(Patient2),W(Doctor1)} \\
&= 0 \cdot (-2) + 0 \cdot (-2) = 0 \\
\Sigma_{T(Nurse2),W(Doctor1)} &= \Sigma_{T(Nurse2),H(Patient1)}\beta_{H(Patient1),W(Doctor1)} \\
&+ \Sigma_{T(Nurse2),H(Patient2)}\beta_{H(Patient2),W(Doctor1)} \\
&= 0 \cdot (-2) + 0 \cdot (-2) = 0 \\
\Sigma_{T(Nurse3),W(Doctor1)} &= \Sigma_{T(Nurse3),H(Patient1)}\beta_{H(Patient1),W(Doctor1)} \\
&+ \Sigma_{T(Nurse3),H(Patient2)}\beta_{H(Patient2),W(Doctor1)} \\
&= 0 \cdot (-2) + 0 \cdot (-2) = 0 \\
\Sigma_{W(Doctor1),W(Doctor1)} &= \Sigma_{W(Doctor1),H(Patient1)}\beta_{H(Patient1),W(Doctor1)} \\
&+ \Sigma_{W(Doctor1),H(Patient2)}\beta_{H(Patient2),W(Doctor1)} \\
&+ \sigma^2_{W(Doctor1)} \\
&= -2 \cdot (-2) + -2 \cdot (-2) + 3 = 11 \\
\Sigma_{H(Patient1),W(Doctor2)} &= \Sigma_{H(Patient1),H(Patient1)}\beta_{H(Patient1),W(Doctor2)} \\
&+ \Sigma_{H(Patient1),H(Patient2)}\beta_{H(Patient2),W(Doctor2)} \\
&= 1 \cdot (-2) + 0 \cdot (-2) = -2 \\
\Sigma_{H(Patient2),W(Doctor2)} &= \Sigma_{H(Patient2),H(Patient1)}\beta_{H(Patient1),W(Doctor2)} \\
&+ \Sigma_{H(Patient2),H(Patient2)}\beta_{H(Patient2),W(Doctor2)} \\
&= 0 \cdot (-2) + 1 \cdot (-2) = -2 \\
\Sigma_{T(Nurse1),W(Doctor2)} &= \Sigma_{T(Nurse1),H(Patient1)}\beta_{H(Patient1),W(Doctor2)} \\
&+ \Sigma_{T(Nurse1),H(Patient2)}\beta_{H(Patient2),W(Doctor2)} \\
&= 0 \cdot (-2) + 0 \cdot (-2) = 0 \\
\Sigma_{T(Nurse2),W(Doctor2)} &= \Sigma_{T(Nurse2),H(Patient1)}\beta_{H(Patient1),W(Doctor2)} \\
&+ \Sigma_{T(Nurse2),H(Patient2)}\beta_{H(Patient2),W(Doctor2)} \\
&= 0 \cdot (-2) + 0 \cdot (-2) = 0 \\
\Sigma_{T(Nurse3),W(Doctor2)} &= \Sigma_{T(Nurse3),H(Patient1)}\beta_{H(Patient1),W(Doctor2)} \\
&+ \Sigma_{T(Nurse3),H(Patient2)}\beta_{H(Patient2),W(Doctor2)} \\
&= 0 \cdot (-2) + 0 \cdot (-2) = 0 \\
\Sigma_{W(Doctor1),W(Doctor2)} &= \Sigma_{W(Doctor1),H(Patient1)}\beta_{H(Patient1),W(Doctor2)} \\
&+ \Sigma_{W(Doctor1),H(Patient2)}\beta_{H(Patient2),W(Doctor2)} \\
&= -2 \cdot (-2) + -2 \cdot (-2) = 8
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_{W(Doctor2),W(Doctor2)} &= \Sigma_{W(Doctor2),H(Patient1)}\beta_{H(Patient1),W(Doctor2)}\\
&+ \Sigma_{W(Doctor2),H(Patient2)}\beta_{H(Patient2),W(Doctor2)}\\
&+ \sigma^2_{W(Doctor2)}\\
&= -2\cdot(-2) + -2\cdot(-2) + 3 = 11\\
\Sigma_{H(Patient1),U(Coffee1)} &= \Sigma_{H(Patient1),T(Nurse1)}\beta_{T(Nurse1),U(Coffee1)}\\
&+ \Sigma_{H(Patient1),T(Nurse2)}\beta_{T(Nurse2),U(Coffee1)}\\
&+ \Sigma_{H(Patient1),T(Nurse3)}\beta_{T(Nurse3),U(Coffee1)}\\
&+ \Sigma_{H(Patient1),W(Doctor1)}\beta_{W(Doctor1),U(Coffee1)}\\
&+ \Sigma_{H(Patient1),W(Doctor2)}\beta_{W(Doctor2),U(Coffee1)}\\
&= 0\cdot 2 + 0\cdot 2 + 0\cdot 2 + (-2)\cdot 3 + (-2)\cdot 3 = -12\\
\Sigma_{H(Patient2),U(Coffee1)} &= \Sigma_{H(Patient2),T(Nurse1)}\beta_{T(Nurse1),U(Coffee1)}\\
&+ \Sigma_{H(Patient2),T(Nurse2)}\beta_{T(Nurse2),U(Coffee1)}\\
&+ \Sigma_{H(Patient2),T(Nurse3)}\beta_{T(Nurse3),U(Coffee1)}\\
&+ \Sigma_{H(Patient2),W(Doctor1)}\beta_{W(Doctor1),U(Coffee1)}\\
&+ \Sigma_{H(Patient2),W(Doctor2)}\beta_{W(Doctor2),U(Coffee1)}\\
&= 0\cdot 2 + 0\cdot 2 + 0\cdot 2 + (-2)\cdot 3 + (-2)\cdot 3 = -12\\
\Sigma_{T(Nurse1),U(Coffee1)} &= \Sigma_{T(Nurse1),T(Nurse1)}\beta_{T(Nurse1),U(Coffee1)}\\
&+ \Sigma_{T(Nurse1),T(Nurse2)}\beta_{T(Nurse2),U(Coffee1)}\\
&+ \Sigma_{T(Nurse1),T(Nurse3)}\beta_{T(Nurse3),U(Coffee1)}\\
&+ \Sigma_{T(Nurse1),W(Doctor1)}\beta_{W(Doctor1),U(Coffee1)}\\
&+ \Sigma_{T(Nurse1),W(Doctor2)}\beta_{W(Doctor2),U(Coffee1)}\\
&= 2\cdot 2 + 0\cdot 2 + 0\cdot 2 + 0\cdot 3 + 0\cdot 3 = 4\\
\Sigma_{T(Nurse2),U(Coffee1)} &= \Sigma_{T(Nurse2),T(Nurse1)}\beta_{T(Nurse1),U(Coffee1)}\\
&+ \Sigma_{T(Nurse2),T(Nurse2)}\beta_{T(Nurse2),U(Coffee1)}\\
&+ \Sigma_{T(Nurse2),T(Nurse3)}\beta_{T(Nurse3),U(Coffee1)}\\
&+ \Sigma_{T(Nurse2),W(Doctor1)}\beta_{W(Doctor1),U(Coffee1)}\\
&+ \Sigma_{T(Nurse2),W(Doctor2)}\beta_{W(Doctor2),U(Coffee1)}\\
&= 0\cdot 2 + 2\cdot 2 + 0\cdot 2 + 0\cdot 3 + 0\cdot 3 = 4\\
\Sigma_{T(Nurse3),U(Coffee1)} &= \Sigma_{T(Nurse3),T(Nurse1)}\beta_{T(Nurse1),U(Coffee1)}\\
&+ \Sigma_{T(Nurse3),T(Nurse2)}\beta_{T(Nurse2),U(Coffee1)}\\
&+ \Sigma_{T(Nurse3),T(Nurse3)}\beta_{T(Nurse3),U(Coffee1)}
\end{aligned}
$$

$$\begin{aligned}
&+ \Sigma_{T(Nurse3),W(Doctor1)}\beta_{W(Doctor1),U(Coffee1)} \\
&+ \Sigma_{T(Nurse3),W(Doctor2)}\beta_{W(Doctor2),U(Coffee1)} \\
&= 0\cdot 2 + 0\cdot 2 + 2\cdot 2 + 0\cdot 3 + 0\cdot 3 = 4 \\
\Sigma_{W(Doctor1),U(Coffee1)} &= \Sigma_{W(Doctor1),T(Nurse1)}\beta_{T(Nurse1),U(Coffee1)} \\
&+ \Sigma_{W(Doctor1),T(Nurse2)}\beta_{T(Nurse2),U(Coffee1)} \\
&+ \Sigma_{W(Doctor1),T(Nurse3)}\beta_{T(Nurse3),U(Coffee1)} \\
&+ \Sigma_{W(Doctor1),W(Doctor1)}\beta_{W(Doctor1),U(Coffee1)} \\
&+ \Sigma_{W(Doctor1),W(Doctor2)}\beta_{W(Doctor2),U(Coffee1)} \\
&= 0\cdot 2 + 0\cdot 2 + 0\cdot 2 + 11\cdot 3 + 8\cdot 3 = 57 \\
\Sigma_{W(Doctor2),U(Coffee1)} &= \Sigma_{W(Doctor2),T(Nurse1)}\beta_{T(Nurse1),U(Coffee1)} \\
&+ \Sigma_{W(Doctor2),T(Nurse2)}\beta_{T(Nurse2),U(Coffee1)} \\
&+ \Sigma_{W(Doctor2),T(Nurse3)}\beta_{T(Nurse3),U(Coffee1)} \\
&+ \Sigma_{W(Doctor2),W(Doctor1)}\beta_{W(Doctor1),U(Coffee1)} \\
&+ \Sigma_{W(Doctor2),W(Doctor2)}\beta_{W(Doctor2),U(Coffee1)} \\
&= 0\cdot 2 + 0\cdot 2 + 0\cdot 2 + 8\cdot 3 + 11\cdot 3 = 57 \\
\Sigma_{U(Coffee1),U(Coffee1)} &= \Sigma_{U(Coffee1),T(Nurse1)}\beta_{T(Nurse1),U(Coffee1)} \\
&+ \Sigma_{U(Coffee1),T(Nurse2)}\beta_{T(Nurse2),U(Coffee1)} \\
&+ \Sigma_{U(Coffee1),T(Nurse3)}\beta_{T(Nurse3),U(Coffee1)} \\
&+ \Sigma_{U(Coffee1),W(Doctor1)}\beta_{W(Doctor1),U(Coffee1)} \\
&+ \Sigma_{U(Coffee1),W(Doctor2)}\beta_{W(Doctor2),U(Coffee1)} \\
&+ \sigma^2_{U(Coffee1)} \\
&= 4\cdot 2 + 4\cdot 2 + 4\cdot 2 + 57\cdot 3 + 57\cdot 3 + 4 = 370 \\
\Sigma_{H(Patient1),U(Coffee2)} &= \Sigma_{H(Patient1),T(Nurse1)}\beta_{T(Nurse1),U(Coffee2)} \\
&+ \Sigma_{H(Patient1),T(Nurse2)}\beta_{T(Nurse2),U(Coffee2)} \\
&+ \Sigma_{H(Patient1),T(Nurse3)}\beta_{T(Nurse3),U(Coffee2)} \\
&+ \Sigma_{H(Patient1),W(Doctor1)}\beta_{W(Doctor1),U(Coffee2)} \\
&+ \Sigma_{H(Patient1),W(Doctor2)}\beta_{W(Doctor2),U(Coffee2)} \\
&= 0\cdot 2 + 0\cdot 2 + 0\cdot 2 + (-2)\cdot 3 + (-2)\cdot 3 = -12 \\
\Sigma_{H(Patient2),U(Coffee2)} &= \Sigma_{H(Patient2),T(Nurse1)}\beta_{T(Nurse1),U(Coffee2)} \\
&+ \Sigma_{H(Patient2),T(Nurse2)}\beta_{T(Nurse2),U(Coffee2)} \\
&+ \Sigma_{H(Patient2),T(Nurse3)}\beta_{T(Nurse3),U(Coffee2)}
\end{aligned}$$

$$
\begin{aligned}
&+ \Sigma_{H(Patient2),W(Doctor1)}\beta_{W(Doctor1),U(Coffee2)}\\
&+ \Sigma_{H(Patient2),W(Doctor2)}\beta_{W(Doctor2),U(Coffee2)}\\
&= 0\cdot 2 + 0\cdot 2 + 0\cdot 2 + (-2)\cdot 3 + (-2)\cdot 3 = -12\\
\Sigma_{T(Nurse1),U(Coffee2)} &= \Sigma_{T(Nurse1),T(Nurse1)}\beta_{T(Nurse1),U(Coffee2)}\\
&+ \Sigma_{T(Nurse1),T(Nurse2)}\beta_{T(Nurse2),U(Coffee2)}\\
&+ \Sigma_{T(Nurse1),T(Nurse3)}\beta_{T(Nurse3),U(Coffee2)}\\
&+ \Sigma_{T(Nurse1),W(Doctor1)}\beta_{W(Doctor1),U(Coffee2)}\\
&+ \Sigma_{T(Nurse1),W(Doctor2)}\beta_{W(Doctor2),U(Coffee2)}\\
&= 2\cdot 2 + 0\cdot 2 + 0\cdot 2 + 0\cdot 3 + 0\cdot 3 = 4\\
\Sigma_{T(Nurse2),U(Coffee2)} &= \Sigma_{T(Nurse2),T(Nurse1)}\beta_{T(Nurse1),U(Coffee2)}\\
&+ \Sigma_{T(Nurse2),T(Nurse2)}\beta_{T(Nurse2),U(Coffee2)}\\
&+ \Sigma_{T(Nurse2),T(Nurse3)}\beta_{T(Nurse3),U(Coffee2)}\\
&+ \Sigma_{T(Nurse2),W(Doctor1)}\beta_{W(Doctor1),U(Coffee2)}\\
&+ \Sigma_{T(Nurse2),W(Doctor2)}\beta_{W(Doctor2),U(Coffee2)}\\
&= 0\cdot 2 + 2\cdot 2 + 0\cdot 2 + 0\cdot 3 + 0\cdot 3 = 4\\
\Sigma_{T(Nurse3),U(Coffee2)} &= \Sigma_{T(Nurse3),T(Nurse1)}\beta_{T(Nurse1),U(Coffee2)}\\
&+ \Sigma_{T(Nurse3),T(Nurse2)}\beta_{T(Nurse2),U(Coffee2)}\\
&+ \Sigma_{T(Nurse3),T(Nurse3)}\beta_{T(Nurse3),U(Coffee2)}\\
&+ \Sigma_{T(Nurse3),W(Doctor1)}\beta_{W(Doctor1),U(Coffee2)}\\
&+ \Sigma_{T(Nurse3),W(Doctor2)}\beta_{W(Doctor2),U(Coffee2)}\\
&= 0\cdot 2 + 0\cdot 2 + 2\cdot 2 + 0\cdot 3 + 0\cdot 3 = 4\\
\Sigma_{W(Doctor1),U(Coffee2)} &= \Sigma_{W(Doctor1),T(Nurse1)}\beta_{T(Nurse1),U(Coffee2)}\\
&+ \Sigma_{W(Doctor1),T(Nurse2)}\beta_{T(Nurse2),U(Coffee2)}\\
&+ \Sigma_{W(Doctor1),T(Nurse3)}\beta_{T(Nurse3),U(Coffee2)}\\
&+ \Sigma_{W(Doctor1),W(Doctor1)}\beta_{W(Doctor1),U(Coffee2)}\\
&+ \Sigma_{W(Doctor1),W(Doctor2)}\beta_{W(Doctor2),U(Coffee2)}\\
&= 0\cdot 2 + 0\cdot 2 + 0\cdot 2 + 11\cdot 3 + 8\cdot 3 = 57\\
\Sigma_{W(Doctor2),U(Coffee2)} &= \Sigma_{W(Doctor2),T(Nurse1)}\beta_{T(Nurse1),U(Coffee2)}\\
&+ \Sigma_{W(Doctor2),T(Nurse2)}\beta_{T(Nurse2),U(Coffee2)}\\
&+ \Sigma_{W(Doctor2),T(Nurse3)}\beta_{T(Nurse3),U(Coffee2)}\\
&+ \Sigma_{W(Doctor2),W(Doctor1)}\beta_{W(Doctor1),U(Coffee2)}
\end{aligned}
$$

$$\begin{aligned}
&+ \Sigma_{W(Doctor2),W(Doctor2)}\beta_{W(Doctor2),U(Coffee2)} \\
&= 0\cdot 2 + 0\cdot 2 + 0\cdot 2 + 8\cdot 3 + 11\cdot 3 = 57 \\
\Sigma_{U(Coffee1),U(Coffee2)} &= \Sigma_{U(Coffee1),T(Nurse1)}\beta_{T(Nurse1),U(Coffee2)} \\
&+ \Sigma_{U(Coffee1),T(Nurse2)}\beta_{T(Nurse2),U(Coffee2)} \\
&+ \Sigma_{U(Coffee1),T(Nurse3)}\beta_{T(Nurse3),U(Coffee2)} \\
&+ \Sigma_{U(Coffee1),W(Doctor1)}\beta_{W(Doctor1),U(Coffee2)} \\
&+ \Sigma_{U(Coffee1),W(Doctor2)}\beta_{W(Doctor2),U(Coffee2)} \\
&= 4\cdot 2 + 4\cdot 2 + 4\cdot 2 + 57\cdot 3 + 57\cdot 3 + 4 = 366 \\
\Sigma_{U(Coffee2),U(Coffee2)} &= \Sigma_{U(Coffee1),T(Nurse1)}\beta_{T(Nurse1),U(Coffee2)} \\
&+ \Sigma_{U(Coffee1),T(Nurse2)}\beta_{T(Nurse2),U(Coffee2)} \\
&+ \Sigma_{U(Coffee1),T(Nurse3)}\beta_{T(Nurse3),U(Coffee2)} \\
&+ \Sigma_{U(Coffee1),W(Doctor1)}\beta_{W(Doctor1),U(Coffee2)} \\
&+ \Sigma_{U(Coffee1),W(Doctor2)}\beta_{W(Doctor2),U(Coffee2)} \\
&+ \sigma^2_{U(Coffee2)} \\
&= 4\cdot 2 + 4\cdot 2 + 4\cdot 2 + 57\cdot 3 + 57\cdot 3 + 4 = 370
\end{aligned}$$

Calculations for $U(Coffee3)$ are omitted. The resulting covariance matrix is

$$\mathbf{\Sigma} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & -2 & -2 & -12 & -12 & -12 \\
0 & 1 & 0 & 0 & 0 & -2 & -2 & -12 & -12 & -12 \\
0 & 0 & 2 & 0 & 0 & 0 & 0 & 4 & 4 & 4 \\
0 & 0 & 0 & 2 & 0 & 0 & 0 & 4 & 4 & 4 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 4 & 4 & 4 \\
-2 & -2 & 0 & 0 & 0 & 11 & 8 & 57 & 57 & 57 \\
-2 & -2 & 0 & 0 & 0 & 8 & 11 & 57 & 57 & 57 \\
-12 & -12 & 4 & 4 & 4 & 57 & 57 & 370 & 366 & 366 \\
-12 & -12 & 4 & 4 & 4 & 57 & 57 & 366 & 370 & 366 \\
-12 & -12 & 4 & 4 & 4 & 57 & 57 & 366 & 366 & 370
\end{bmatrix}.$$

A.2 Lifted Covariance Construction - Matrix Notation

We use Expression (3.38) to construct the covariance matrix. Factors infront of all-ones matrices and identity matrices can be stored lifted using $\boldsymbol{\rho}$ and $\boldsymbol{\lambda}$.

$$\mathbf{\Sigma}_{\mathbf{gr}(H(P)),\mathbf{gr}(H(P))} = \lambda_{H(P)}\mathbf{I}_{\tau_P\times\tau_P} = 1\mathbf{I}_{\tau_P\times\tau_P}$$
$$\mathbf{\Sigma}_{\mathbf{gr}(H(P)),\mathbf{gr}(T(N))} = \mathbf{\Sigma}_{\mathbf{gr}(H(P)),\mathbf{gr}(H(P))}\boldsymbol{T}_{\mathbf{gr}(H(P)),\mathbf{gr}(T(N))} = \mathbf{I}_{\tau_P\times\tau_P}\mathbf{0}_{\tau_P\times\tau_N} = \mathbf{0}_{\tau_P\times\tau_N}$$

$$\begin{aligned}
\mathbf{\Sigma}_{\mathbf{gr}(T(N)),\mathbf{gr}(T(N))} &= \mathbf{\Sigma}_{\mathbf{gr}(T(N)),\mathbf{gr}(H(P))}\boldsymbol{T}_{\mathbf{gr}(H(P)),\mathbf{gr}(T(N))} + \lambda_{T(N)}\mathbf{I}_{\tau_N\times\tau_N}\\
&= \mathbf{0}_{\tau_N\times\tau_P}\mathbf{0}_{\tau_N\times\tau_P} + \lambda_{T(N)}\mathbf{I}_{\tau_N\times\tau_N} = 0\mathbf{J}_{\tau_N\times\tau_N} + 2\mathbf{I}_{\tau_N\times\tau_N}\\
\mathbf{\Sigma}_{\mathbf{gr}(H(P)),\mathbf{gr}(W(D))} &= \mathbf{\Sigma}_{\mathbf{gr}(H(P)),\mathbf{gr}(H(P))}\boldsymbol{T}_{\mathbf{gr}(H(P)),\mathbf{gr}(W(D))}\\
&+ \mathbf{\Sigma}_{\mathbf{gr}(H(P)),\mathbf{gr}(T(N))}\boldsymbol{T}_{\mathbf{gr}(T(N)),\mathbf{gr}(W(D))}\\
&= 1\mathbf{I}_{\tau_P\times\tau_P}(-2)\mathbf{J}_{\tau_P\times\tau_D} + \mathbf{0}_{\tau_P\times\tau_N}\mathbf{0}_{\tau_N\times\tau_D} = (-2)\mathbf{J}_{\tau_P\times\tau_D}\\
\mathbf{\Sigma}_{\mathbf{gr}(T(N)),\mathbf{gr}(W(D))} &= \mathbf{\Sigma}_{\mathbf{gr}(T(N)),\mathbf{gr}(H(P))}\boldsymbol{T}_{\mathbf{gr}(H(P)),\mathbf{gr}(W(D))}\\
&+ \mathbf{\Sigma}_{\mathbf{gr}(T(N)),\mathbf{gr}(T(N))}\boldsymbol{T}_{\mathbf{gr}(T(N)),\mathbf{gr}(W(D))}\\
&= \mathbf{0}_{\tau_N\times\tau_P}(-2)\mathbf{J}_{\tau_P\times\tau_D} + 2\mathbf{I}_{\tau_N\times\tau_N}\mathbf{0}_{\tau_N\times\tau_D} = 0\mathbf{J}_{\tau_N\times\tau_D}\\
\mathbf{\Sigma}_{\mathbf{gr}(W(D)),\mathbf{gr}(W(D))} &= \mathbf{\Sigma}_{\mathbf{gr}(W(D)),\mathbf{gr}(H(P))}\boldsymbol{T}_{\mathbf{gr}(H(P)),\mathbf{gr}(W(D))}\\
&+ \mathbf{\Sigma}_{\mathbf{gr}(W(D)),\mathbf{gr}(T(N))}\boldsymbol{T}_{\mathbf{gr}(T(N)),\mathbf{gr}(W(D))}\\
&+ \lambda_{W(D)}\mathbf{I}_{\tau_D\times\tau_D}\\
&= (-2)\mathbf{J}_{\tau_D\times\tau_P}(-2)\mathbf{J}_{\tau_P\times\tau_D} + \mathbf{0}_{\tau_D\times\tau_N}\mathbf{0}_{\tau_N\times\tau_D} + \lambda_{W(D)}\mathbf{I}_{\tau_D\times\tau_D}\\
&= (-2)\cdot(-2)\cdot\tau_P\mathbf{J}_{\tau_D\times\tau_D} + \lambda_{W(D)}\mathbf{I}_{\tau_D\times\tau_D} = 8\mathbf{J}_{\tau_D\times\tau_D} + 3\mathbf{I}_{\tau_D\times\tau_D}\\
\mathbf{\Sigma}_{\mathbf{gr}(H(P)),\mathbf{gr}(U(C))} &= \mathbf{\Sigma}_{\mathbf{gr}(H(P)),\mathbf{gr}(H(P))}\boldsymbol{T}_{\mathbf{gr}(H(P)),\mathbf{gr}(U(C))}\\
&+ \mathbf{\Sigma}_{\mathbf{gr}(H(P)),\mathbf{gr}(T(N))}\boldsymbol{T}_{\mathbf{gr}(T(N)),\mathbf{gr}(U(C))}\\
&+ \mathbf{\Sigma}_{\mathbf{gr}(H(P)),\mathbf{gr}(W(D))}\boldsymbol{T}_{\mathbf{gr}(W(D)),\mathbf{gr}(U(C))}\\
&= 1\mathbf{I}_{\tau_P\times\tau_P}0\mathbf{J}_{\tau_P\times\tau_C} + 0\mathbf{J}_{\tau_P\times\tau_N}2\mathbf{J}_{\tau_N\times\tau_C} + -2\mathbf{J}_{\tau_P\times\tau_D}3\mathbf{J}_{\tau_D\times\tau_C}\\
&= -2\cdot 3\cdot\tau_D\mathbf{J}_{\tau_P\times\tau_C} = -12\mathbf{J}_{\tau_P\times\tau_C}\\
\mathbf{\Sigma}_{\mathbf{gr}(T(N)),\mathbf{gr}(U(C))} &= \mathbf{\Sigma}_{\mathbf{gr}(T(N)),\mathbf{gr}(H(P))}\boldsymbol{T}_{\mathbf{gr}(H(P)),\mathbf{gr}(U(C))}\\
&+ \mathbf{\Sigma}_{\mathbf{gr}(T(N)),\mathbf{gr}(T(N))}\boldsymbol{T}_{\mathbf{gr}(T(N)),\mathbf{gr}(U(C))}\\
&+ \mathbf{\Sigma}_{\mathbf{gr}(T(N)),\mathbf{gr}(W(D))}\boldsymbol{T}_{\mathbf{gr}(W(D)),\mathbf{gr}(U(C))}\\
&= 0\mathbf{J}_{\tau_N\times\tau_P}0\mathbf{J}_{\tau_P\times\tau_C} + 2\mathbf{I}_{\tau_N\times\tau_N}2\mathbf{J}_{\tau_N\times\tau_C} + 0\mathbf{J}_{\tau_N\times\tau_D}3\mathbf{J}_{\tau_D\times\tau_C}\\
&= 2\cdot 2\mathbf{J}_{\tau_N\times\tau_C} = 4\mathbf{J}_{\tau_N\times\tau_C}\\
\mathbf{\Sigma}_{\mathbf{gr}(W(D)),\mathbf{gr}(U(C))} &= \mathbf{\Sigma}_{\mathbf{gr}(W(D)),\mathbf{gr}(H(P))}\boldsymbol{T}_{\mathbf{gr}(H(P)),\mathbf{gr}(U(C))}\\
&+ \mathbf{\Sigma}_{\mathbf{gr}(W(D)),\mathbf{gr}(T(N))}\boldsymbol{T}_{\mathbf{gr}(T(N)),\mathbf{gr}(U(C))}\\
&+ \mathbf{\Sigma}_{\mathbf{gr}(W(D)),\mathbf{gr}(W(D))}\boldsymbol{T}_{\mathbf{gr}(W(D)),\mathbf{gr}(U(C))}\\
&= -2\mathbf{J}_{\tau_D\times\tau_P}0\mathbf{J}_{\tau_P\times\tau_C} + 0\mathbf{J}_{\tau_D\times\tau_N}2\mathbf{J}_{\tau_N\times\tau_C}\\
&+ (8\mathbf{J}_{\tau_D\times\tau_D} + 3\mathbf{I}_{\tau_D\times\tau_D})3\mathbf{J}_{\tau_D\times\tau_C}\\
&= (8\tau_D + 3)3\cdot 2\mathbf{J}_{\tau_D\times\tau_C} = 57\mathbf{J}_{\tau_D\times\tau_C}\\
\mathbf{\Sigma}_{\mathbf{gr}(U(C)),\mathbf{gr}(U(C))} &= \mathbf{\Sigma}_{\mathbf{gr}(U(C)),\mathbf{gr}(H(P))}\boldsymbol{T}_{\mathbf{gr}(H(P)),\mathbf{gr}(U(C))}\\
&+ \mathbf{\Sigma}_{\mathbf{gr}(U(C)),\mathbf{gr}(T(N))}\boldsymbol{T}_{\mathbf{gr}(T(N)),\mathbf{gr}(U(C))}\\
&+ \mathbf{\Sigma}_{\mathbf{gr}(U(C)),\mathbf{gr}(W(D))}\boldsymbol{T}_{\mathbf{gr}(W(D)),\mathbf{gr}(U(C))}
\end{aligned}$$

$$\begin{aligned}
&+ \lambda_{U(C)}\mathbf{I}_{\tau_C,\tau_C} \\
&= -12\mathbf{J}_{\tau_C\times\tau_P}0\mathbf{J}_{\tau_P\times\tau_C} + 4\mathbf{J}_{\tau_C\times\tau_N}2\mathbf{J}_{\tau_N\times\tau_C} \\
&+ (57\mathbf{J}_{\tau_C\times\tau_D})3\mathbf{J}_{\tau_D\times\tau_C} + \lambda_{U(C)}\mathbf{I}_{\tau_C,\tau_C} \\
&= (4\cdot 2\tau_N)\mathbf{J}_{\tau_C\times\tau_C} + (57\cdot 3\tau_D)\mathbf{J}_{\tau_C\times\tau_C} = 366\mathbf{J}_{\tau_C\times\tau_C} + 4\mathbf{I}_{\tau_C,\tau_C}
\end{aligned}$$

A.3 Full Lifted Joint Covariance

Here, we list all values of the lifted representation of the covariance matrix for the full example visualized in Fig. 3.1.

$$\begin{aligned}
\boldsymbol{\rho}_{1,1} &= \begin{pmatrix} \rho_{1,1}^{0_M} \\ \rho_{1,1}^{1_M} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\
\boldsymbol{\rho}_{1,2} &= \begin{pmatrix} \rho_{1,2}^{1_M 1_P} \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix} \\
\boldsymbol{\rho}_{2,2} &= \begin{pmatrix} \rho_{2,2}^{0_P} \\ \rho_{2,2}^{1_P} \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \\
\boldsymbol{\rho}_{1,3} &= \begin{pmatrix} \rho_{1,3}^{0_M 1_P} \\ \rho_{1,3}^{1_M 1_P} \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \\
\boldsymbol{\rho}_{2,3} &= \begin{pmatrix} \rho_{2,3}^{1_M 0_P} \\ \rho_{2,3}^{1_M 1_P} \end{pmatrix} = \begin{pmatrix} 10 \\ 0 \end{pmatrix} \\
\boldsymbol{\rho}_{3,3} &= \begin{pmatrix} \rho_{3,3}^{0_M 0_P} \\ \rho_{3,3}^{0_M 1_P} \\ \rho_{3,3}^{1_M 0_P} \\ \rho_{3,3}^{1_M 1_P} \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 50 \\ 0 \end{pmatrix} \\
\boldsymbol{\rho}_{1,4} &= \begin{pmatrix} \rho_{1,4}^{1_M 1_P} \end{pmatrix} = \begin{pmatrix} 8 \end{pmatrix} \\
\boldsymbol{\rho}_{2,4} &= \begin{pmatrix} \rho_{2,4}^{0_P} \\ \rho_{2,4}^{1_P} \end{pmatrix} = \begin{pmatrix} 114 \\ 0 \end{pmatrix} \\
\boldsymbol{\rho}_{3,4} &= \begin{pmatrix} \rho_{3,4}^{1_M 0_P} \\ \rho_{3,4}^{1_M 1_P} \end{pmatrix} = \begin{pmatrix} 582 \\ 16 \end{pmatrix} \\
\boldsymbol{\rho}_{4,4} &= \begin{pmatrix} \rho_{4,4}^{0_P} \\ \rho_{4,4}^{1_P} \end{pmatrix} = \begin{pmatrix} 6642 \\ 192 \end{pmatrix} \\
\boldsymbol{\rho}_{1,5} &= \begin{pmatrix} \rho_{1,5}^{1_M 1_T} \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix} \\
\boldsymbol{\rho}_{2,5} &= \begin{pmatrix} \rho_{2,5}^{1_P 1_T} \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix}
\end{aligned}$$

$$\boldsymbol{\rho}_{3,5} = \begin{pmatrix} \rho_{3,5}^{1_M 1_P 1_T} \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix}$$

$$\boldsymbol{\rho}_{4,5} = \begin{pmatrix} \rho_{4,5}^{1_P 1_T} \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix}$$

$$\boldsymbol{\rho}_{5,5} = \begin{pmatrix} \rho_{5,5}^{0_T} \\ \rho_{5,5}^{1_T} \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

$$\boldsymbol{\rho}_{1,6} = \begin{pmatrix} \rho_{1,6}^{1_M 1_D} \end{pmatrix} = \begin{pmatrix} -32 \end{pmatrix}$$

$$\boldsymbol{\rho}_{2,6} = \begin{pmatrix} \rho_{2,6}^{1_P 1_D} \end{pmatrix} = \begin{pmatrix} -228 \end{pmatrix}$$

$$\boldsymbol{\rho}_{3,6} = \begin{pmatrix} \rho_{3,6}^{1_M 1_P 1_D} \end{pmatrix} = \begin{pmatrix} -1228 \end{pmatrix}$$

$$\boldsymbol{\rho}_{4,6} = \begin{pmatrix} \rho_{4,6}^{1_P 1_D} \end{pmatrix} = \begin{pmatrix} -14054 \end{pmatrix}$$

$$\boldsymbol{\rho}_{5,6} = \begin{pmatrix} \rho_{5,6}^{1_T 1_D} \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix}$$

$$\boldsymbol{\rho}_{6,6} = \begin{pmatrix} \rho_{6,6}^{0_D} \\ \rho_{6,6}^{1_D} \end{pmatrix} = \begin{pmatrix} 3 \\ 56216 \end{pmatrix}$$

$$\boldsymbol{\rho}_{1,7} = \begin{pmatrix} \rho_{1,7}^{1_M 1_C} \end{pmatrix} = \begin{pmatrix} -192 \end{pmatrix}$$

$$\boldsymbol{\rho}_{2,7} = \begin{pmatrix} \rho_{2,7}^{1_P 1_C} \end{pmatrix} = \begin{pmatrix} -1368 \end{pmatrix}$$

$$\boldsymbol{\rho}_{3,7} = \begin{pmatrix} \rho_{3,7}^{1_M 1_P 1_C} \end{pmatrix} = \begin{pmatrix} -7368 \end{pmatrix}$$

$$\boldsymbol{\rho}_{4,7} = \begin{pmatrix} \rho_{4,7}^{1_P 1_C} \end{pmatrix} = \begin{pmatrix} -84324 \end{pmatrix}$$

$$\boldsymbol{\rho}_{5,7} = \begin{pmatrix} \rho_{5,7}^{1_T 1_C} \end{pmatrix} = \begin{pmatrix} 4 \end{pmatrix}$$

$$\boldsymbol{\rho}_{6,7} = \begin{pmatrix} \rho_{6,7}^{1_T 1_D} \end{pmatrix} = \begin{pmatrix} 337305 \end{pmatrix}$$

$$\boldsymbol{\rho}_{7,7} = \begin{pmatrix} \rho_{7,7}^{0_C} \\ \rho_{7,7}^{1_C} \end{pmatrix} = \begin{pmatrix} 4 \\ 2023854 \end{pmatrix}$$

Bibliography

Daniel L. Alspach and Harold W. Sorenson. Nonlinear Bayesian Estimation Using Gaussian Sum Approximations. *IEEE Transactions on Automatic Control*, 17(4):439–448, 1972.

Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for Vector-valued Functions: A Review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

Mauricio A. Álvarez, Wil Ward, and Cristian Guarnizo. Non-linear Process Convolutions for Multi-output Gaussian Processes. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, pages 1969–1977. PMLR, 2019.

Ankur Ankan and Abinash Panda. pgmpy: Probabilistic Graphical Models using Python. In *SCIPY 2015 Proceedings of the 14th Python in Science Conference*, volume 10, 2015.

Dennis S. Bernstein. *Matrix Mathematics: Theory, Facts, and Formulas.* Princeton university press, 2009.

Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

Stephen Boyd and Lieven Vandenberghe. *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares.* Cambridge University Press, 2018.

Tanya Braun and Ralf Möller. Lifted Junction Tree Algorithm. In *Proceedings of KI 2016: Advances in Artificial Intelligence*, pages 30–42. Springer, 2016.

Bobbi J. Broxson. The Kronecker Product. Master's thesis, University of North Florida, 2006.

Jaesik Choi, David J Hill, and Eyal Amir. Lifted Inference for Relational Continuous Models. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence.* AAAI, 2010.

Alexander M. Davie and Andrew J. Stothers. Improved Bound for Complexity of Matrix Multiplication. *Proceedings of the Royal Society of Edinburgh Section A: Mathematics*, 143(2):351–369, 2013.

Abraham de Moivre. *The Doctrine of Chances.* H. Woodfall, 1738.

M. Julia Flores, Ann E. Nicholson, Andrew Brunskill, Kevin B. Korb, and Steven Mascaro. Incorporating Expert Knowledge when Learning Bayesian Network Structure: A Medical Case Study. *Artificial Intelligence in Medicine*, 53(3):181–204, 2011.

Catherine Forbes, Merran Evans, Nicholas Hastings, and Brian Peacock. *Statistical Distributions.* John Wiley & Sons, 2011.

Brendan J. Frey and Nebojsa Jojic. A Comparison of Algorithms for Inference and Learning in Probabilistic Graphical Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1392–1416, 2005.

Roger Frigola-Alcalde. *Bayesian Time Series Learning with Gaussian Processes.* PhD thesis, University of Cambridge, 2016.

Carl F. Gauss and Charles H. Davis. *Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections: A Translation of Gauss's "Theoria Motus." With an Appendix.* Little, Brown and Company, 1857.

Marcel Gehrke, Ralf Möller, and Tanya Braun. Taming Reasoning in Temporal Probabilistic Relational Models. In *ECAI 2020 Proceedings of the 24th European Conference on Artificial Intelligence*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2592–2599. IOS Press, 2020.

Isobel C. Gormley and Sylvia Frühwirth-Schnatter. Mixtures of Experts Models. In Sylvia Frühwirth-Schnatter, Gilles Celeux, and Christian P. Robert, editors, *Handbook of Mixture Analysis.* CRC Press, 2018.

Dilan Görür and Carl E. Rasmussen. Dirichlet Process Gaussian Mixture Models: Choice of the Base Distribution. *Journal of Computer Science and Technology*, 25(4):653–664, 2010.

Anca Hanea, Oswaldo Morales Napoles, and Dan Ababei. Non-parametric Bayesian Networks: Improving Theory and Reviewing Applications. *Reliability Engineering & System Safety*, 144:265–284, 2015.

Mattis Hartwig and Ralf Möller. How to Encode Dynamic Gaussian Bayesian Networks as Gaussian Processes? In *AJCAI-20 Proceedings of the Australasian Joint Conference on Artificial Intelligence*, pages 371–382. Springer, 2020.

Mattis Hartwig and Ralf Möller. Lifted Query Answering in Gaussian Bayesian Networks. In *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 233–244. PMLR, 2020.

Mattis Hartwig and Achim Peters. Cooperation and Social Rules Emerging From the Principle of Surprise Minimization. *Frontiers in Psychology*, 11:3668, 2020.

Mattis Hartwig, Ralf Möller, and Tanya Braun. An Extended View on Lifting Gaussian Bayesian Networks. Submitted to Elsevier Artificial Intelligence Journal.

Mattis Hartwig, Marcel Gehrke, and Ralf Möller. Approximate Query Answering in Complex Gaussian Mixture Models. In *ICBK-19 Proceedings of the 2019 IEEE International Conference on Big Knowledge*, pages 81–86. IEEE, 2019.

Mattis Hartwig, Marisa Mohr, and Ralf Möller. Constructing Gaussian Processes for Probabilistic Graphical Models. In *FLAIRS-20 Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*. AAAI Press, 2020.

Mattis Hartwig, Tanya Braun, and Ralf Möller. Handling Overlaps When Lifting Gaussian Bayesian Networks. In *IJCAI-21 Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 4228–4234. IJCAI Organization, 2021.

David E. Heckerman, Eric J. Horvitz, and Bharat N. Nathwani. Toward Normative Expert Systems: Part I. The Pathfinder Project. *Methods of Information in Medicine*, 31(02):90–105, 1992.

Charles A. R. Hoare. Algorithm 65: Find. *Communications of the ACM*, 4(7):321–322, 1961.

Steven Holtzen, Todd Millstein, and Guy Van den Broeck. Generating and Sampling Orbits for Lifted Probabilistic Inference. In *UAI-20 Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 985–994. AUAI Press, 2020.

Katja Ickstadt, Björn Bornkamp, Marco Grzegorczyk, Jakob Wieczorek, M Rahuman Sheriff, Hernan E Grecco, and Eli Zamir. Nonparametric Bayesian Networks. In *Proceedings of the Valencia International Meetings on Bayesian Statistics*, volume 9. Oxford University Press, 2011.

Angelika Kimmig, Lilyana Mihalkova, and Lise Getoor. Lifted Graphical Models: A Survey. *Machine Learning*, 99(1), 2015.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques.* MIT press, 2009.

Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor Graphs and the Sum-product Algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

Steffen L. Lauritzen and Frank Jensen. Stable Local Computation with Conditional Gaussian Distributions. *Statistics and Computing*, 11(2):191–203, 2001.

Steffen L. Lauritzen and David J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, 1988.

François Le Gall. Powers of Tensors and Fast Matrix Multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pages 296–303. Association for Computing Machinery, 2014.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep Neural Networks as Gaussian Processes. In *ICLR-18 Sixth International Conference on Learning Representations*, volume 6. ICLR, 2018.

Haitao Liu, Jianfei Cai, and Yew-Soon Ong. Remarks on Multi-output Gaussian Process Regression. *Knowledge-Based Systems*, 144:102–121, 2018.

Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When Gaussian Process Meets Big Data: A Review of Scalable GPs. *IEEE transactions on neural networks and learning systems*, 31(11):4405–4423, 2020.

Anders L. Madsen, Frank Jensen, Uffe B. Kjaerulff, and Michael Lang. The Hugin Tool for Probabilistic Graphical Models. *International Journal on Artificial Intelligence Tools*, 14(03):507–543, 2005.

Alexander G. de G. Matthews, Mark Rowland, Jiri Hron, Richard E. Turner, and Zoubin Ghahramani. Gaussian Process Behaviour in Wide Deep Neural Networks. 6, 2018.

James C. Maxwell. Illustrations of the Dynamical Theory of Gases. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 20(130):21–37, 1860.

Geoffrey J. McLachlan, Sharon X. Lee, and Suren I. Rathnayake. Finite Mixture Models. *Annual Review of Statistics and its Application*, 6:355–378, 2019.

Marisa Mohr, Florian Wilhelm, Mattis Hartwig, Ralf Möller, and Karsten Keller. New Approaches in Ordinal Pattern Representations for Multivariate Time Series. In *FLAIRS-20 Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference.* AAAI Press, 2020.

Joris M. Mooij. libDAI: A Free and Open Source C++ Library for Discrete Approximate Inference in Graphical Models. *The Journal of Machine Learning Research*, 11:2169–2173, 2010.

Markus Müller and Dierk Schleicher. How to Add a Non-integer Number of Terms, and How to Produce Unusual Infinite Summations. *Journal of Computational and Applied Mathematics*, 178(1-2):347–360, 2005.

Kevin P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, University of California, Berkeley, 2002.

Jóan P. Petersen. *Mining of Ship Operation Data for Energy Conservation.* PhD thesis, Technical University of Denmark, 2011.

David Poole. First-order Probabilistic Inference. In *IJCAI-03 Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 985–991. IJCAI Organization, 2003.

Malcolm Pradhan, Gregory Provan, Blackford Middleton, and Max Henrion. Knowledge Engineering for Large Belief Networks. In *UAI-94 Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 484–490. Morgan Kaufmann Publishers Inc., AUAI Press, 1994.

Joaquin Quinonero-Candela and Carl E. Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

Carl E. Rasmussen. The Infinite Gaussian Mixture Model. In *NIPS-20 Advances in Neural Information Processing Systems*, volume 12, pages 554–560, 2000.

Carl E. Rasmussen. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

Steven Reece and Stephen Roberts. An Introduction to Gaussian Processes for the Kalman Filter Expert. In *Proceedings of the 13th International Conference on Information Fusion*, pages 1–9. IEEE, 2010.

Steven Reece and Stephen Roberts. The Near Constant Acceleration Gaussian Process Kernel for Tracking. *IEEE Signal Processing Letters*, 17(8):707–710, 2010.

Stephen Roberts, Michael Osborne, Mark Ebden, Steven Reece, Neale Gibson, and Suzanne Aigrain. Gaussian Processes for Time-series Modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110550, 2013.

Mark F. Schilling, Ann E. Watkins, and William Watkins. Is Human Height Bimodal? *The American Statistician*, 56(3):223–229, 2002.

Eric Schulz, Maarten Speekenbrink, and Andreas Krause. A Tutorial on Gaussian Process Regression: Modelling, Exploring, and Exploiting Functions. *Journal of Mathematical Psychology*, 85:1–16, 2018.

Shayle R. Searle and Harold Y. Henderson. Dispersion Matrices for Variance Components Models. *Journal of the American Statistical Association*, 74(366):465–470, 1979.

Matthias Seeger. Bayesian Model Selection for Support Vector Machines, Gaussian Processes and Other Kernel Classifiers. In *NIPS-99 Advances in Neural Information Processing Systems 12*, volume 12. MIT Press, 199.

Ross D. Shachter and C. Robert Kenley. Gaussian Influence Diagrams. *Management Science*, 35(5):527–550, 1989.

Vishal Sharma, Noman Ahmed Sheikh, Happy Mittal, Vibhav Gogate, and Parag Singla. Lifted Marginal MAP Inference. In *UAI-18 Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*, pages 917–926. AUAI Press, 2018.

Andy Shih and Stefano Ermon. Probabilistic Circuits for Variational Inference in Discrete Graphical Models. 33:4635–4646, 2020.

Michael A Shwe, Blackford Middleton, David E Heckerman, Max Henrion, Eric J Horvitz, Harold P Lehmann, and Gregory F Cooper. Probabilistic Diagnosis Using a Reformulation of the INTERNIST-1/QMR Knowledge Base. *Methods of information in Medicine*, 30(4):241–255, 1991.

Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel. Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. *Journal of Artificial Intelligence Research*, 47(1):393–439, 2013.

Neil H. Timm. *Applied Multivariate Analysis*. Springer Texts in Statistics. Springer, 2002.

Ryan D. Turner. *Gaussian Processes for State Space Models and Change Point Detection*. PhD thesis, University of Cambridge, 2012.

Guy Van den Broeck and Mathias Niepert. Lifted Probabilistic Inference for Asymmetric Graphical Models. In *AAAI-15 Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 3599–3605. AAAI Press, 2015.

Zhao Xu, Kristian Kersting, and Volker Tresp. Multi-relational Learning with Gaussian Processes. In *IJCAI-09 Proceedings of the 21th International Joint Conference on Artificial Intelligence*. IJCAI Organization, 2009.

Kai Yu, Wei Chu, Shipeng Yu, Volker Tresp, and Zhao Xu. Stochastic Relational Models for Discriminative Link Prediction. In *NIPS-06 Advances in Neural Information Processing Systems 19*, volume 19. MIT Press, 2006.

Cha Zhang and Yunqian Ma. *Ensemble Machine Learning: Methods and Applications*. Springer, 2012.

Publications

Mattis Hartwig, Ralf Möller, and Tanya Braun. An Extended View on Lifting Gaussian Bayesian Networks. Submitted to Elsevier Artificial Intelligence Journal

Mattis Hartwig, Tanya Braun, and Ralf Möller. Handling Overlaps When Lifting Gaussian Bayesian Networks. In *IJCAI-21 Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 4228–4234. IJCAI Organization, 2021

Mattis Hartwig and Achim Peters. Cooperation and Social Rules Emerging From the Principle of Surprise Minimization. *Frontiers in Psychology*, 11:3668, 2020

Mattis Hartwig and Ralf Möller. How to Encode Dynamic Gaussian Bayesian Networks as Gaussian Processes? In *AJCAI-20 Proceedings of the Australasian Joint Conference on Artificial Intelligence*, pages 371–382. Springer, 2020

Mattis Hartwig and Ralf Möller. Lifted Query Answering in Gaussian Bayesian Networks. In *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 233–244. PMLR, 2020

Marisa Mohr, Florian Wilhelm, Mattis Hartwig, Ralf Möller, and Karsten Keller. New Approaches in Ordinal Pattern Representations for Multivariate Time Series. In *FLAIRS-20 Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*. AAAI Press, 2020

Mattis Hartwig, Marisa Mohr, and Ralf Möller. Constructing Gaussian Processes for Probabilistic Graphical Models. In *FLAIRS-20 Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*. AAAI Press, 2020

Mattis Hartwig, Marcel Gehrke, and Ralf Möller. Approximate Query Answering in Complex Gaussian Mixture Models. In *ICBK-19 Proceedings of the 2019 IEEE International Conference on Big Knowledge*, pages 81–86. IEEE, 2019