An Improved Message-Passing Scheme for Approximate Inference using Cluster Graphs

by

Ruan Henry van Tonder

Thesis presented in partial fulfilment of the requirements for
the degree of Master of Engineering in the
Faculty of Electrical and Electronic Engineering at
Stellenbosch University

Supervisors:
Dr. C.E. van Daalen
Prof. J.A. du Preez

April 2022

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: April 2022

Abstract

This thesis presents a novel message-passing scheme for probabilistic inference using general cluster graphs. By relaxing the running intersection property as a constraint on the structure of cluster graphs, it is possible to create a message-passing scheme that generally produces better approximations than that of standard message passing. With this method, which will be referred to as *conditional message passing*, cluster graphs can be created that produce approximations similar to those produced by the region graph method. This includes the cluster variation method, which is known to produce better approximate marginal distributions than standard message passing on cluster graphs. An algorithm will be presented that allows for automatically constructing cluster graphs and performing conditional message passing for inference. Conditional message passing is compared to standard message passing (cluster graphs) as well as the parent-to-child method (region graphs) by measuring the Kullback-Leibler divergence between the approximate marginal distributions produced by these methods and the true marginal distributions. The methods are tested on a wide variety of joint distributions including the Ising model, which is notoriously difficult for message-passing algorithms to solve. Conditional message passing produced approximate marginals distributions which where closer to the true marginal distributions for 78.75% and 99% of discrete and continuous tests performed respectively. The tests also show that conditional message passing produces approximations on par with parent-to-child message passing while having an execution time that is up to 4 times faster. The improvement in execution time is due to the messages used in conditional message passing, specifically the messages based on the belief update variant, being less computationally expensive to compute. Conditional message passing is derived from the point of view of loop-correction while the region graph method is derived from maximizing an approximate energy functional. Therefore, conditional message passing provides a new interpretation for the cluster variation method as one of loop-correction that opens the door for future investigation into general loop-correction algorithms for cluster graphs.

Uittreksel

Hierdie tesis dra voor ’n oorspronklike inferensie algoritme vir toepassing op algemene trosgrafieke (cluster graphs). Hierdie nuwe metode vorm deel van bootskap-oordag (message-passing) algoritmes en spreek die huidige beperkings van trosgrafieke aan deur om die ‘running intersection property’ the verslap. Ons wys dat hierdie metode, genoemd *voorwaardelike boodskap-oordrag* (conditional message passing), bewerkstellig benaderde marginale verspreidings wat nader is aan die egte verspreidings in vergelyking met standaard inferensie op trosgrafieke. Ons verskaf ’n algoritme waarmee trosgrafieke vir die gebruik van voorwaardelike boodskap-oordrag outomaties bepaal kan word. Die inferensie algoritmes is toegepas op ’n verskeidenheid van digtheids funksies. Dit sluit die Ising model in, wat dikwels as ’n maatstaf vir inferenie algoritmes gebruik word. Alhoewel die resultate van die nuwe metode soortgelyk is aan die van die ‘region graph method’, wat die ‘cluster variation method’ insluit, het dit ’n korter uitvoer tyd. Die nuwe metode is hoofsaaklik ’n vorm van lus korreksie, terwyl bestaande metodes vanuit ’n optimeerings perspektief afgelei is. Voorwaardelike boodskap-oordrag verkaf daarom ’n nuwe lens waardeur toekomstige verbeteringe in bootskap-oordag algoritmes vorendag gebring kan word.

Acknowledgements

I would like to thank my study leaders, doctor Corné van Daalen and Professor Johan du Preez, for their guidance and expertise given during the past two years. I would also like to thank everyone that formed part of the ESL for creating such a fun and memorable postgraduate experience. Finally, I would like to thank my parents for the love and support they gave during this project.

Contents

Declaration i

Abstract ii

Uittreksel iii

List of Figures vii

Nomenclature ix

1. Introduction 3
- 1.1. Problem Statement . . . 4
- 1.2. Solution Overview and Contributions . . . 5

2. Inference Using Probabilistic Graphical Models 6
- 2.1. Probability Theory Review . . . 6
  - 2.1.1. The Joint Probability Distribution . . . 6
  - 2.1.2. The Marginal Probability Distribution . . . 7
  - 2.1.3. The Conditional Probability Distribution . . . 8
  - 2.1.4. Bayes' Theorem . . . 9
  - 2.1.5. Kullback-Leibler Divergence . . . 10
- 2.2. Inference as Marginalization . . . 11
- 2.3. The Factorized Probability Distribution . . . 12
- 2.4. Probabilistic Graphical Models . . . 13
  - 2.4.1. Bayesian Networks . . . 13
  - 2.4.2. Cluster Graphs . . . 14
  - 2.4.3. Message Passing on Cluster Graphs . . . 15
  - 2.4.4. The Region Graph Method . . . 20
- 2.5. Variational Inference . . . 24
  - 2.5.1. The Cluster Variation Method . . . 24
- 2.6. Summary . . . 27

3. Conditional Message Passing 28
- 3.1. The Running Intersection Property (RIP) . . . 28
- 3.2. Reducing Constraints on Graph Structure . . . 30

3.3. Conditional Messages . . . 31
3.4. The Conditioning Factor . . . 33
3.4.1. Conditioning Factor Calculation Example . . . 33
3.4.2. General Calculation of the Conditioning Factor . . . 37
3.4.3. Algorithm for Determining the Conditioning Factor . . . 40
3.5. Loopy Belief Update . . . 43
3.5.1. Standard Loopy Belief Update . . . 43
3.5.2. Conditional Loopy Belief Update . . . 44
3.6. Efficient Calculation of the Conditioning Factor . . . 45
3.7. Conditional Message Passing at Convergence . . . 46
3.8. Summary . . . 49

4. Experimental Investigation **50**
4.1. Methodology . . . 50
4.1.1. Generating Factor Scopes . . . 50
4.1.2. Generating Factor Distributions from Factor Scopes . . . 51
4.1.3. Comparing Approximate Marginal Distributions . . . 52
4.1.4. Message Scheduling . . . 54
4.1.5. Implementation . . . 54
4.2. Discrete Experiments . . . 55
4.2.1. The Categorical Distribution . . . 55
4.2.2. The Dirichlet Distribution . . . 55
4.2.3. Results . . . 56
4.3. Continuous Experiments . . . 60
4.3.1. The Gaussian Distribution . . . 61
4.3.2. The Gaussian-Wishart Distribution . . . 61
4.3.3. Negative Definite Information Matrices during Message Passing . . . 62
4.3.4. Results . . . 64
4.4. The Ising Model . . . 68
4.5. Improving Convergence . . . 71
4.6. Summary . . . 73

5. Conclusion and Future Work **74**
5.1. Conclusion . . . 74
5.2. Future Work . . . 75

Bibliography **76**

A. Ising Model Graphs **78**

List of Figures

2.1. Bayesian network example. . . . 13
2.2. An example of a cluster graph with its associated factors. . . . 15
2.3. Example of a loopy cluster graph. . . . 16
2.4. Example of a cluster graph that has a tree structure. . . . 17
2.5. An example of two region graphs that both satisfy the region graph condition. . . 22
2.6. Region graph example. . . . 23

3.1. A comparison between graphs that satisfy the RIP and the TRIP. . . . 29
3.2. MRIP, TRIP and RIP. . . . 31
3.3. Clusters connected to satisfy RIP and TRIP respectively. . . . 33
3.4. Conditioning factor calculation example (step 1). . . . 34
3.5. Conditioning factor calculation example (step 2) 35
3.6. Conditioning factor calculation example (step 3) 36

4.1. Categorical distributions with parameters sampled from a Dirichlet distribution. 56
4.2. Comparison between the approximation error of the different message-passing schemes. . . . 57
4.3. Direct comparison of methods. . . . 58
4.4. Mean log Kullback-Leibler divergence. . . . 59
4.5. Plotting convergence rates against two properties that influence convergence. . 59
4.6. Log execution time. . . . 60
4.7. Contour plots of two-dimensional multivariate-Gaussian distributions sampled from a Gaussian-Wishart distribution. . . . 62
4.8. An example of an invalid Gaussian distribution with a negative definite information matrix that resulted from the division of two Gaussian distributions. . . . 63
4.9. Comparison between the approximate marginal distributions computed by standard and conditional LBU. . . . 65
4.10. Comparison between the distance to the true mean and true covariance of the approximate marginal distributions estimated by CLBU and standard LBU. . . 66
4.11. Mean error for Gaussian tests. . . . 66
4.12. Convergence rates for both CLBU and LBU on Gaussian networks. . . . 67
4.13. Execution time comparison for LBU and CLBU on Gaussian networks. . . . 67
4.14. A 4×4 Markov network for the Ising model. . . . 68

4.15. Comparison between the message-passing schemes applied on a 7-by-7 Ising model. 70
4.16. Execution time comparison for the Ising model. 71
4.17. CLBU has better convergence when not including purely conditional messages. 72
4.18. Comparing CLBU with and without pure conditional messages. 72

A.1. Example of a cluster graph satisfying the RIP for a 4×4 Ising grid. 78
A.2. Example of a cluster graph that satisfies the TRIP for a 4×4 Ising grid. This graph has no purely conditional messages. 79
A.3. Example of a cluster graph that satisfies the TRIP for a 4×4 Ising grid. This graph does include purely conditional messages (red). 79
A.4. Example of a cluster graph that satisfies the TRIP for a 4×4 Ising grid. This example has the minimum amount of edges between the clusters. Algorithm 1 . 80
A.5. A region graph for a 4×4 Ising grid with regions from the cluster variation method and that satisfies the region graph condition. The counting numbers c are also included 81

Nomenclature

Acronyms and abbreviations

CLBP	Conditional loopy belief propagation
CLBU	Conditional loopy belief update
LBP	Loopy belief propagation
LBU	Loopy belief update
MRIP	Multiple running intersection property
PGM	Probabilistic graphical model
PC	Parent-to-child message passing
RIP	Running intersection property
sepset	Separation set
TRIP	Total running intersection property

Notation

X	Random variable
x	Scalar
$\mathcal{X} = \{X_1, X_2, \ldots\}$	Set of random variables
$\boldsymbol{x}$	A Specific state of a set of random variables
$\mathcal{I}_{\mathcal{X}}$	Set iterator such that $\mathcal{X} = \{X_i : i \in \mathcal{I}_{\mathcal{X}}\}$
$P(X)$	Probability distribution over the random variable X
$P(\mathcal{X})$	Joint probability distribution over the random variables in $\mathcal{X}$
$\mathbf{x} = [x_1, x_2, \ldots]^{\mathrm{T}}$	Vector
$\mathbf{X} = [X_1, X_2, \ldots]^{\mathrm{T}}$	Vector of random variables
$\mathcal{X}^{(k)}, \mathbf{X}^{(k)}, \mathbf{x}^{(k)}$	The superscript (k) indicates the size of a set or vector.
$[a \in \mathcal{A}]$	Iverson brackets: reduces to 1 if the enclosed logic statement evaluates as true and 0 if false.
$\forall\, i, j : X_{i,j} \in \mathcal{X}$	This statement reads: *for all indices i, j where the random variable $X_{i,j}$ is an element of the set $\mathcal{X}$*. To simplify notation "i, j :" will often be omitted. This results in $\forall\, X_{i,j} \in \mathcal{X}$, where the association with the indices i, j should be obvious from the context.

Variables, Functions and Distributions

$\phi(\mathcal{X})$	Non-negative function over the random variables in $\mathcal{X}$ known as a factor
Φ	Set of factor distributions
$P_\Phi(\mathcal{X})$	Probability distribution consisting of the factors in Φ
$\tilde{P}_\Phi(\mathcal{X})$	Unnormalized distribution such that $\tilde{P}_\Phi(\mathcal{X}) \propto P_\Phi(\mathcal{X})$
$\Psi(\mathcal{X})$	Probability distribution referred to as the *cluster* or *region* belief
$\psi(\mathcal{X})$	Probability distribution referred to as the *sepset belief*
$\mu(\mathcal{X})$	Message
$\text{Dir}(\mathcal{X})$	Dirichlet distribution
$\mathcal{N}(\mathcal{X}\|\mathbf{u}, \mathbf{K}^{-1})$	Multi-variate normal distribution with mean vector $\mathbf{u}$ and information matrix $\mathbf{K}$
$\mathcal{W}(\mathcal{X})$	Wishart distribution
$E_P[\mathcal{X}]$	Expectation of $\mathcal{X}$ with respect to $P(\mathcal{X})$
$H_P(\mathcal{X})$	Entropy of the distribution $P(\mathcal{X})$
$D(Q\|\|P)$	Kullback-Leibler divergence
$F[P, Q]$	Energy functional
I	Identity matrix
Z	Normalization constant also known as the *partition function*
$\emptyset$	The empty set $\{\}$.

Chapter 1

Introduction

Probabilistic methods allow us to reason intelligently in domains with many components that interact in complex ways. Many real-world problems include uncertainties, and probabilistic methods allow for logical reasoning with a solid theoretical underpinning even when faced with these uncertainties.

In engineering, uncertainty can arise from noisy sensors or unknown parameters. For example, an autonomous delivery robot may have to operate effectively despite receiving noisy accelerometer data and having an unknown payload. The Kalman filter is an example of a probabilistic method that can estimate the state of the system in the presence of these uncertainties [1].

Uncertainty can also arise from approximations that are necessary to make a problem tractable. In statistical physics, uncertainty arises due to computational limitations. The state or position of an atom in a system is uncertain due to the impracticality of computing it. Here probabilistic models are used to estimate the state in a computationally tractable way. The Ising model is an example of a probabilistic model that estimates the energy of a system of interacting atoms [2].

Probabilistic models can be viewed as joint probability distributions that are used to characterize the uncertainties of the system being modelled. The process of querying a model is known as *inference* and is the focus of this thesis. Specifically, focus is placed on inference that can be reduced to the problem of computing the marginal probability distributions of some joint probability distribution, as will be shown in Section 2.2. Unfortunately, computing the exact marginal distribution of a joint probability distribution is intractable for most cases of practical interest. A distribution may be impractical to use as a model due to the large number of parameters required to represent the distribution as well as the number of calculations required to perform operations on the distribution, such as marginalization. For example, the size of a discrete distribution grows exponentially with the number of random variables, making it impractical to represent, let alone perform operations on. In the case of continuous distributions, the required integrals may not have analytical solutions and can be too complex to estimate numerically [3]. For these high dimensional distributions, we require an inference algorithm that is efficient and that can provide approximations that are both tractable and accurate enough for practical use. One family of algorithms that strives to solve this problem is known as *message passing*. Message-passing algorithms exploit the conditional independencies between random variables to compute marginal distributions without ever having to represent the full joint probability distribution. The marginal distributions produced by message passing are exact if the conditional

independence relationships between the random variables can be represented by a graph that has a tree structure (such a distribution is known as being decomposable [4, p 108]). Unfortunately, this is usually not the case. One form of approximation that is investigated in this thesis, is to apply message-passing algorithms on graphs that contain loops. These general graphs are also known as *loopy graphs*.

1.1. Problem Statement

One method of performing inference on high dimensional probability distributions is to use a graph or network-like structure to represent the distribution in such a way as to allow for inference to be performed efficiently. This approach forms part of inference in the field of *probabilistic graphical models* (PGM), which is introduced in Section 2.4.

The specific PGM used in this thesis is known as a *cluster graph*, and message-passing algorithms, applied to cluster graphs that represent a joint distribution, will be used to compute the desired marginal distributions. Cluster graphs are discussed in Subsection 2.4.2, but a brief background is included here to provide the necessary context for understanding the problem addressed by this thesis.

Each vertex in a cluster graph has a set or *cluster* of random variables associated with it and can be viewed as representing a subset of random variables from the full joint probability distribution. Clusters that share random variables can be connected to one another via edges. During message passing, the clusters communicate information about the random variables that they share by passing messages to neighbouring clusters. Each cluster is trying to determine the marginal distribution of its own random variables by incorporating information from other clusters. Note that the term *information* is used in an informal sense to provide an intuitive explanation of certain concepts and does not refer to the term from an information theory perspective.

Message passing on cluster graphs allow for efficient computation of the exact marginal distribution when the graph has a tree structure [2]. However, when a cluster graph contains loops, message passing will not produce the exact marginal distributions as it does not take into account the information feedback that occurs when information passed out by a cluster returns via a loop. In other words, a cluster cannot distinguish between incoming information that is novel versus information that has already been incorporated. As a result, old information is treated as if it is novel, commonly leading to marginal distributions that are overconfident [1]. If the loopy graph has weak feedback loops, message passing can produce surprisingly good approximate marginal distributions, allowing for the implicit use of joint distributions for which the exact marginal distributions are intractable to compute. Unfortunately, some joint probability distributions have cluster graphs with particularly severe feedback loops. Within these loops, every message communicates information about the same random variables, leading to extremely

[1] Overconfidence is a term used to describe probability distributions that underestimate the variability of a outcome. In other words it assigns a too high probability to certain outcomes.

overconfident marginal distributions. Currently, the only method for overcoming this problem in cluster graphs is to omit the variable from certain messages [2]. This omission limits the amount of feedback but also limits the information clusters may exchange.

In this thesis, inference performed via message passing on cluster graphs is improved by creating a novel message-passing scheme that takes into account the feedback that occurs within loops where the same random variables are present throughout the loop. This should both reduce overconfidence and allow for more informative messages, leading to an overall improvement in the approximate marginal distributions produced by loopy message passing on cluster graphs.

1.2. Solution Overview and Contributions

This thesis presents a novel message-passing algorithm that produces improved approximations compared to previous message-passing algorithms applied on general cluster graphs. The new method achieves this improvement by allowing for a larger space of valid cluster graphs than was previously possible. With the new message-passing scheme, which will be referred to as *conditional message passing*, graphs can be created such that all cluster beliefs agree not only on univariate marginal distributions but on joint marginal distributions as well. Empirical evidence is provided that the new message-passing scheme generally produces better approximations than standard message-passing schemes.

The new conditional message-passing scheme is compared to the parent-to-child algorithm applied to region graphs. The region graph method is more general than cluster graphs and includes cluster graphs as a special case [5]. This thesis empirically shows that the new message-passing scheme produces approximations similar to that of the parent-to-child algorithm while having a faster execution time. This is because conditional message passing can be efficiently implemented as a belief update algorithm. Conditional message passing provides a different interpretation than that of the message-passing schemes of the region graph method, since conditional message passing is developed as a loop-correction method, while the region graph method is derived by optimizing an approximate energy functional.

Chapter 2

Inference Using Probabilistic Graphical Models

This chapter investigates existing methods for performing probabilistic inference using Probabilistic Graphical Models. Before introducing these methods, we review the relevant probability theory concepts that are required for performing inference. Following this, a clear definition of inference will be provided as used in this thesis.

2.1. Probability Theory Review

This section provides a brief review of some probability theory concepts used in this thesis and introduces some notation. The reader is expected to be familiar with certain core concepts from probability theory and set theory such as random variables, expectation and set operations. The book *Probabilistic Graphical Models* [2] includes an extensive review of all the necessary probability theory. Alternatively, the book *Probability, Random Variables and Random Signal Principles* [6] provides an in-depth introduction into probability theory.

2.1.1. The Joint Probability Distribution

Joint probability distributions describe the likelihood of two or more random variables jointly being in a certain state [1] [6]. They capture the dependencies and interactions between different random variables. It is these interactions combined with uncertainty that make probabilistic methods so powerful since it allows for soft logic. In other words, we can model systems that follow logical rules even though their outcomes are uncertain. For example, the human body is subject to 'logical rules' that result in men being taller than women, but not all men are taller than all women, which we can model using a joint probability distribution modelling the correlation between sex and height.

[1] *State* refers to a specific value or permutation of values a random variable or set of random variables can have. For example, $\{X_1 = 0, X_2 = 0\}$ is one possible state that the set $\{X_1, X_2\}$ can be in and $\{X_1 = 1, X_2 = 0\}$ is another.

We will now construct an example joint probability distribution that will be used in Subsection 2.1.3 to illustrate the interactions between random variables. Consider two binary random variables, R and M. Let R model rain where $R(\text{clear sky}) = 0$ and $R(\text{rain}) = 1$. Similarly, let M be the mood of our friend Bob where $M(\text{bad mood}) = 0$ and $M(\text{good mood}) = 1$. The discrete joint probability distribution $P(R, M)$ is shown in Table 2.1. The random variables described by a joint distribution, R and M in the example, is referred to as the *scope* of the distribution.

R	M	$P(R, M)$
0	0	0.28
0	1	0.52
1	0	0.04
1	1	0.16

Table 2.1: An example of a joint probability distribution.

Table 2.1 indicates that we are most likely to see Bob in a good mood on a sunny day.

2.1.2. The Marginal Probability Distribution

We are not always interested in the full joint probability distribution. Consider again the example distribution in Table 2.1. We might want to know if our friend Bob is generally in a good mood, irrespective of the weather. For our discrete example we can use the sum rule to calculate the *marginal* distribution $P(M)$

$$P(M) = \sum_{R} P(R, M), \tag{2.1}$$

where we sum over all possible states of the random variable we want to marginalize out [6]. When marginalizing continuous probability distributions, instead of a summation one integrates the distribution over the range of possible continuous states of the random variable. The probability that Bob is in a good mood is then

$$\begin{aligned} P(M = 1) &= P(R = 0, M = 1) + P(R = 1, M = 1), \\ &= 0.52 + 0.16, \\ &= 0.68. \end{aligned}$$

We can marginalize out any number of random variables. For example, consider a distribution with N random variables. We can calculate the marginal distribution of any K random variables, where $K < N$, as

$$P(X_1, \cdots, X_K) = \sum_{X_{K+1}} \cdots \sum_{X_N} P(X_1, \cdots, X_N). \tag{2.2}$$

To simplify notation, sets of random variables will be used when working with multiple random variables. Using the sets $\mathcal{X} = \{X_1 \cdots X_N\}$ and $\mathcal{Y} = \{X_1 \cdots X_K\}$ we can rewrite the above equation as

$$P(\mathcal{Y}) = \sum_{\mathcal{X} \backslash \mathcal{Y}} P(\mathcal{X}), \tag{2.3}$$

where $\mathcal{X} \backslash \mathcal{Y}$ is used to indicate all the elements that are within set $\mathcal{X}$ but not in set $\mathcal{Y}$.

2.1.3. The Conditional Probability Distribution

A conditional probability distribution, is a distribution that depends on the state of some other random variable or variables. $P(A|B)$ is a conditional probability distribution where the distribution over A changes depending on the state of B. It is said to be conditioned on B. For two mutually exclusive sets of random variables $\mathcal{X}$ and $\mathcal{Y}$, $\mathcal{Y} \cap \mathcal{X} = \emptyset$, a conditional distribution is defined as

$$P(\mathcal{Y}|\mathcal{X}) \triangleq \frac{P(\mathcal{Y}, \mathcal{X})}{P(\mathcal{X})}. \tag{2.4}$$

We can use this definition to calculate $P(M = 1|R = 1)$ for the example distribution in Table 2.1. That is, we want to know the probability of Bob being in a good mood *given* that we know it is raining, which is equal to

$$\begin{aligned} P(M = 1|R = 1) &= \frac{P(M = 1, R = 1)}{P(R = 1)} \\ &= \frac{0.16}{0.04 + 0.16} \\ &= 0.8. \end{aligned} \tag{2.5}$$

From this result we can see that Bob is more likely to be in a good mood on rainy days because the probability of being in a good mood increased from the marginal probability calculated in the previous subsection. Rainy days are less common according to the joint distribution. By conditioning the distribution on $R = 1$ we do not take sunny days into consideration, resulting in a different distribution over M. This illustrates how random variables interact within a joint probability distribution. Changes to R will affect M and vice versa.

2.1.4. Bayes' Theorem

In the previous subsection on conditional probabilities, we saw that conditioning on a random variable changes the distribution over the remaining variables. Bayes' theorem uses this principle to update a probability distribution such that it incorporates new information. Bayes' theorem is given as

$$P(\mathcal{X}|\mathcal{Y}) = \frac{P(\mathcal{Y}|\mathcal{X})P(\mathcal{X})}{P(\mathcal{Y})}, \tag{2.6}$$

where $P(\mathcal{X})$ is known as the *prior* distribution and $P(\mathcal{X}|\mathcal{Y})$ is known as the *posterior* distribution.

Consider again the example distribution $P(R, M)$ in Table 2.1. Currently, we are using yearly rainfall figures and not taking the seasons into account. If we know what season it is, we should have a more accurate rainfall model for the current time of the year, and therefore a more accurate model of Bob's mood. We can use the random variable W to model whether it is winter or not, where W(not winter)$= 0$ and W(winter)$= 1$.

It might seem odd to model the seasons using a random variable, after all they follow a consistent pattern. However, it does not seem so strange if you view probabilities as the *subjective belief* that an individual has on the state of a variable. That is, it makes sense for someone to say they are 90% sure summer starts next month. This interpretation is known as the *subjective interpretation* and is useful when working with Bayes' theorem. Due to the subjective interpretation, the starting probability distribution $P(\mathcal{X})$ is sometimes referred to as the prior *belief*, while the resultant distribution $P(X|Y)$ is referred to as the posterior *belief*. This interpretation of uncertainty will be useful when discussing message-passing algorithms.

We can use Bayes' theorem to update our belief about R and M prior to the incorporation of the current season, to obtain a more informed model as follows:

$$P(R, M|W) = \frac{P(W|R, M)P(R, M)}{P(W)}. \tag{2.7}$$

Let us assume that it is currently winter. The distribution $P(W = 1|R, M)$ describes how likely the observation $W = 1$ is for different values of R and M, and is therefore known as the *likelihood function*. Let us assume that $P(W = 1|R = 0, M = 0) = P(W = 1|R = 0, M = 1) = 0.125$, $P(W = 1|R = 1, M = 0) = P(W = 1|R = 1, M = 1) = 0.75$ and $P(W = 1) = 0.25$. Our likelihood function states that it is six times more likely to be winter if it rains. Using these

values we can compute a posterior probability distribution $P(R, M|W = 1)$ as follows:

$$P(R = 0, M = 0|W = 1) = \frac{P(W = 1|R = 0, M = 0)P(R = 0, M = 0)}{P(W = 1)}$$
$$= \frac{0.125 * 0.28}{0.25} = 0.14$$
$$P(R = 0, M = 1|W = 1) = \frac{0.125 * 0.52}{0.25} = 0.26$$
$$P(R = 1, M = 0|W = 1) = \frac{0.75 * 0.04}{0.25} = 0.12$$
$$P(R = 1, M = 1|W = 1) = \frac{0.75 * 0.16}{0.25} = 0.48$$

R	M	$P(R, M\|W = 1)$
0	0	0.14
0	1	0.26
1	0	0.12
1	1	0.48

Table 2.2: An example of a posterior joint probability distribution.

From the posterior distribution we can see that, in the winter, we are most likely to see Bob in a good mood on a rainy day, as opposed to the prior distribution where we were most likely to encounter Bob in a good mood on a sunny day. This illustrates how evidence can create more informed models through the interactions the random variables have with the evidence and with one another.

2.1.5. Kullback-Leibler Divergence

The Kullback-Leibler divergence [7], also known as relative entropy, measures how dissimilar two distributions are to one another. It is defined as

$$D(\ P(\mathcal{X})\ ||\ Q(\mathcal{X})\ ) = E_P\left[\log\left(\frac{P(\mathcal{X})}{Q(\mathcal{X})}\right)\right], \tag{2.8}$$

where E_P is the expected value with respect to $P(\mathcal{X})$.

One useful interpretation can be explained by the following example. Let the set of random variables $\mathcal{X}$ model a piece of English text. Now let the probability distribution $P(\mathcal{X})$ be the true probability distribution of written English and $Q(\mathcal{X})$ be some approximate model. Perhaps $Q(\mathcal{X})$ considers each word in a sentence as being independent of every other word. The Kullback-

Leibler (KL) divergence describes how much more information, on average, would be required to store the piece of text if we compress it according to Q instead of P [3].

The base of the logarithm in Equation 2.8 determines the unit of information. Base two is used in applications such as telecommunication where the unit of information is binary digits (bits). The natural logarithm is used in applications such as statistical physics where the unit is then known as natural units (nats) [8].

The KL divergence increases the more dissimilar two distributions are. It is always positive since it will always, on average, require more information to encode something when not using the true probability distribution. It will equal zero if and only if $Q = P$ (see [3, Section 1.6.1] for a proof). Also note that the KL-divergence is asymmetric in that $D(P(\mathcal{X}) \mid\mid Q(\mathcal{X}))$ is not the same as $D(Q(\mathcal{X}) \mid\mid P(\mathcal{X}))$ and will generally return a different quantity.

2.2. Inference as Marginalization

Generally when working with probabilistic problems, we have some joint probability distribution that forms a model that we want to query given some known information. This query process is also referred to as *inference*, and the known information is referred to as *evidence*. To incorporate the evidence into the model probability distribution, it is conditioned on some subset of its *scope*, where scope refers to the set of random variables over which a probability distribution varies. If we define $\mathcal{X}$ and $\mathcal{E}$ to be sets of random variables, where $\mathcal{E}$ is the evidence, then from Equation 2.4 we have

$$P(\mathcal{X}|\mathcal{E}) = \frac{P(\mathcal{X}, \mathcal{E})}{P(\mathcal{E})}. \tag{2.9}$$

We are generally not interested in the distribution over all the variables after conditioning on the evidence, but rather some subset of variables $\mathcal{Y} \subset \mathcal{X}$. A realization of the random variables in $\mathcal{E}$ are denoted with $\mathbf{e}$. Therefore, $\mathbf{e}$ is the observed evidence that we want to condition on. From this, we can calculate the marginal joint probability distribution over $\mathcal{Y}$ given some evidence as

$$P(\mathcal{Y}|\mathcal{E} = \mathbf{e}) = \frac{1}{Z} \sum_{\mathcal{X} \setminus \mathcal{Y}} P(\mathcal{X}, \mathcal{E} = \mathbf{e}), \tag{2.10}$$

where $P(\mathcal{E} = \mathbf{e})$ is replaced with a normalization constant Z also known as the *partition function*. Section 2.4.3 provides an efficient method of calculating this constant (see Equation 2.24).

This thesis focuses on the problem of calculating the marginal distribution after evidence has been incorporated. The process of calculating the marginal distribution is the same for a distribution that is conditioned on evidence versus one that is not. For example, both the prior distributions $P(R, M)$ in Table 2.1 and the posterior distribution $P(R, M|W = 1)$ in Table 2.2 are distributions over R and M, for which the calculation of $P(M)$ and $P(M|W = 1)$ will be identical. For this reason distributions that have been conditioned on evidence will not be

considered explicitly. Instead, focus is placed on the problem of computing

$$P(\mathcal{Y}) = \sum_{\mathcal{X} \setminus \mathcal{Y}} P(\mathcal{X}). \tag{2.11}$$

2.3. The Factorized Probability Distribution

We are faced with the problem of computing the marginal distribution $P(\mathcal{Y})$ when the set $\mathcal{X}$ contains many random variables, leading to a large and complex joint probability distribution $P(\mathcal{X})$. One possible solution is to use the various independencies between the random variables to rewrite the joint distribution as a product of smaller distributions also known as *factors*. A factor $\phi(\mathcal{X})$ is a positive, finite distribution over a set of random variables $\mathcal{X}$. A factor is not required to be normalized and is therefore not necessarily a probability distribution.

For example, consider the joint distribution $P(A, B, C)$, where each random variable has 10 possible states. Even this simple distribution already has 1000 states, requiring 1000 table entries to represent. Let us say that A is independent of C given B. In other words, A and C do not influence each other directly, but do so indirectly through B. This conditional independence allows us to factorize the distribution as

$$P(A, B, C) = \frac{1}{Z}\phi(A, B)\phi(B, C). \tag{2.12}$$

Each factor is a function with 100 possible states, bringing the combined number of table entries needed to represent $P(A, B, C)$ down from 1000 to 200. In Subsection 2.4.3, we will see that the factorized form is also computationally more efficient to operate on.

A factorized joint probability distribution is denoted as $P_\Phi(\mathcal{X})$, where the subscript denotes the set of factors of the distribution. Together, a set of factors Φ construct a joint probability distribution as follows:

$$\Phi = \{\phi_i(\mathcal{X}_i) : i \in \mathcal{I}_\Phi\}, \tag{2.13}$$

$$P_\Phi(\mathcal{X}) = \frac{1}{Z}\prod_{i \in \mathcal{I}_\Phi} \phi_i(\mathcal{X}_i), \tag{2.14}$$

where $\mathcal{I}$ denotes a set iterator and $\mathcal{X} = \bigcup_{i \in \mathcal{I}_\Phi} \mathcal{X}_i$. In the following sections inference methods are discussed that efficiently compute marginal distributions by using the factorized form of the joint distribution.

2.4. Probabilistic Graphical Models

Probabilistic graphical models (PGMs) can be used to create probabilistic models. They are graphical representations of probability distributions that clearly and intuitively display the relationships between the different random variables. Pearl states that "Graphs, networks, and diagrams can be viewed as inference engines devised for efficiently representing and manipulating relevance relationships" [4, p 14]. PGMs do not only help describe complex systems but also provide a framework for performing inference on such systems. They allow for inference on arbitrary models without having to rederive an inference algorithm for each model. One such method can be described as messages being passed between vertices along edges of a graph. Belief propagation is an example of such a message-passing algorithm [4]. Other inference methods, such as sampling methods, also make use of graphical structures.

PGMs can be divided into two main categories. The one is used to create a probabilistic *model* and includes Bayesian networks and Markov networks. The other, such as factor graphs, cluster graphs or region graphs, is used for *inference* on a probabilistic model. PGMs from these two categories are usually used as part of a two-step process for solving probabilistic problems. First the problem is modelled using a Bayesian or Markov network, which describes a factorized joint distribution. After the model has been created, inference is performed on the model using a graph from the second category, such as a factor graph. The graph used for inference can be obtained from the graph describing the model by either converting from the one type of graph to the other, for example converting a Markov network into a cluster graph, or it can be constructed directly from the factorized probability distribution described by the model.

2.4.1. Bayesian Networks

Bayesian networks are PGMs that can be used to create probabilistic models. They are directed acyclic graphs where the vertices represent random variables and the edges convey dependencies between the random variables [9]. Figure 2.1 shows an example. A product of conditional probability distributions can be used to create a joint distribution that has the same dependencies described by a Bayesian network.

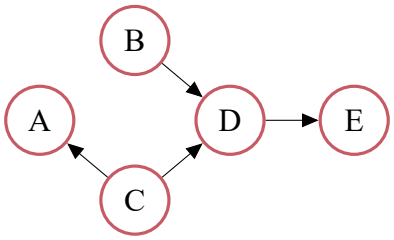


Figure 2.1: Bayesian network example.

For example, the random variable D in Figure 2.1 is directly dependent on B and C as indicated by the edges pointing toward vertex D. Variables B and C are said to be parents of D. This relationship between the random variables can be represented by the joint distribution $P(B, C, D)$, since the joint distributions capture the interaction between B and D as well as C and D. However, this also implies a direct dependence between B and C, which is not present in the network. We therefore use a conditional probability distribution $P(D|B, C) = \frac{P(B,C,D)}{P(B,C)}$, in which B and C are conditionally dependant given D, not directly dependant. Repeating this process for each variable in the example graph produces the joint probability distribution

$$P(A, B, C, D, E) = P(A|C)P(B)P(C)P(D|B, C)P(E|D). \tag{2.15}$$

A Bayesian network can be viewed as describing a factorized joint probability distribution. Consider a general Bayesian network consisting of random variables in the set $\mathcal{X} = \{X_i : i \in \mathcal{I}_\mathcal{X}\}$. The function 'parents$(X)$' returns the set of random variables that are parents of X. From this we can write the factors of the general factorized probability distribution $P_\Phi(\mathcal{X})$ described by a Bayesian network as

$$\phi_i(\mathcal{X}_i) = P(X_i|\text{parents}(X_i)), \qquad \forall\, i \in \mathcal{I}_\mathcal{X}. \tag{2.16}$$

These factors form a joint probability distribution as described by Equation 2.14. Note that the normalization constant Z is always equal to one for joint probability distributions described by Bayesian networks.

2.4.2. Cluster Graphs

Cluster graphs are used for performing inference on a joint probability distribution through a method known as message passing. They are undirected graphs where the vertices and edges are associated with sets of random variables [2]. They can be viewed as an alternative representation of a probability distribution, or a data structure that allow for efficient calculation of marginal distributions.

Each vertex in a cluster graph, labelled $C_1 \dots C_N$, is associated with a set of random variables labelled $\mathcal{C}_1 \dots \mathcal{C}_N$. The vertices and their associated sets are referred to as *clusters*. The clusters are connected via edges, where $S_{a,b}$ is used to denote an edge between clusters C_a and C_b, and each edge is associated with a set of random variables known as a separator set or sepset, and denoted with $\mathcal{S}_{a,b}$. A sepset is equal to or a subset of the intersection between the two clusters it connects (see Figure 2.2)

$$\mathcal{S}_{a,b} \subseteq \mathcal{C}_a \cap \mathcal{C}_b. \tag{2.17}$$

The clusters in a cluster graph can be determined from the factor scopes of the particular joint distribution on which inference needs to be performed. For example, consider the joint $P_\Phi(A, B, C)$ with factors $\Phi = \{\phi_1(A, B), \phi_2(B, C, D), \phi_3(A, C, D)\}$. A cluster for each factor can be created,

where the cluster scopes are $\mathcal{C}_1 = \{A, B\}, \mathcal{C}_2 = \{B, C, D\}$ and $\mathcal{C}_3 = \{A, C, D\}$. The clusters that share random variables can then be connected via sepsets as shown in Figure 2.2. When a graph contains loops, such as in Figure 2.2, it is known as a loopy cluster graph. Loops create complications when it comes to inference via message passing, which will be discussed in Subsection 2.4.3.

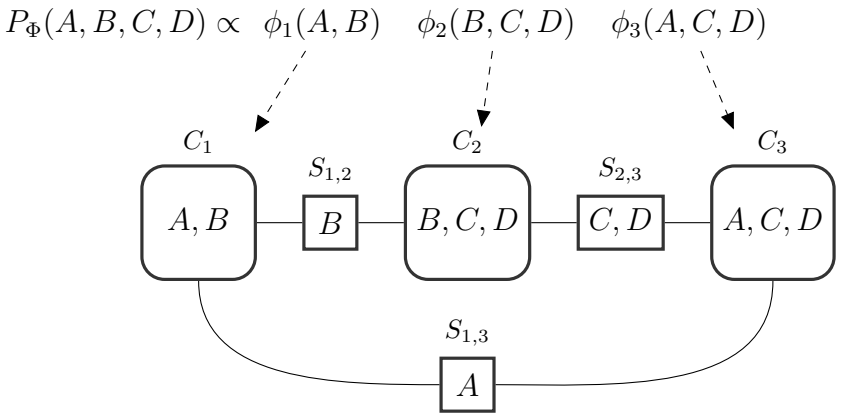


Figure 2.2: An example of a cluster graph with its associated factors.

In general, each factor in Φ is assigned to a cluster. A factor can be assigned to a cluster if its scope is equal to or a subset of that cluster $\mathcal{X}_i \subseteq \mathcal{C}_a$. Even though clusters may be assigned more than one factor, we can assume without loss of generality that each factor in Φ has a single corresponding cluster with equal scope

$$\mathcal{C}_i = \mathcal{X}_i, \qquad \forall i \in \mathcal{I}_\Phi. \tag{2.18}$$

This assumption simplifies notation and can always be satisfied by combining factors in Φ such that no factor has a scope that is equal to or a subset of another factor. For example, the set of factors $\Phi = \{\phi_1(A, B, C), \phi_2(A, B, D), \phi_3(A, D)\}$ can be simplified by creating a new factor $\phi_4(A, B, D) = \phi_2(A, B, D)\phi_3(A, D)$. The set of factors is then equal to $\Phi = \{\phi_1(A, B, C), \phi_4(A, B, D)\}$, which will result in a cluster graph where each cluster has one corresponding factor.

2.4.3. Message Passing on Cluster Graphs

One possible method for performing inference is through a process whereby clusters propagate their beliefs to connected clusters via messages. A message from cluster C_a to C_b is a distribution with the same scope as the sepset that connects the two clusters and is denoted as $\mu_{a,b}(\mathcal{S}_{a,b})$. Belief propagation, also known as the Shafer-Shenoy algorithm [10], defines a message from C_a to C_b as the product of the cluster factor ϕ_a with all incoming messages, excluding the message originating from C_b, marginalizing out all variables not in the sepset $\mathcal{S}_{a,b}$. For example, the

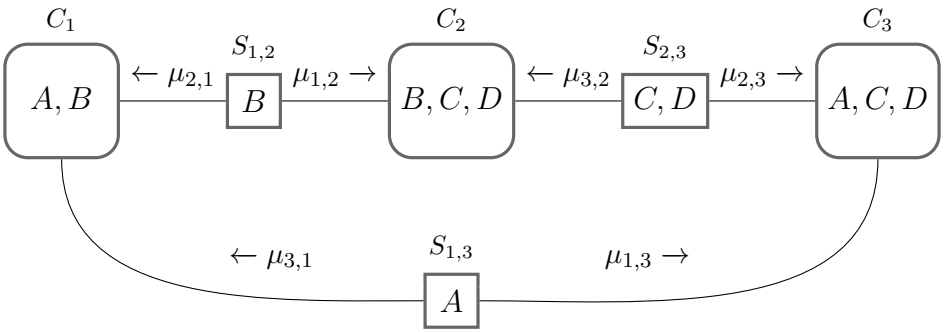


Figure 2.3: Example of a loopy cluster graph.

message $\mu_{2,3}(C)$ of the example graph in Figure 2.3 is calculated as

$$\mu_{2,3}(C, D) = \sum_{B} \phi_2(B, C, D)\mu_{1,2}(B).$$

Each cluster in the graph can be thought of as having its own subjective belief over how its variables are distributed. This belief is referred to as the *cluster belief* and denoted Ψ. It is defined as the product of the cluster factor $\phi_a(\mathcal{C}_a)$ with all incoming messages. The cluster belief of C_2 in Figure 2.3 is calculated as

$$\Psi_2(B, C, D) = \phi_2(B, C, D)\mu_{1,2}(B)\mu_{3,2}(C, D).$$

For a general graph with edges $\boldsymbol{S}$, the message and cluster belief distributions are calculated as

$$\mu_{a,b}(\mathcal{S}_{a,b}) = \sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \phi_a(\mathcal{C}_a) \prod_{i:S_{i,a} \in \boldsymbol{S}, i \neq b} \mu_{i,a}(\mathcal{S}_{i,a}), \tag{2.19}$$

$$\Psi_a(\mathcal{C}_a) = \phi_a(\mathcal{C}_a) \prod_{i:S_{i,a} \in \boldsymbol{S}} \mu_{i,a}(\mathcal{S}_{i,a}). \tag{2.20}$$

The cluster belief can be viewed as a local posterior belief based on its initial factor belief $\phi(\mathcal{C})$ and all the messages that it receives. A cluster sends messages to its neighbours to inform them on its own beliefs. These messages are passed iteratively between clusters until all the connected clusters agree on the marginal beliefs over the variables in the sepsets that connect them. When this state is reached, the cluster graph is said to be calibrated [2, p 358]. Formally, in a calibrated graph,

$$\sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \Psi_a(\mathcal{C}_a) = \sum_{\mathcal{C}_b \setminus \mathcal{S}_{a,b}} \Psi_b(\mathcal{C}_b), \tag{2.21}$$

for all messages in the graph.

The goal of message passing is to have the cluster beliefs $\Psi_a(\mathcal{C}_a)$ of the calibrated graph be close to the marginal distribution of $P_\Phi(\mathcal{C}_a)$. If this can be achieved, then message passing provides an efficient and tractable way of approximating the marginal distributions for large and complex probability distributions. Message passing is tractable because the execution time of

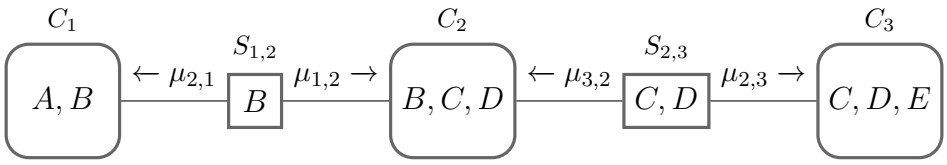


Figure 2.4: Example of a cluster graph that has a tree structure and satisfies the RIP.

the operations performed during message passing is dependent on the cluster and sepset sizes, not the size of the full joint distribution. The cluster graph invariant shows that a cluster graph can be viewed as an alternative representation of $P_\Phi(\mathcal{X})$ [2, Theorem 11.4].

Theorem 1 (Cluster Graph Invariant)**.** *At any point during message passing, a cluster graph satisfies*

$$P_\Phi(\mathcal{X}) = \frac{1}{Z} \frac{\prod_{i \in \mathcal{I}_\Phi} \Psi_i(\mathcal{C}_i)}{\prod_{S_{i,j} \in \boldsymbol{S}} \mu_{i,j}(\mathcal{S}_{i,j}) \mu_{j,i}(\mathcal{S}_{i,j})}. \tag{2.22}$$

This holds for any cluster graph and any message initialization.

This invariant property ensures that message passing cannot change the distribution represented by the cluster graph, it can only reparameterize it by 'moving information around'. It is easy to prove, since each message will occur once in the numerator as part of a cluster belief and once in the denominator. Dividing by all the messages effectively reverses anything done during message passing. We will now investigate the relationship between the cluster beliefs and the marginal distributions of P_Φ.

Exact Inference

There are two conditions that a cluster graph must satisfy for inference via message passing to be exact; that is, the calibrated cluster beliefs are the marginal distributions of the joint distribution. The first condition is that a cluster graph must have a tree structure. The second is that, for every random variable X in the graph, all clusters and sepsets containing X form a connected subgraph. The second condition is known as the *running intersection property* (RIP) as defined by Koller and Friedman [2, Definition 10.2].

Definition 1 (Running Intersection Property (RIP))**.** *A cluster graph satisfies the running intersection property when, for each random variable X in the graph, there exists a unique path connecting all the clusters containing X such that each sepset in the path contains X.*

As an example, let us calculate the cluster belief $\Psi_3(C, D)$ for the cluster graph in Figure 2.4 using message passing. The graph has a tree structure and satisfies the RIP. The cluster belief should therefore be equal to the marginal probability distribution of P_Φ. We start with the

definition of the cluster belief (Equation 2.20), and then calculate each message (Equation 2.19),

$$\begin{aligned}\Psi_3(C,D,E) &= \phi_3(C,D,E)\mu_{2,3}(C,D) \\ &= \phi_3(C,D,E)\sum_B \phi_2(B,C,D)\mu_{1,2}(B) \\ &= \phi_3(C,D)\sum_B \phi_2(B,C)\sum_A \phi(A,B).\end{aligned}$$

The summation terms can be moved outward,

$$\begin{aligned}\Psi_3(C,D,E) &= \sum_{A,B} \phi_3(C,D,E)\phi_2(B,C,D)\phi_1(A,B) \\ &= \sum_{A,B}\prod_{i\in\mathcal{I}_\Phi} \phi_i(\mathcal{X}_i),\end{aligned}$$

then substituting in Equation 2.14 gives

$$\Psi_3(C,D,E) = Z\sum_{A,B} P_\Phi(\mathcal{X}).$$

Therefore, the cluster belief is equal to the unnormalized marginal distribution $P_\Phi(C,D,E)$. This is true for all cluster graphs that are trees and satisfy the RIP

$$\sum_{\mathcal{X}\setminus\mathcal{C}_i} P_\Phi(\mathcal{X}) = \frac{1}{Z}\Psi_i(\mathcal{C}_i) \qquad \forall i \in \mathcal{I}_\Phi. \tag{2.23}$$

For a proof see Koller and Friedman Section 10.2.1.3 [2, p 353]. From Equation 2.23, we can compute the normalization constant Z from the cluster belief as

$$Z = \sum_{\mathcal{C}_i} \Psi_i(\mathcal{C}_i). \tag{2.24}$$

Most distributions in practice are not decomposable, which means their individual factors cannot be arranged to form a tree structure. By combining factors it is possible to create a new set of factors such that the resulting cluster graph forms a tree, but usually the new factors required are prohibitively large.

Approximate Inference

Message passing on cluster graphs that are not trees will, for most cases, not produce cluster beliefs that are equal to the correct marginal beliefs. Belief propagation performed on these graphs is referred to as *loopy belief propagation*.

In loopy graphs, each message is defined in terms of other messages that are in turn also defined in terms of other messages and so on. Eventually this will reach a message that is

dependent on the starting message. Therefore, a message cannot be calculated independently since itself is required for the calculation. An example of such a case can be seen in Figure 2.3. To overcome this deadlock state, messages are initialized to some starting distribution that is usually chosen to be a *vacuous* [2] distribution. Messages are then passed iteratively, where the same message is usually passed multiple times before the graph reaches a calibrated state. Messages in a calibrated graph will not change from one message passing iteration to the next. The process whereby a message tends toward its calibrated state is known as *convergence*, and a calibrated graph is said to have converged. Message passing in loopy graphs are not guaranteed to converge and may oscillate endlessly.

We can investigate the marginal distributions produced by loopy graphs by using the fact that having a tree-structure results in exact inference. Consider two sets of factors $\Phi = \{\phi_1(A,B), \phi_2(B,C,D), \phi_3(A,C,D)\}$ and $\Phi' = \{\phi'_1(A,B), \phi'_2(B,C,D)\}$. P_Φ is represented by the loopy graph in Figure 2.3 while $P_{\Phi'}$ is represented by the same graph but without cluster C_3. Therefore, the cluster graph describing $P_{\Phi'}$ is a subtree of the graph describing P_Φ. We can use this subtree to reason about the loopy graph.

Consider the case where we perform message passing on both graphs, resulting in both the loopy graph and the subtree to be calibrated, leading both to satisfy Equation 2.21. We have no guarantees about the cluster beliefs of the loopy graph, but we know the cluster beliefs of the subtree are the exact marginal distributions of $P_{\Phi'}$. We can construct the factors Φ' of the subtree to produce the same cluster beliefs of the loopy graph by multiplying the factors in Φ by the calibrated messages of the loopy graph as follows:

$$\phi'_1(A,B) = \phi_1(A,B)\mu_{3,1}(A), \tag{2.25}$$

$$\phi'_2(B,C,D) = \phi_2(B,C,D)\mu_{3,2}(C,D). \tag{2.26}$$

Therefore, the cluster belief for C_1 of the loopy graph is given by,

$$\begin{aligned}\Psi_1(A,B) &= \phi_1(A,B)\mu_{3,1}(A)\mu_{2,1}(B)\\ &= \phi'_1(A,B)\mu_{2,1}(B)\\ &= \Psi'_1(A,B).\end{aligned}$$

Since we know $\Psi'_1(A,B)$ and $\Psi_1(A,B)$ both are the exact marginal distribution of $P_{\Phi'}(A,B)$, it follows that if P_Φ is not equal to $P_{\Phi'}$, $\Psi_1(A,B)$ is not the exact marginal distribution of P_Φ.

[2] A vacuous distribution is defined such that the result of multiplying it with a probability distribution is equal to the probability distribution. A vacuous distribution is generally not a probability distribution.

The two joint distributions described by the loopy graph and the subtree are given by the product of their factors

$$P_\Phi(A,B,C,D) \propto \phi_1(A,B)\phi_2(B,C,D)\phi_3(A,C,D), \tag{2.27}$$
$$P_{\Phi'}(A,B,C,D) \propto \phi'_1(A,B)\phi'_2(B,C,D). \tag{2.28}$$

We can write P_Φ in terms of P'_Φ by substituting in Equations 2.25, 2.26 and 2.28 as follows:

$$\begin{aligned} P_\Phi(A,B,C,D) &\propto \phi_1(A,B)\phi_2(B,C,D)\phi_3(A,C,D) \qquad (2.29)\\ &= \frac{\phi'_1(A,B)}{\mu_{3,1}(A)}\frac{\phi'_2(B,C,D)}{\mu_{3,2}(C,D)}\phi_3(A,C,D) \\ &= P_{\Phi'}(A,B,C,D)\frac{\phi_3(A,C,D)}{\mu_{3,1}(A),\mu_{3,2}(C,D)}, \\ P_\Phi(A,B,C,D) &= \frac{1}{Z}P_{\Phi'}(A,B,C,D)\frac{\phi_3(A,C,D)}{\mu_{3,1}(A),\mu_{3,2}(C,D)}. \qquad (2.30) \end{aligned}$$

From the above equation we can see that only when $\phi_3(A,C,D) = \mu_{3,1}(A)\mu_{,2}(C,D)$ will P_Φ equal $P_{\Phi'}$, resulting in Ψ_1 and Ψ_2 to be the exact marginal distributions of P_Φ. Since there always exists a subtree within any general cluster graph, we can always show that, if the factors in Φ cannot be factorized further, the cluster beliefs of the loopy graph cannot be the marginal distributions of P_Φ [2, Section 11.3.3.2]. In fact, they may not be the marginal distributions of any distribution; that is to say, there does not exist a distribution whose marginal distributions are the cluster beliefs.

For loopy cluster graphs to achieve good approximate marginal distributions, they have to satisfy the RIP, which can restrict the graph structure to the extent that the quality of inference is reduced. The RIP is discussed in more detail in Section 3.1.

One might wonder why one would use loopy graphs if they have these limitations. The answer is that the marginal distributions produced by loopy graphs are often close to the true marginal distributions while potentially taking orders of magnitude less time to compute. Consider Equation 2.30 again. We can view $P_{\Phi'}$ as an approximate joint with an error of $\phi_3/(\mu_{3,1}\mu_{3,2})$. Since exact inference is intractable in most cases, even when using efficient inference algorithms, approximations are necessary.

2.4.4. The Region Graph Method

The region graph method and its accompanying general belief propagation algorithms, introduced by Yedidia, Freeman and Weiss, is a method that generalizes several other belief propagation algorithms [5]. Specifically, it generalizes standard message passing using cluster graphs as well as the cluster variation method that will be discussed in Subsection 2.5.1. This generalized message passing can produce improved approximate marginal distributions compared to standard message passing on cluster graphs for certain graph structures, and it is therefore important that it

is compared to conditional message passing. In this section, a brief explanation of region graphs and one of its accompanying message-passing algorithms will be given. Everything discussed in this section is adapted from Yedidia, Freeman and Weiss [5] for easy comparison with cluster graphs.

Region Graphs

Region graphs are directed acyclic graphs where the vertices are associated with sets of random variables known as regions. Vertices are denoted by R and the associated set of random variables are denoted by $\mathcal{R}$. A directed edge $E_{p,c}$ points from a parent region R_p to a child region R_c. A child region R_c must be a subset of its parent region R_p, $\mathcal{R}_c \subset \mathcal{R}_p$. If there exists a directed path from region R_a to region R_b, then region R_a is referred to as the *ancestor* of R_b and R_b as the *descendant* of R_a.

Similar to cluster graphs, regions can be assigned a factor ϕ if its scope is equal to or a subset of the region. As with cluster graphs, we assume that each region has one corresponding factor with equal scope,

$$\mathcal{R}_i = \mathcal{X}_i \qquad \forall i \in \mathcal{I}_\Phi.$$

This assumption can always be satisfied without altering $P_\Phi(\mathcal{X})$ by either combining factors in Φ or by adding vacuous factors to Φ. Vacuous factors are unnormalized uniform distributions that have no effect on a probability distribution.

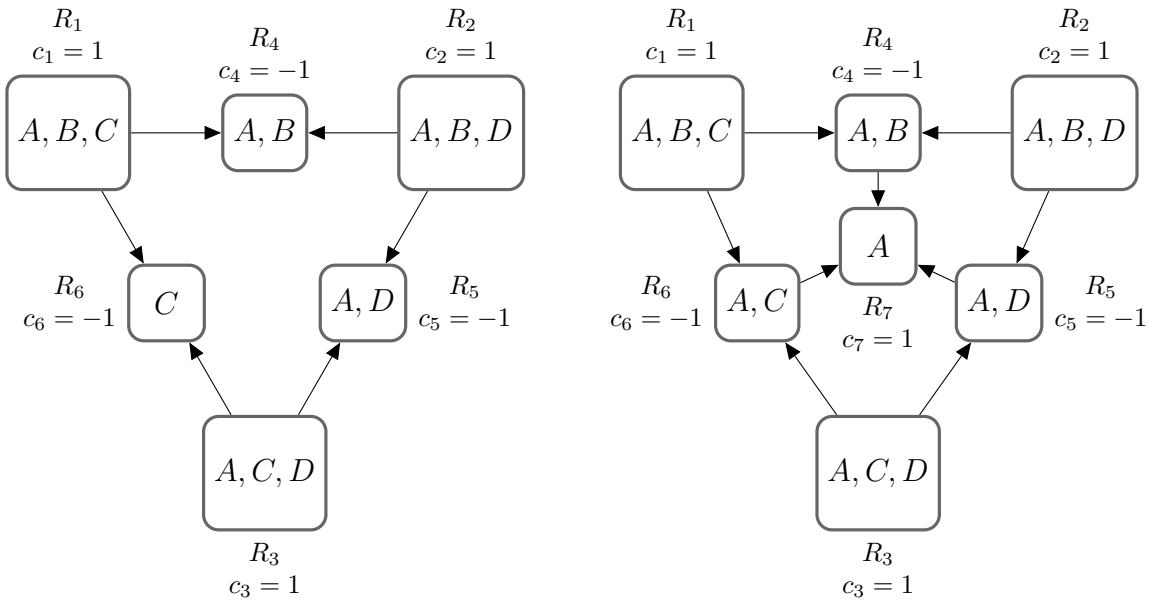
Every region in the graph is associated with a coefficient c referred to as a *counting number*. Counting numbers are calculated as

$$c_i = 1 - \sum_{j \in \mathcal{I}_\Phi} c_j [\mathcal{R}_j \text{ ancestor of } \mathcal{R}_i], \tag{2.31}$$

where the condition in within the brackets, known as Iverson brackets, reduces to 1 when true and 0 when false. The counting numbers ensure that the information of each region is 'counted' only once during inference. This idea is explored in more detail in Subsection 2.5.1. Region graphs also have a generalized version of the running intersection property that is referred to as the *region graph condition*.

Definition 2 (Region Graph Condition)**.** *For every variable X in the region graph, all the regions containing X must form a connected subgraph with counting numbers that sum to one.*

The region graph condition allows for a larger space of valid graphs than the running intersection property defined for loopy cluster graphs. Consequently, it can avoid the structural problems that cluster graphs face as a result of the RIP.



(a) A region graph that is equivalent to a cluster graph that satisfies the RIP.

(b) Region graph with regions obtained from the cluster variation method described is Section 2.5.1.

Figure 2.5: An example of two region graphs that both satisfy the region graph condition.

The Parent-to-Child Message-passing Algorithm

The parent-to-child message-passing algorithm is one of the belief propagation algorithms used for inference on region graphs. In this algorithm, messages are only passed from parent-to-child regions.

Unlike the message-passing algorithms discussed thus far, messages are not only a combination of incoming messages, but also other messages across the region graph. To find the relevant messages in the graph, we define $\mathcal{D}_i$ to be the set of regions that are descendants of the region R_i, and also *includes* R_i (see Figure 2.6). Each region has a region belief, similar to a cluster belief, which is denoted $\Psi_a(\mathcal{R}_a)$. The region belief is defined as the region factor ϕ_i and all incoming messages, including messages passed to its descendants from regions that are not descendants. For example, the region belief $\Psi_5(\mathcal{R}_5)$ in Figure 2.6 includes all messages passed into $\mathcal{D}_5$ as follows:

$$\Psi_5(\mathcal{R}_5) = \phi_5(\mathcal{R}_5)\mu_{1,5}(\mathcal{R}_5)\mu_{2,7}(\mathcal{R}_7)\mu_{3,6}(\mathcal{R}_6)\mu_{4,9}(\mathcal{R}_9). \tag{2.32}$$

The region belief for a general region graph, where $\mathcal{E}$ is the set of edges in the graph, is calculated as

$$\Psi_a(\mathcal{R}_a) = \phi_a(\mathcal{R}_a) \prod_{i,j:E_{i,j}\in\mathcal{E}} \mu_{i,j}(\mathcal{R}_j)^{[R_i \notin \mathcal{D}_a \,\wedge\, R_j \in \mathcal{D}_a]}. \tag{2.33}$$

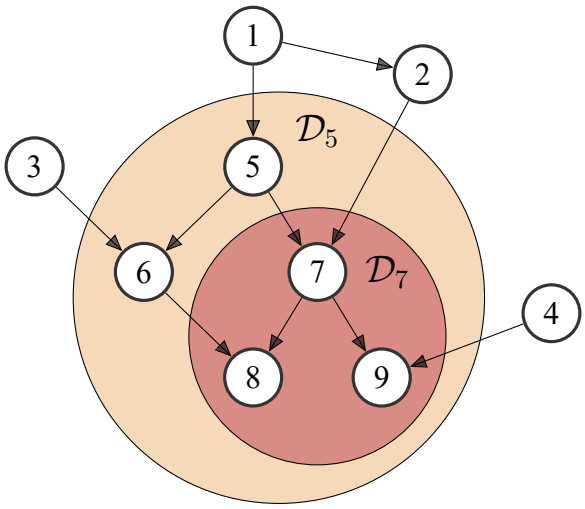


Figure 2.6: Region graph example showing the sets $\mathcal{D}_5$ and $\mathcal{D}_7$ for the regions R_5 and R_7 respectively. Each numbered vertex represents a region. The specific region scopes and counting numbers are omitted to place focus on the different parent and child relationships. For example, the region R_8 represented by vertex 8 is the child of R_6 and R_7 represented by the vertices 6 and 7.

From the region belief it can be shown that a message is calculated as follows:

$$\mu_{p,c}(\mathcal{R}_c) = \frac{\sum_{\mathcal{R}_p \backslash \mathcal{R}_c} \phi_p(\mathcal{R}_p) \prod_{E_{i,j} \in \mathcal{E}} \mu_{i,j}(\mathcal{R}_j)^{[R_i \notin \mathcal{D}_p \ \wedge \ R_j \in \mathcal{D}_p \backslash \mathcal{D}_c]}}{\prod_{E_{i,j} \in \mathcal{E}} \mu_{i,j}(\mathcal{R}_j)^{[R_i \in \mathcal{D}_p \ \wedge \ R_i \neq R_p \ \wedge \ R_j \in \mathcal{D}_c]}}. \tag{2.34}$$

Let us break down the message equation and calculate $\mu_{5,7}$ for the example graph in Figure 2.6. The messages in the numerator originate from regions outside $\mathcal{D}_5$ ($R_i \notin \mathcal{D}_p$) and arrive at regions that are in $\mathcal{D}_5$ but not in $\mathcal{D}_7$ ($R_j \in \mathcal{D}_p \backslash \mathcal{D}_c$). We therefore multiply with messages $\mu_{1,5}$ and $\mu_{3,6}$. The messages in the denominator originate from regions inside $\mathcal{D}_5$, except for R_5 ($R_i \neq R_p$), and arrive at regions within $\mathcal{D}_7$. The resultant message is then calculated as

$$\mu_{5,7}(\mathcal{R}_7) = \frac{\sum_{\mathcal{R}_5 \backslash \mathcal{R}_7} \phi_5(\mathcal{R}_5) \mu_{1,5}(\mathcal{R}_5) \mu_{3,6}(\mathcal{R}_6)}{\mu_{6,8}(\mathcal{R}_8)}. \tag{2.35}$$

As with cluster graphs, messages are passed iteratively until convergence.

The region graph method is derived by applying optimization techniques as well as the calculus of variations to probabilistic graphical models. In the next chapter this approach to inference and how it relates to PGMs is introduced.

2.5. Variational Inference

This section introduces variational inference, which approaches inference from a different perspective and provides more control over how the marginal distributions are approximated. Counting numbers, which are used by conditional message passing, will be motivated from a variational perspective.

In variational inference, instead of trying to *compute* the marginal distribution of the joint probability distribution, we *search* for a probability distribution that equals the marginal distribution. This might seem an even harder problem, as the search space can be large, and we need to know when we have found the marginal distribution. It turns out, however, that this approach is very useful. By using calculus of variations one can derive not only the message-passing schemes discussed thus far, but also other inference methods. It provides insight into why loopy message passing is so effective by rephrasing the problem as one of optimization. These methods are generally known as variational inference (see *Pattern Recognition and Machine Learning* [3]).

In calculus of variations we define a functional that takes one or more functions as input, and maps them to an output value. This is analogous to how a function maps input variables to an output value. The Kullback-Leibler divergence defined in Subsection 2.1.5 is an example of a functional and takes P and Q as inputs and returns their difference in nats. Just like in standard calculus, we can define operations on functionals, such as derivatives and integrals [3].

Now consider we use the KL divergence $D(P||Q)$ as an objective function and apply optimization techniques to minimize it. We therefore search for the distribution Q that will minimize $D(P||Q)$ for a specific probability distribution P. If we constrain the optimization such that Q is a probability distribution, then the minimum will be reached when Q is equal to P. We can further constrain Q, for example, by limiting it to only Gaussian distributions. The minimum will then be reached for the Gaussian distribution Q that is the best approximation of P according to this specific choice of objective function. This type of approximation is known as the *moment* projection (M-projection), while minimizing $D(Q||P)$ is known as the *information* projection (I-projection) [2].

2.5.1. The Cluster Variation Method

We will now investigate the cluster variation method as an approximate variational inference technique. This will provide some insight into counting numbers that are used in region graphs as well as the new message-passing scheme introduced in this thesis.

Let us again consider the factorized probability distribution $P_\Phi(\mathcal{X})$. The KL divergence

between P_Φ and Q is then

$$\begin{aligned} D(Q||P_\Phi) &= E_Q[\ln\left(\frac{Q}{P_\Phi}\right)] \\ &= E_Q[\ln(Q)] - E_Q[\ln(P_\Phi)]. \end{aligned} \tag{2.36}$$

Substituting P_Φ with Equation 2.14 we get

$$D(Q||P_\Phi) = E_Q[\ln(Q)] - E_Q\left[\sum_{i\in\mathcal{I}_\Phi}\ln(\phi_i(\mathcal{X}_i))\right] + \ln(Z), \tag{2.37}$$

which can be written in terms of other known functionals from statistical physics as

$$\begin{aligned} D(Q||P_\Phi) &= -\left(\overbrace{E_Q\left[\frac{1}{\ln(Q)}\right]}^{\text{entropy term } H_Q} + \overbrace{\sum_{i\in\mathcal{I}_\Phi} E_{Q(\mathcal{X}_i)}[\ln(\phi_i(\mathcal{X}_i))]}^{\text{energy term}}\right) + \ln(Z) \\ &= -F[\tilde{P}_\Phi, Q] + \ln(Z), \end{aligned} \tag{2.38}$$

where $F[\tilde{P}_\Phi, Q]$ is known as the energy functional with $\tilde{P}_\Phi$ being an unnormalized version of P_Φ. Since $\ln(Z)$ is independent of Q, we can minimize $D(Q||P_\Phi)$ by searching for Q that maximizes the energy functional [2]. We are specifically interested in computing the marginal distributions of P_Φ. We therefore need to rewrite the objective function in terms of the marginal distributions of Q; as we approach the maximum of the energy functional, the marginal distributions of Q will approach the marginal distributions of P_Φ. If we find the maximum of the energy functional, we find the marginal distributions of P_Φ. The energy term is already defined in terms of the marginal distributions of Q denoted by $Q(\mathcal{X}_i)$. The entropy term, however, is not specified in terms of the marginal distributions of Q. Luckily we can transform the entropy term by way of a Möbius inversion such that it is the weighted sum of marginal entropies [11]. The entropy of the marginal distribution $Q(\mathcal{X}_i)$ is denoted as $H_Q(\mathcal{X}_i)$. The resultant entropy formula is then

$$H_Q(\mathcal{X}) = \sum_{i\in\mathcal{I}_{\boldsymbol{B}}}\sum_{j\in\mathcal{I}_{\boldsymbol{B}}}(-1)^{n_j-n_i}H_Q(\mathcal{B}_j)[\mathcal{B}_j\subseteq\mathcal{B}_i], \tag{2.39}$$

where the set $\boldsymbol{B}$ contains $\mathcal{X}$ and every possible subset of $\mathcal{X}$, n_j and n_i are the number of variables in $\mathcal{B}_j$ and $\mathcal{B}_i$ respectively. Rearranging the summations, the entropy term can be simplified as

$$H_Q(\mathcal{X}) = \sum_{j\in\mathcal{I}_{\boldsymbol{B}}} c_j H_Q(\mathcal{B}_j), \tag{2.40}$$

$$\sum_{i\in\mathcal{I}_{\boldsymbol{B}}} c_i[\mathcal{B}_j\subseteq\mathcal{B}_i] = 1 \qquad \forall\, j\in\mathcal{I}_{\boldsymbol{B}}, \tag{2.41}$$

where the coefficients c_i are known as Möbius numbers [11] or counting numbers [4]. The square brackets are Iverson brackets and select which counting numbers are included within the summation. Equation 2.41 states that the counting numbers for a specific set and all its supersets must sum to one.

We can now find the exact marginal distributions of P_Φ by searching for the set of marginal distributions $Q(\mathcal{B}_i)$ that maximize the energy functional. Unfortunately this is intractable, since we are not restricting the complexity of Q and therefore calculating the marginal distributions $Q(\mathcal{B}_i)$ is as difficult as computing them for P_Φ.

Thus, we finally arrive at the cluster variation method, which introduces two approximations to make maximizing the energy functional tractable. The first is to approximate the entropy term by removing sets from $\boldsymbol{B}$. The approximate entropy term effectively ignores certain marginal entropies. For the second approximation, we replace the marginal distributions $Q(\mathcal{B}_i)$, with any arbitrary distribution $\Psi_i(\mathcal{B}_i)$ that satisfies the following normalization and compatibility constraints [11]

$$\sum_{\mathcal{B}_i} \Psi_i(\mathcal{B}_i) = 1, \tag{2.42}$$

$$\sum_{\mathcal{B}_i \setminus \mathcal{B}_j} \Psi_i(\mathcal{B}_i) = \Psi_j(\mathcal{B}_j). \tag{2.43}$$

We do not explicitly constrain Ψ to be a marginal distribution of the full joint Q. This combined with the simplified entropy term can make the optimization problem tractable, but also results in distributions Ψ that are not the marginal distributions of any joint distribution. Let us define the set $\boldsymbol{R}$ to be the reduced version of $\boldsymbol{B}$. The sets chosen to be included in $\boldsymbol{R}$ are known as *clusters*. The cluster variation method defines $\boldsymbol{R}$ as follows [5]: First a set of unique starting clusters is chosen such that no cluster is a subset of any other chosen cluster. These starting clusters can be the scopes of the factors in Φ. One then proceeds to compute all possible intersections of the starting clusters, and then all intersections of those clusters and so on, at each iteration taking the intersections of the clusters generated in the previous iteration and discarding any generated clusters that are subsets of other generated clusters. This continues until all the intersections are equal to the empty set. The starting clusters and the clusters generated in this way form $\boldsymbol{R}$. The resultant energy functional can then be written in a factorized form

$$\begin{aligned} F[\tilde{P}_\Phi, Q] &\approxeq \sum_{i \in \mathcal{I}_{\boldsymbol{R}}} c_i H_{\Psi_i}(\mathcal{R}_i) + c_i E_{\Psi_i}[\ln(\phi_i(\mathcal{X}_i))] \\ &= \sum_{i \in \mathcal{I}_{\boldsymbol{R}}} c_i F[\phi_i, \Psi_i]. \end{aligned} \tag{2.44}$$

Lagrangian multipliers can be used to derive equations for optimizing this approximate energy functional under the normalization and compatibility constraints mentioned [11]. A very similar approach can be used to derive message-passing equations for region graphs [5] and

cluster graphs [2, Section 11]. Figure 2.5b shows a region graph with regions that are equal to the clusters defined by the cluster variation method. The message equations of the parent-to-child message-passing scheme for this region graph are fixed-point equations of the approximate energy functional in Equation 2.44. In other words, it is necessary, but not sufficient, for the solution to the optimization problem to satisfy these message equations.

In general, when a graph is calibrated through message passing, a solution is found that satisfies the fixed-point equations that characterizes the optimum of some energy functional. The energy functional can be exact (tree structures) or approximate (loopy structures). The calibrated solution is also not necessarily the global optimum, since the fixed-point equations are not always sufficient.

2.6. Summary

In this chapter the problem of performing probabilistic inference was introduced. We looked at two approaches to inference. The first represents a distribution as a graph and uses message passing to find approximate marginal distributions. The other approach is one of optimization. In the latter approach, an objective function is minimized or maximized subject to certain constraints.

These two approaches provide two different criteria for producing a good approximate marginal distribution. PGMs use a graph structure and message-passing scheme that results in an approximate joint $P_{\Phi'}$ that has a small error (Equation 2.30). The cluster variation method tries to maximize an approximate energy functional that is close to the true energy functional (Equation 2.44).

The rest of this thesis will be focusing on cluster graphs. The novel message-passing algorithm developed in the following chapter will largely resemble standard message passing for cluster graphs, with some aspects resembling ideas from the region graph method and the cluster variation method.

Chapter 3

Conditional Message Passing

This chapter introduces a novel message-passing algorithm for performing inference using cluster graphs. The message-passing scheme, referred to as *conditional message passing*, improves upon standard message passing by allowing for a larger space of valid cluster graphs.

An investigation of cluster graphs satisfying the running intersection property will be made and some of their limitations discussed. Alternative cluster graph structures that address these limitations will be investigated after which their graph structures will be defined by novel properties, namely the multiple running intersection property (MRIP) and the total running intersection property (TRIP).

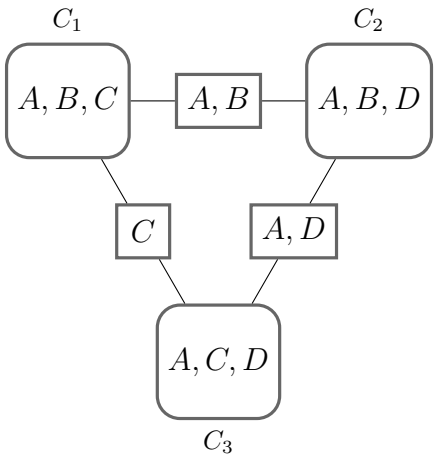
After the desired graph structures have been defined by the MRIP and TRIP, conditional message-passing is introduced that can be used for performing inference on these graphs. An algorithm for automatically constructing graphs on which conditional message passing can be performed will be presented along with some message optimization.

3.1. The Running Intersection Property (RIP)

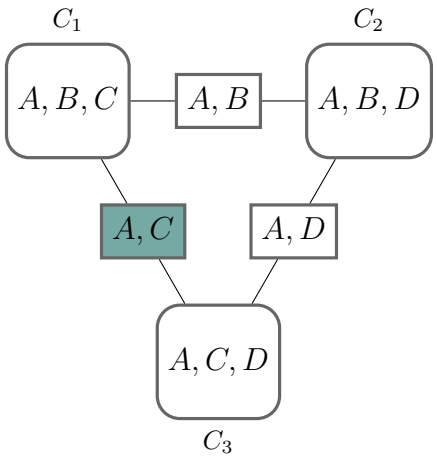
In Section 2.4.3, we saw that the RIP is necessary for exact inference on cluster graphs with tree structures. A generalized version of the RIP is used for general cluster graphs. For these graphs, failure to satisfy the RIP results in inference via standard message-passing algorithms to produce poor approximate marginal distributions. The definition of the RIP for general cluster graphs is given as follows [2, p. 396]:

Definition 3 (RIP)**.** *A cluster graph satisfies the running intersection property when, for each variable X in the graph, all the clusters containing X are connected via a* single *path, such that each sepset in the path contains X.*

Figure 3.1a shows an example of a cluster graph that satisfies the RIP. By requiring that all clusters sharing X be connected by a path containing X, calibrated clusters will agree on the marginal distribution over X [2, p. 397]. Additionally, by allowing only a *single* path to contain X between any two clusters, loops are prevented from forming by which information about X can loop endlessly. These feedback loops would result in clusters directly influencing their own beliefs through outgoing information from the cluster returning as “new” information via the loop.



(a) Cluster graph that satisfies the RIP. In this example sepset $\mathcal{S}_{1,3}$ excludes variable A in order to satisfy the RIP.



(b) Cluster graph that satisfies the TRIP. Sepset $\mathcal{S}_{1,3}$ now includes variable A. The messages between C_1 and C_3 are conditioned on A to prevent belief over A from directly looping.

Figure 3.1: A comparison between graphs that satisfy the RIP and the TRIP, showing how the TRIP allows for larger sepsets. The larger messages require a conditioning factor so that the messages do not form feedback loops where a cluster is influenced by its own belief returning through another path.

As an example, consider the graph in Figure 3.1b that does not satisfy the RIP. The belief that C_1 has over A is passed to C_2; C_2 then uses its updated belief to update C_3 that, in turn, will update C_1. However, C_1 is not aware that the message it receives from C_3 contains the information from the message it initially sent to C_2. Therefore, C_1 incorporates its own belief as if it were new information, commonly leading to overconfidence. This does not happen to such a large extent in Figure 3.1a, where the RIP is satisfied, since the message from C_3 will not include the belief about A. Omitting A from sepset $\mathcal{S}_{1,3}$ does reduce the amount of feedback, but it does not eliminate it. When A is marginalized out to create the message, it affects the belief over C, which in turn affects the belief the receiving cluster has over A, causing an indirect loop. Therefore, the RIP only prevents the most obvious of feedback loops from forming since information about X can still loop indirectly due to the correlations it has with variables in other paths [2, p. 397].

The constraints imposed by the RIP create cluster graphs that have some limitations. For instance, although the RIP leads to clusters that agree on the marginal distributions over *single* variables, it does not necessarily lead to clusters that agree on the *joint* marginal distributions over all the variables they share [2, p. 397]. This happens when a loopy graph is required by the RIP to make certain sepsets smaller than the full intersection of the clusters they connect. Figure 3.1a illustrates this through an example where sepset $S_{1,3}$ omits variable A in order to satisfy the running intersection property. By omitting variables from sepsets, the correlation between random variables is not propagated.

3.2. Reducing Constraints on Graph Structure

The goal of this thesis is to create a message-passing scheme that achieves better approximate marginal beliefs by removing the constraint that there may only exist a *single* path connecting two clusters where the same variable occurs in every sepset along that path. This goal can be achieved by constraining the belief passed by messages rather than constraining the graph structure. As will be shown, this leads to a message-passing scheme that results in calibrated cluster beliefs that are closer to the true marginal beliefs. To this end, a relaxed version of the RIP is proposed that allows for multiple paths to exist between two clusters where their running intersections contain the same variables. This version of the RIP when applied on tree structures is essentially the same as the one used for trees in Subsection 2.4.3. To distinguish the two, the novel *multiple running intersection property* (MRIP) is defined as follows:

Definition 4 (MRIP)**.** *A cluster graph satisfies the multiple running intersection property when, for each variable X in the graph, all the clusters containing X are connected via at least one path, such that each sepset in the path contains X.*

The reason for allowing multiple paths is to allow joint information to be propagated where it otherwise might not be due to the RIP. Since the goal is to enable the propagation of joint information, a property that ensures maximal propagation of joint information is also created. This property is referred to as the *total running intersection property* (TRIP) and it is defined as:

Definition 5 (TRIP)**.** *A cluster graph satisfies the total running intersection property when, for each pair of clusters with scope $\mathcal{C}_a$ and $\mathcal{C}_b$, there exist one or more paths connecting them such that each sepset in that path contains their intersection $\mathcal{C}_a \cap \mathcal{C}_b$.*

This definition is the same as the definition for the RIP as defined by Barber for junction trees [9, p 113], but the novelty of the TRIP is that it applies to loopy cluster graphs as well as trees. The TRIP property ensures marginal consistency between clusters in a calibrated graph. In other words, any two clusters are guaranteed to agree on the joint marginal distribution over the variables they share.

The three properties MRIP, TRIP and RIP are not mutually exclusive. Cluster graphs may satisfy more than one property at a time as illustrated by Figure 3.2. If a cluster graph has a tree structure and satisfies one of the three properties it also satisfies the other two. It is also possible for loopy cluster graphs to satisfy all three simultaneously, in which case conditional message passing will not improve upon standard message passing.

Now that we have a formal way of describing valid cluster graph structures, we can investigate a message-passing scheme that is suited for cluster graphs that adhere to the MRIP and TRIP.

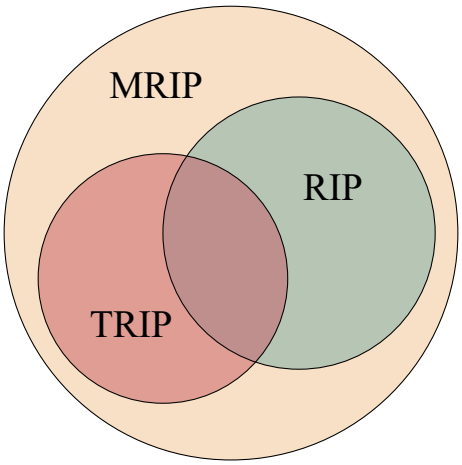


Figure 3.2: A Venn diagram showing the relationship between the different properties we defined for cluster graphs. If a cluster graph satisfies the RIP or the TRIP, it also satisfies the MRIP but not vice versa. MRIP allows for a larger space of valid cluster graphs than the RIP.

3.3. Conditional Messages

In this section, conditional message passing is proposed as a novel method for constraining the information within a message to prevent belief in a cluster graph from propagating through the multiple loops that are now allowed to exist by the MRIP and the TRIP. Conditional message passing can therefore be seen as including loop correction, specifically correcting for loops that are usually prevented by the RIP.

Consider Figure 3.1a. Variable A can be included in the message $\mu_{1,3}$ from C_1 to C_3 (shown in Figure 3.1b) as long as the information about A that will reach C_3 through the messages $\mu_{1,2},\ \mu_{2,3}$ is removed. In this example, the belief that C_1 has over A already has a path through which to reach C_3 and must be removed from the expanded message $\mu_{1,3}$. The example message should therefore be

$$\mu_{1,3}(A,C) = \frac{\sum_B \phi_1(A,B,C)\mu_{2,1}(A,B)}{\sum_{B,C} \Psi_1(A,B,C)}, \tag{3.1}$$

$$= \frac{\mu_{1,3}^{\text{standard}}(A,C)}{\Psi_1(A)}, \tag{3.2}$$

where one first calculates the message as usual (without the denominator as shown in Equation 2.19), and then "remove" the belief C_1 has over A by dividing the message by the marginal distribution $\Psi_1(A)$. The marginal distribution $\Psi_1(A)$ can be interpreted as containing a summation of the belief that the cluster has over A. This newly defined message will be referred to as a conditional message since it is conditioned by a factor to prevent information from looping. Note that in general, a conditional message is *not* a conditional distribution as defined in Subsection 2.1.3.

To help with the general definition of a conditional message a factor $\Theta_{a,b}$ with scope $\mathcal{Q}$ is defined to be a distribution that contains all the information that needs to be removed from a

message to limit the looping of information. The calculation of this factor will be discussed in detail in the following sections. The general formula for calculating a conditional message is then

$$\mu_{a,b}(\mathcal{S}_{a,b}) = \frac{\sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \phi_a(\mathcal{C}_a) \prod_{S_{i,a} \in \boldsymbol{S}, i \neq b} \mu_{i,a}(\mathcal{S}_{a,b})}{\Theta_{a,b}(\mathcal{Q}_{a,b})}, \tag{3.3}$$

where $\Theta_{a,b}(\mathcal{Q}_{a,b})$ contains all the information that can propagate from cluster C_a to C_b through other paths in the graph (and therefore should not be passed on by message $\mu_{a,b}$). $\Theta_{a,b}$ is referred to as the conditioning factor. See Section 3.4 on how the conditioning factor is calculated. Apart from $\Theta_{a,b}$, a message is calculated exactly as with standard loopy belief propagation as defined in Subsection 2.4.3.

The conditional message-passing algorithm will include a mixture of standard and conditional messages. Whether a message is conditional, and its accompanying conditioning factor, depends on the structure of the graph and is determined in a preprocessing step before the messages are passed.

The definition of a cluster belief for conditional message passing is the same as that of standard message passing (Equation 2.20). As a result the cluster graph invariant (Definition 1) also holds for conditional message passing.

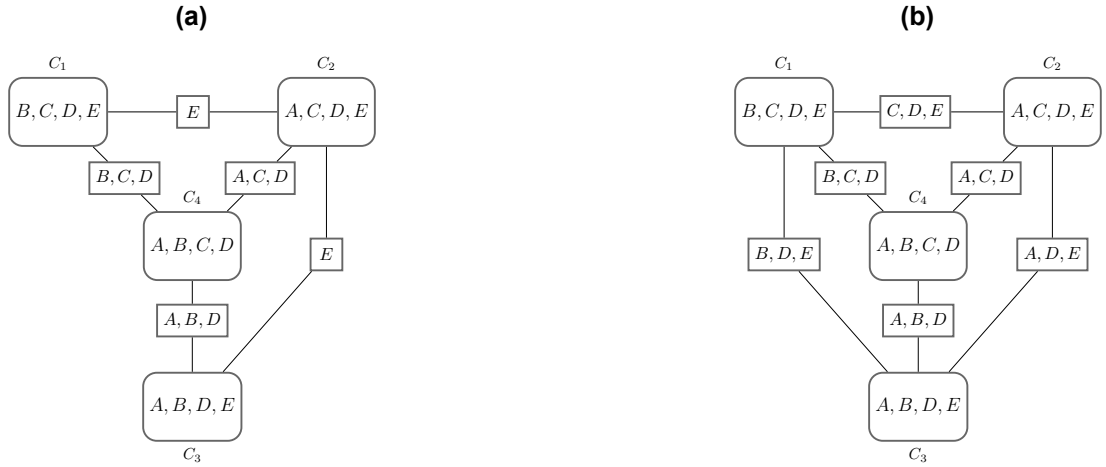


Figure 3.3: Figure 3.3a is an example of a cluster graph that satisfies the RIP. Figure 3.3b shows the same clusters but connected such that the graph satisfies the TRIP.

3.4. The Conditioning Factor

This section discusses one possible process for determining the conditioning factors $\Theta_{a,b}(\mathcal{Q}_{a,b})$. The general idea is as follows: Edges are iteratively added to some starting cluster graph. The structure of the graph at each iteration is used to determine the conditioning factor for the edge that is being added. The starting graph can be any graph that does not already contain loops where one or more variables are present through the entire loop. The starting graph does not need to contain any edges (it could just be a set of clusters), or it could be a cluster graph that satisfies the RIP, as well as anything in between. The algorithm can then iteratively add additional edges (if needed) and this process can continue for any number of edges or until the graph satisfies either the MRIP or the TRIP. The process only has to be performed once for any specific graph.

This process will be illustrated in more detail by applying it to the example starting graph in Figure 3.3b. After this the general case will be considered and finally an algorithm that can automatically construct the cluster graph and determine the conditioning factors for any set of clusters will be developed.

A concept that will be used in the following sections is the notion of a running intersection. A running intersection is a set of random variables, denoted $\mathcal{R}$, that is obtained from the intersection of all the sepsets along a path in the cluster graph. The i^{th} path between C_a and C_b is denoted by $p^i_{a,b}$ and is a sequence of distinct edges. The running intersection of a path can be calculated as

$$\mathcal{R}_{a,b} = \bigcap_{S_{m,n} \in p_{a,b}} \mathcal{S}_{m,n}. \tag{3.4}$$

3.4.1. Conditioning Factor Calculation Example

In this subsection the conditioning factors required for the cluster graph is in Figure 3.3b is calculated. This is done by inspecting, in no particular order, the running intersections between

the pairs of clusters in the cluster graph in Figure 3.3a.

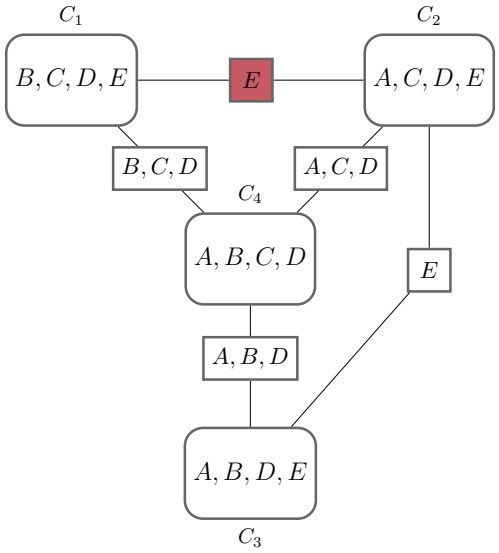


Figure 3.4: This cluster graph satisfies the RIP and is the starting graph for the conditioning factor calculation example. The first sepset to be inspected is indicated in red.

i) We start by inspecting the non-empty running intersections between C_1 and C_2 of the starting graph as shown above (see Equation 3.4 for how a running intersection is calculated). The running intersections are

   (a) $C_1 - C_2 : \{E\}$

   (b) $C_1 - C_4 - C_2 : \{C, D\}$.

   Currently, there does not exist a running intersection containing their full intersection $C_1 \cap C_2 = \{C, D, E\}$ and therefore the scope of $S_{1,2}$ is expanded to $\{C, D, E\}$. To ensure the expanded messages do not contain information sent through $C_1 - C_4 - C_2$ we divide by the accompanying conditioning factor. The conditioning factor contains the information that goes through path (b) and is calculated as

$$\Theta_{a,b}(C, D) = \sum_{\mathcal{C}_a \setminus \{C,D\}} \Psi_a(\mathcal{C}_a).$$

   Therefore, the conditioning factor for the messages $\mu_{1,2}$ and $\mu_{2,1}$ will be

$$\Theta_{1,2}(C, D) = \sum_{B,E} \Psi_1(B, C, D, E), \tag{3.5}$$

   and

$$\Theta_{2,1}(C, D) = \sum_{A,E} \Psi_2(A, C, D, E)$$

   respectively.

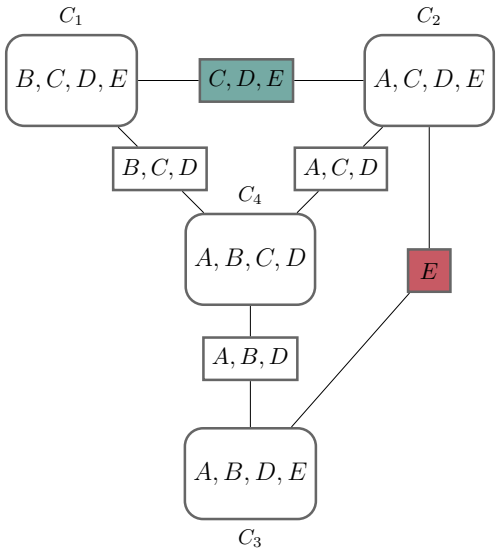


Figure 3.5: This cluster graph no longer satisfies the RIP due to the expansion of sepset $\mathcal{S}_{1,2}$ in the previous step (indicated in blue-green) but still satisfies the MRIP.

ii) We repeat the process in step i, choosing to expand sepset $\mathcal{S}_{2,3}$. Before the expansion the running intersections between C_2 and C_3 are:

(a) $C_2 - C_3 : \{E\}$

(b) $C_2 - C_4 - C_3 : \{A, D\}$

(c) $C_2 - C_1 - C_4 - C_3 : \{D\}$

Since none of the existing running intersections include the intersection of C_2 and C_3 we expand the sepset to be $\mathcal{S}_{2,3} = \{A, D, E\}$. Notice that the running intersection in (c) is a subset of (b) and is due to the graph no longer satisfying the RIP. The conditioning factors $\Theta_{1,2}(C, D)$ and $\Theta_{2,1}(C, D)$ added in the previous step will prevent information about D from flowing through path (c). We can therefore ignore that running intersection. In general, any running intersection that is a subset of another running intersection can be ignored for this reason. The accompanying conditioning factor for messages $\mu_{2,3}(A, D, E)$ and $\mu_{3,2}(A, D, E)$ need to remove information sent via path (b) and is therefore calculated as

$$\Theta_{a,b}(A, D) = \sum_{\mathcal{C}_a \setminus \{A,D\}} \Psi_a(\mathcal{C}_a).$$

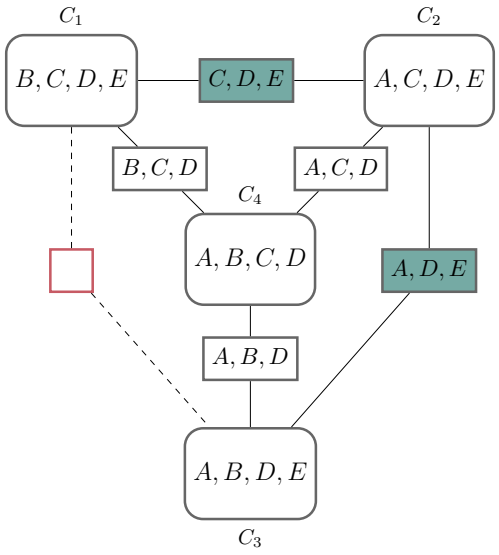


Figure 3.6: As with the cluster graph in the previous step, this cluster graph no longer satisfies the RIP but still satisfies the MRIP. In this stage of the example we consider adding an edge between C_1 and C_3 indicated in red.

iii) Finally, we *add* a sepset between C_1 and C_3. Before the addition the running intersections are:

(a) $C_1 - C_4 - C_3 : \{B, D\}$

(b) $C_1 - C_2 - C_3 : \{D, E\}$

(c) $C_1 - C_2 - C_4 - C_3 : \{D\}$

(d) $C_1 - C_4 - C_2 - C_3 : \{D\}$

To ensure no direct information loop is added, we need to calculate what information can already be communicated between C_1 and C_3, taking the effect of conditioning factors already present in the graph into account. When looking at the running intersections we notice that (c) and (d) can be omitted since they are subsets of other running intersections. However, (a) and (b) overlap, having an intersection containing D. In a normal graph, this overlap would lead to a direct information loop, but the conditioning factors already present prevent this. The current information that is passed along (a) and (b) is calculated as

$$\Theta_{a,b}(B, D, E) = \frac{\sum_{\mathcal{C}_a \setminus \{B,D\}} \Psi_a(\mathcal{C}_a) \sum_{\mathcal{C}_a \setminus \{E,D\}} \Psi_a(\mathcal{C}_a)}{\sum_{\mathcal{C}_a \setminus \{D\}} \Psi_a(\mathcal{C}_a)}.$$

The two terms in the numerator correspond to the two existing running intersections. The term in the denominator prevents the conditioning factor from removing the information about D more than once from the message.

The final graph satisfies the TRIP and is shown in figure 3.3b. The example illustrates that by building the graph iteratively, it is possible to limit the severe feedback loops that occur when the same variables are present in every sepset in a loop.

By using the logic shown in the example, it is possible to determine the conditioning factors for any TRIP graph. Changing the order in which the sepsets are expanded changes the conditioning factors and can also change the structure of the final graph. The optimal starting graph and order of expansion is far from obvious. Intuitively, the number of sepsets that need to be added as well as the number of terms in the conditioning factors should be minimized.

To implement this for an arbitrary graph, one needs to be able to determine the running intersections between any two clusters. Note that it is not necessary to calculate all possible paths between two clusters, since in loopy graphs this could be infinite. One only needs to calculate all *unique* running intersections, excluding those who are subsets of other intersections. The running intersections can be determined by a process similar to a breadth-first search that is limited to a subgraph, where each cluster in the subgraph contains at least one variable from the set $\mathcal{C}_a \cap \mathcal{C}_b$.

In summary, by looking at the running intersections between clusters and by using the knowledge that at any stage in the process the graph does not have any direct information loops, the form of the existing belief that is being propagated between two clusters can be determined and used to prevent information from looping. The key property on which the method relies is the fact that both before and after adding or expanding an edge, there are no direct information loops thanks to the accompanying conditioning factor preventing them. This allows us to approach every iteration of the algorithm the same as the first iteration, which greatly simplifies the problem.

3.4.2. General Calculation of the Conditioning Factor

In the previous section, we saw how the conditioning factor is determined from the running intersection of existing paths between clusters. Here this process will be described for the general case when *adding* an edge. Edges will not be expanded since extra care needs to be taken when expanding edges for more complicated graphs than the one used in Subsection 3.4.1. Instead of expanding edges, one modifies the starting graph by removing all the edges that need to be expanded. During the iterative process of adding edges, the removed edges may be added back having the net effect of being expanded.

Consider the case where the edge $S_{a,b}$ is being added to some arbitrary graph, and we want to compute the conditioning factor $\Theta_{a,b}(\mathcal{Q}_{a,b})$ for that edge. Let N be the number of existing paths $p_{a,b}$ between C_a and C_b before the edge $S_{a,b}$ is added

$$\boldsymbol{p}_{a,b} = \{p^1_{a,b}, p^2_{a,b}, \cdots, p^N_{a,b}\}. \tag{3.6}$$

We exclude paths from $\boldsymbol{p}_{a,b}$ whose running intersection are subsets of other running intersections.

Using Equation 3.4, we can compute the running intersections for each path in $\boldsymbol{p}_{a,b}$ as

$$\mathcal{R}^n_{a,b} = \bigcap_{S_{i,j} \in p^n_{a,b}} \mathcal{S}_{i,j}, \qquad \forall\, n : p^n_{a,b} \in \boldsymbol{p}_{a,b}, \tag{3.7}$$

$$\boldsymbol{\mathcal{R}}_{a,b} = \{\mathcal{R}^1_{a,b}, \mathcal{R}^2_{a,b}, \cdots, \mathcal{R}^N_{a,b}\}. \tag{3.8}$$

We want to determine what information needs to be removed from the message $\mu_{a,b}$.

Let us define $\boldsymbol{\mathcal{V}}_{a,b}$ as the set that contains the sets of random variables whose marginal distribution need to be divided out of the message and therefore must form part of the conditioning factor. Each set of random variables in $\boldsymbol{\mathcal{R}}_{a,b}$ indicates the scope of an existing path through which joint information can propagate between C_a and C_b. These sets indicate joint distributions that must be removed from $\mu_{a,b}$ and should therefore be in $\boldsymbol{\mathcal{V}}_{a,b}$.

It is however possible that the sets in $\boldsymbol{\mathcal{R}}_{a,b}$ overlap, resulting in the information of some subset of variables being divided out more than once by $\boldsymbol{\mathcal{V}}_{a,b}$. Since we require that there are currently no information loops in the graph, we know that the overlap, which is as a result of two paths containing the same random variables, does not mean that both paths communicate information about those variables. The conditioning factor should therefore not divide the information out more than once. To this end we define $\boldsymbol{\mathcal{W}}_{a,b}$ as the set of sets whose marginal distributions must be divided out of the conditioning factor and add to it the intersections of the sets in $\boldsymbol{\mathcal{R}}_{a,b}$. These added sets will correct for any information that would otherwise be divided out more than once by the conditioning factor. However, the same is true for the sets added to $\boldsymbol{\mathcal{W}}_{a,b}$: any overlap indicates information that is removed from the conditioning factor more than once and should be multiplied with the conditioning factor. This process continues until there is no overlap. We can use intermediary sets $\boldsymbol{\mathcal{V}}^i_{a,b}$ and $\boldsymbol{\mathcal{W}}^i_{a,b}$ to help calculate $\boldsymbol{\mathcal{V}}_{a,b}$ and $\boldsymbol{\mathcal{W}}_{a,b}$. The function 'intersections($\boldsymbol{\mathcal{V}}^i_{a,b}$)' returns the set of all possible intersections between the sets in $\boldsymbol{\mathcal{V}}^i_{a,b}$. The intermediary sets are then calculated as

$$\begin{aligned}
\boldsymbol{\mathcal{V}}^1_{a,b} &= \boldsymbol{\mathcal{R}}_{a,b}, \\
\boldsymbol{\mathcal{W}}^1_{a,b} &= \text{intersections}(\boldsymbol{\mathcal{V}}^1_{a,b}), \\
\boldsymbol{\mathcal{V}}^2_{a,b} &= \text{intersections}(\boldsymbol{\mathcal{W}}^1_{a,b}), \\
&\vdots, \\
\boldsymbol{\mathcal{W}}^{K-1}_{a,b} &= \text{intersections}(\boldsymbol{\mathcal{V}}^{K-1}_{a,b}), \\
\boldsymbol{\mathcal{V}}^K_{a,b} &= \text{intersections}(\boldsymbol{\mathcal{W}}^{K-1}_{a,b}), \\
\emptyset &= \text{intersections}(\boldsymbol{\mathcal{V}}^K_{a,b}).
\end{aligned}$$

The union of all the intermediary sets gives us $\boldsymbol{\mathcal{V}}_{a,b}$ and $\boldsymbol{\mathcal{W}}_{a,b}$

$$\boldsymbol{\mathcal{V}}_{a,b} = \bigcup_{k=1}^{K} \boldsymbol{\mathcal{V}}_{a,b}^{k},$$
$$\boldsymbol{\mathcal{W}}_{a,b} = \bigcup_{k=1}^{K-1} \boldsymbol{\mathcal{W}}_{a,b}^{k}.$$

We now have the required sets, but the sets on their own are not enough to ensure that the joint marginal distribution of every set and subset in $\boldsymbol{\mathcal{R}}_{a,b}$ will only be removed once by the conditioning factor. For this to be the case, the sets in $\boldsymbol{\mathcal{V}}_{a,b}$ and $\boldsymbol{\mathcal{W}}_{a,b}$ need weights. These weights will be referred to as counting numbers and denoted with v_i and w_i, for the sets $\mathcal{V}_i \in \boldsymbol{\mathcal{V}}_{a,b}$ and $\mathcal{W}_i \in \boldsymbol{\mathcal{W}}_{a,b}$ respectively. The goal is to have every set and subset be divided out once more than is multiplied into the message $\mu_{a,b}$ by $\Theta_{a,b}(\mathcal{Q}_{a,b})$. From this we have that, for any set $\mathcal{X} \in \boldsymbol{\mathcal{V}}_{a,b} \cup \boldsymbol{\mathcal{W}}_{a,b}$, the counting numbers must satisfy

$$\sum_{i \in I_{\boldsymbol{\mathcal{V}}_{a,b}}} v_i[\mathcal{X} \subseteq \mathcal{V}_i] - \sum_{i \in I_{\boldsymbol{\mathcal{W}}_{a,b}}} w_i[\mathcal{X} \subseteq \mathcal{W}_i] = 1. \tag{3.9}$$

We can now define the conditioning factor as

$$\mathcal{Q}_{a,b} = \mathcal{R}_{a,b}^{1} \cup \mathcal{R}_{a,b}^{2} \cup \cdots \cup \mathcal{R}_{a,b}^{N}, \tag{3.10}$$
$$\Theta_{a,b}(\mathcal{Q}_{a,b}) = \frac{\prod_{i \in I_{\boldsymbol{\mathcal{V}}_{a,b}}} \left(\sum_{\mathcal{C}_a \setminus \mathcal{V}_i} \Psi_a(\mathcal{C}_a)\right)^{v_i}}{\prod_{i \in I_{\boldsymbol{\mathcal{W}}_{a,b}}} \left(\sum_{\mathcal{C}_a \setminus \mathcal{W}_i} \Psi_a(\mathcal{C}_a)\right)^{w_i}}. \tag{3.11}$$

We have encountered this idea of weighted distributions in Subsection 2.5.1 when discussing the cluster variation method. There, the weights, or counting numbers, were also required to add up to one. We can combine our two sets of weighted distribution into a single set by defining $\boldsymbol{\mathcal{U}}_{a,b} = \boldsymbol{\mathcal{V}}_{a,b} \cup \boldsymbol{\mathcal{W}}_{a,b}$ and the counting number u for every set in $\boldsymbol{\mathcal{U}}_{a,b}$. The sets in $\boldsymbol{\mathcal{U}}_{a,b}$ will be referred to as conditioning sets. From Equation 2.41 the counting numbers u_i must satisfy

$$\sum_{i \in I_{\boldsymbol{\mathcal{U}}_{a,b}}} u_i[\mathcal{U}_j \subseteq \mathcal{U}_i] = 1, \qquad \forall\, j \in I_{\boldsymbol{\mathcal{U}}_{a,b}} \tag{3.12}$$

from which u_j can be calculated as

$$\sum_{i \in I_{\boldsymbol{\mathcal{U}}_{a,b}}} (u_i[\mathcal{U}_j \subset \mathcal{U}_i] + u_i[\mathcal{U}_j = \mathcal{U}_i]) = 1, \tag{3.13}$$
$$\sum_{i \in I_{\boldsymbol{\mathcal{U}}_{a,b}}} u_i[\mathcal{U}_j \subset \mathcal{U}_i] + u_j = 1,$$
$$u_j = 1 - \sum_{i \in I_{\boldsymbol{\mathcal{U}}_{a,b}}} u_i[\mathcal{U}_j \subset \mathcal{U}_i].$$

The conditioning factor can then be written as

$$\Theta_{a,b}(\mathcal{Q}_{a,b}) = \prod_{i \in I_{\boldsymbol{\mathcal{U}}_{a,b}}} \left(\sum_{\mathcal{C}_a \setminus \mathcal{U}_i} \Psi_a(\mathcal{C}_a) \right)^{u_i}. \tag{3.14}$$

We now have the equations for determining a conditioning factor. Let us apply them to the example graph in Subsection 3.4.1. Specifically let us redo step (iii) where the edge $S_{1,3}$ is added. The running intersections before the addition are $\boldsymbol{\mathcal{R}}_{1,3} = \{\{B, D\}, \{D, E\}\}$. Taking these sets and all their intersections we get $\boldsymbol{\mathcal{U}} = \{\{B, D\}, \{D, E\}, \{D\}\}$. Finally, using Equation 3.13, the counting numbers are calculated to be $\{u_1 = 1, u_2 = 1, u_3 = -1\}$. Plugging in the sets and their counting numbers into Equation 3.14 gives

$$\begin{aligned} \Theta_{1,3}(\mathcal{Q}_{1,3}) &= \left(\sum_{\mathcal{C}_1 \setminus \mathcal{U}_1} \Psi_1(\mathcal{C}_1) \right)^{u_1} \left(\sum_{\mathcal{C}_1 \setminus \mathcal{U}_2} \Psi_1(\mathcal{C}_1) \right)^{u_2} \left(\sum_{\mathcal{C}_1 \setminus \mathcal{U}_3} \Psi_1(\mathcal{C}_1) \right)^{u_3} \\ &= \frac{\sum_{\mathcal{C}_1 \setminus \{B,D\}} \Psi_1(\mathcal{C}_1) \sum_{\mathcal{C}_1 \setminus \{D,E\}} \Psi_1(\mathcal{C}_1)}{\sum_{\mathcal{C}_1 \setminus \{D\}} \Psi_1(\mathcal{C}_1)}. \end{aligned} \tag{3.15}$$

As expected, the conditioning factor is the same as previously computed in the example.

3.4.3. Algorithm for Determining the Conditioning Factor

This subsection combines the ideas discussed in the previous two subsections to create a novel algorithm that can construct a cluster graph and all the accompanying conditioning sets and counting numbers needed for performing inference using conditional message passing. It can be used to construct the entire graph from scratch using the scopes of the factors in Φ. Alternatively it can start with an existing graph as input so long as the graph does not contain direct information loops.

The novel method for constructing a cluster graph is shown in Algorithm 1 with the functions for calculating the conditioning sets (Function calcRecursiveIntersections) and the counting numbers (Function calcCountingNumbers) also provided. The algorithm takes as input a set of clusters and a set of sepsets which together describe the input cluster graph. As mentioned, the set of sepsets may be empty indicating that the input graph is just a set of clusters. The algorithm will then update the set of sepsets $\boldsymbol{\mathcal{S}}$ by removing and adding sepsets as necessary for the resultant cluster graph to satisfy the TRIP. For every pair of conditional messages $\mu_{a,b}$ and $\mu_{b,a}$ the algorithm produces an accompanying set of conditioning sets $\boldsymbol{\mathcal{U}}_{a,b} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots\}$ and their counting numbers $\boldsymbol{u}_{a,b} = \{u_1, u_2, \ldots\}$ which are contained within the tuples $\mathbf{U}$ and $\mathbf{u}$ respectively. Edges are added in order of decreasing size as seen in line 5, which is an arbitrary order chosen to reduce the number of unnecessary edges from being added. Line 6 can be implemented with a graph search similar to Dijkstra's algorithm, where the edge weights are

replaced with sepsets and instead of keeping track of the path length it keeps track of the current running intersection.

Algorithm 1: Constructing Conditional Cluster Graph

input : Set of clusters $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1, \cdots, \mathcal{C}_I\}$, Set of sepsets $\boldsymbol{\mathcal{S}} = \{\mathcal{S}_1, \cdots, \mathcal{S}_J\}$
output : Updated sepsets $\boldsymbol{\mathcal{S}}$, Tuple of conditioning sets $\mathbf{U} = (\boldsymbol{\mathcal{U}}_1, \cdots, \boldsymbol{\mathcal{U}}_K)$ and their corresponding counting numbers $\mathbf{u} = (\boldsymbol{u}_1, \cdots, \boldsymbol{u}_K)$

```
// Tuple containing all non-zero cluster intersections
```
1 $\mathbf{B} := (\boldsymbol{\mathcal{B}}_{i,j} = \mathcal{C}_i \cap \mathcal{C}_j \neq \emptyset : i < j; i, j \in \mathcal{I}_{\mathcal{C}})$

```
// Remove any sepset that does not satisfy S_{i,j} = C_i ∩ C_j
```
2 **for** *each sepset* $\mathcal{S}_{i,j}$ *in* $\boldsymbol{\mathcal{S}}$ **do**
3 **if** $\mathcal{S}_{i,j} \neq \mathcal{B}_{i,j}$ **then**
4 $\boldsymbol{\mathcal{S}}$.remove($\mathcal{S}_{i,j}$)

```
// Iteratively add sepsets to the graph
```
5 **for** *each intersection* $\mathcal{B}_{i,j}$ *in* $\mathbf{B}$ *in order of decreasing size* **do**
6 $\boldsymbol{\mathcal{R}}_{i,j}$:= set of running intersections between clusters C_i and C_j that are not subsets of other running intersections
7 **if** $\boldsymbol{\mathcal{R}}_{i,j}$ *does not contain* $\mathcal{B}_{i,j}$ **then**
8 $\boldsymbol{\mathcal{S}}$.insert($\mathcal{B}_{i,j}$)
9 **if** $\boldsymbol{\mathcal{R}}_{i,j}$ *is not empty* **then**
10 $\boldsymbol{\mathcal{U}}_{i,j}$ = calcRecursiveIntersections($\boldsymbol{\mathcal{R}}_{i,j}$)
11 $\boldsymbol{u}_{i,j}$ = calcCountingNumbers($\boldsymbol{\mathcal{U}}_{i,j}$)
12 $\mathbf{U}$.insert($\boldsymbol{\mathcal{U}}_{i,j}$)
13 $\mathbf{u}$.insert($\boldsymbol{u}_{i,j}$)
14 **return** $\boldsymbol{\mathcal{S}}, \mathbf{U}, \mathbf{u}$

As mentioned before in Section 3.4.2, edges are never *expanded* as was done in Section 3.4.1, they are only *added*. There are cases when expanding edges can lead to suboptimal graphs and, if done incorrectly, can result in the graph violating the MRIP. The process of adding edges do not have these complications. That is why in line 4 of Algorithm 1 the edges that need to be expanded are removed before the process of adding edges starts.

Algorithm 1 is only one possible algorithm for determining the conditioning factors. It always returns a cluster graph that satisfies the TRIP, which does not need to be the case. Not all sepsets need to have the full intersection. The algorithm can be altered to construct graphs that satisfy the MRIP or the RIP. When adding an edge, if you do not want that edge to be conditional, simply exclude the variables from the sepset that are part of the conditioning factor $\mathcal{S}_{a,b} = (\mathcal{C}_a \cap \mathcal{C}_b) \backslash \mathcal{Q}_{a,b}$. If this is performed for all sepsets, then the graph will satisfy the RIP. Note that this should be performed during the iterative construction of the graph and not after the fact.

In conclusion, Algorithm 1 combines everything discussed in this section to automatically determine everything we need to calculate the conditioning factors and therefore perform conditional message passing.

Function calcRecursiveIntersections($\boldsymbol{\mathcal{R}}$)

```
  // Set of conditioning sets
1 𝒰 := {}
2 for each index i in the indexing set ℐ_ℛ do
3     for for each index j > i in the indexing set ℐ_ℛ do
4         𝒳 := ℛ_i ∩ ℛ_j
5         if 𝒳 not in 𝒰 and 𝒳 ≠ ∅ then
6             𝒰.insert(𝒳)
7 𝒳 := {}
8 if 𝒰 not empty then
9     𝒳 = calcRecursiveIntersections(𝒰)
10 for each set ℛ in ℛ do
11     if ℛ not in 𝒳 then
12         𝒳.insert(ℛ)
13 return 𝒳
```

Function calcCountingNumbers($\boldsymbol{\mathcal{U}}$)

```
  // Set of counting numbers
1 u := {u_i : i ∈ ℐ_𝒰}
2 for each index i in the indexing set ℐ_𝒰 in order of decreasing set size 𝒰_i do
3     u_i = 1
4     for each index j in the indexing set ℐ_𝒰 do
5         if 𝒰_i ⊂ 𝒰_j then
6             u_i = u_i − u_j
7 return u
```

3.5. Loopy Belief Update

In Section 2.4.3 the message passing variant known as loopy belief propagation was introduced and was used in Section 3.3 to define a conditional message. In this section we return to standard message passing on cluster graphs to introduce another existing variant of message passing referred to as the Lauritzen-Spiegelhalter algorithm [12], sum-product-divide or as loopy belief update [2, p 364]. When a cluster graph is densely connected, loopy belief update has some computational advantages over loopy belief propagation. As before, after introducing loopy belief update it will be applied to conditional message passing.

3.5.1. Standard Loopy Belief Update

In loopy belief update, a message is defined in terms of the cluster belief Ψ as opposed to the cluster factor ϕ,

$$\mu_{a,b}(\mathcal{S}_{a,b}) = \sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \frac{\Psi_a(\mathcal{C}_a)}{\mu_{b,a}(\mathcal{S}_{a,b})}. \tag{3.16}$$

The two definitions are equivalent as shown below, where we substitute the definition of the cluster belief (Equation 2.20) into Equation 3.16. The resultant message equation is equivalent to that of loopy belief propagation (Equation 2.19):

$$\begin{aligned}\mu_{a,b}(\mathcal{S}_{a,b}) &= \sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \frac{\phi_a(\mathcal{C}_a) \prod_{S_{i,a} \in \boldsymbol{S}} \mu_{i,a}(\mathcal{S}_{i,a})}{\mu_{b,a}(\mathcal{S}_{a,b})} \\ &= \sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \phi_a(\mathcal{C}_a) \prod_{S_{i,a} \in \boldsymbol{S}, i \neq b} \mu_{i,a}(\mathcal{S}_{i,a}).\end{aligned}$$

Defining the message in terms of the cluster belief allows for the reuse of previous calculations. Instead of recomputing the cluster belief each time we want to send a message, which can involve multiple message multiplications, it is stored for when it is needed. In densely connected graphs this can result in a large reduction in computation time compared to belief propagation. However, when a cluster receives a new message, its cluster belief needs to be recalculated. Since only one message changed, we can efficiently calculate the new cluster belief by using the old one. We can therefore *update* the cluster belief as follows:

$$\Psi_b^{\text{new}}(\mathcal{C}_b) = \Psi_b^{\text{old}} \frac{\mu_{a,b}^{\text{new}}(\mathcal{S}_{a,b})}{\mu_{a,b}^{\text{old}}(\mathcal{S}_{a,b})}, \tag{3.17}$$

where we divide out the old message and multiply in the new one. This way we avoid recalculating the cluster belief form scratch. It is possible to do even better. We can write the update Equation 3.17 in terms of a distribution known as the *sepset belief*, denoted ψ, which is defined as the

product of the two opposing messages between two clusters, or

$$\psi_{a,b}(\mathcal{S}_{a,b}) = \mu_{a,b}(\mathcal{S}_{a,b})\mu_{b,a}(\mathcal{S}_{a,b}). \tag{3.18}$$

A new update equation is obtained when multiplying the numerator and denominator of Equation 3.17 by the message going in the opposite direction as follows:

$$\begin{aligned}\Psi_b^{\text{new}}(\mathcal{C}_b) &= \Psi_b^{\text{old}} \frac{\mu_{a,b}^{\text{new}}(\mathcal{S}_{a,b})\mu_{b,a}^{\text{old}}(\mathcal{S}_{a,b})}{\mu_{a,b}^{\text{old}}(\mathcal{S}_{a,b})\mu_{b,a}^{\text{old}}(\mathcal{S}_{a,b})} \\ &= \Psi_b^{\text{old}} \frac{\psi_{a,b}^{\text{new}}(\mathcal{S}_{a,b})}{\psi_{a,b}^{\text{old}}(\mathcal{S}_{a,b})}.\end{aligned} \tag{3.19}$$

The cluster belief can therefore be updated by multiplying in the new sepset belief and dividing out the old one. This way we only need to store the sepset belief $\psi_{a,b}$ instead of the messages $\mu_{a,b}$ and $\mu_{b,a}$. It is also no longer necessary to calculate the messages, since the messages are sent implicitly when updating a cluster belief using Equation 3.19. Calculating the sepset belief is also faster than calculating a message because we can get rid of one factor division in Equation 3.16 as follows:

$$\psi_{a,b}(\mathcal{S}_{a,b}) = \mu_{b,a}(\mathcal{S}_{a,b})\mu_{a,b}(\mathcal{S}_{a,b}) \tag{3.20}$$

$$= \sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \frac{\Psi_a(\mathcal{C}_a)}{\mu_{b,a}(\mathcal{S}_{a,b})} \mu_{b,a}(\mathcal{S}_{a,b}), \tag{3.21}$$

$$= \sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \Psi_a(\mathcal{C}_a). \tag{3.22}$$

In conclusion, the cost of computing a message in densely connected graphs can be greatly reduced by using loopy belief update as opposed to loopy belief propagation. The improvement comes from belief update's ability to reuse previously computed messages to reduce the cost of sending a message.

3.5.2. Conditional Loopy Belief Update

In order to use the loopy belief update algorithm for conditional message passing, we have to rederive Equation 3.22. The definition of the cluster belief and sepset belief are the same for both conditional and standard message passing. Only the message equations differ. We can therefore substitute the definition of a conditional message (Equation 3.3), into the definition of the sepset

belief (Equation 3.18) resulting in the conditional sepset belief

$$\begin{aligned}\psi_{a,b}(\mathcal{S}_{a,b}) &= \frac{\sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \phi_a(\mathcal{C}_a) \prod_{S_{i,a} \in \boldsymbol{S}, i \neq b} \mu_{i,a}(\mathcal{S}_{a,b})}{\Theta_{a,b}(\mathcal{Q}_{a,b})} \mu_{b,a}(\mathcal{S}_{a,b}) \\ &= \frac{\sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \phi_a(\mathcal{C}_a) \prod_{S_{i,a} \in \boldsymbol{S}} \mu_{i,a}(\mathcal{S}_{a,b})}{\Theta_{a,b}(\mathcal{Q}_{a,b})} \\ &= \frac{\sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \Psi_a(\mathcal{C}_a)}{\Theta_{a,b}(\mathcal{Q}_{a,b})}.\end{aligned} \tag{3.23}$$

Just like we divide a standard message by a conditioning factor to get a conditional message, we divide the standard sepset belief by a conditioning factor to get a conditional sepset belief. The conditional sepset belief is then used to update cluster beliefs exactly as with standard loopy belief update (Equation 3.19).

3.6. Efficient Calculation of the Conditioning Factor

The conditioning factor Θ can consist of multiple factors that need to be calculated and multiplied together. Some of these factors are the marginal distributions of other factors, and all of them are the marginal distributions of the sepset belief. We can therefore reduce the computation time by using previously computed marginal distributions, instead of using the cluster belief each time.

Consider again the calculation of a conditioning factor

$$\Theta_{a,b}(\mathcal{Q}_{a,b}) = \prod_{i \in I_{\boldsymbol{\mathcal{U}}_{a,b}}} \left(\sum_{\mathcal{C}_a \setminus \mathcal{U}_i} \Psi_a(\mathcal{C}_a) \right)^{u_i},$$

where all the sets $\mathcal{U}_i$ are subsets of $\mathcal{S}_{a,b}$. We can therefore first compute $\Psi_a(\mathcal{S}_{a,b}) = \sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \Psi_a(\mathcal{C}_a)$, which can then be reused to calculate Θ as

$$\Theta_{a,b}(\mathcal{Q}_{a,b}) = \prod_{i \in I_{\boldsymbol{\mathcal{U}}_{a,b}}} \left(\sum_{\mathcal{S}_{a,b} \setminus \mathcal{U}_i} \Psi_a(\mathcal{S}_{a,b}) \right)^{u_i}, \tag{3.24}$$

leading to computational savings since $\Psi(\mathcal{S}_{a,b})$ has exponentially fewer states than $\Psi(\mathcal{C}_a)$. In loopy belief update, $\Psi_a(\mathcal{S}_{a,b})$ is already computed in the numerator of the conditional sepset. For loopy belief propagation, we can obtain it by simply multiplying the numerator of the conditional message by $\mu_{b,a}$ as seen in Equation 3.23.

This principle of reusing a previously computed marginal distribution to calculate the marginal distribution of some subset can be extended further. Since it is possible for some sets in $\boldsymbol{\mathcal{U}}$ to be

subsets of other sets in $\boldsymbol{\mathcal{U}}$, we can reuse previously computed factors as follows:

$$\Psi(\mathcal{U}_i) = \sum_{\mathcal{U}_j \setminus \mathcal{U}_i} \Psi(\mathcal{U}_j), \qquad \mathcal{U}_i \subset \mathcal{U}_j. \tag{3.25}$$

A directed tree, similar to a region graph, can be used to arrange the conditioning sets, where the sets in $\boldsymbol{\mathcal{U}}$ are connected such that an edge leads from a superset to a subset with $\mathcal{S}_{a,b}$ as the root of the tree. We can then efficiently compute all the required factors by starting at the root of the tree and moving to the leaves, using the factor of a parent node to compute the factor of the child node.

3.7. Conditional Message Passing at Convergence

When standard message passing converges, all connected clusters agree on the marginal distribution over the variables that are in the sepset connecting the two clusters [2, p 358],

$$\sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \Psi_a(\mathcal{C}_a) = \sum_{\mathcal{C}_b \setminus \mathcal{S}_{a,b}} \Psi_b(\mathcal{C}_b), \qquad \forall S_{a,b} \in \boldsymbol{S}. \tag{3.26}$$

Every inference method discussed in Section 2.4 shares a similar property. For example, when deriving the cluster variation method it manifests in the form of compatibility constraints (see Equation 2.43). Until now, it has been informally suggested that this is also true for conditional message passing due to its similarities with standard message passing. This section provides a proof that shows that it is indeed the case that when conditional message passing converges the cluster beliefs satisfy Equation 3.26.

The proof starts by first showing that Equation 3.26 holds at convergence for standard message passing. To reiterate, we want to show that if the messages in a graph do not change from one iteration of message passing to the next ($\mu^{\text{new}} = \mu^{\text{old}}$), the condition above holds. This is done by rewriting Equation 3.26 as follows:

$$\sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \frac{\Psi_a(\mathcal{C}_a)}{\mu^{\text{old}}_{b,a}(\mathcal{S}_{a,b})} \mu^{\text{old}}_{b,a}(\mathcal{S}_{a,b}) = \sum_{\mathcal{C}_b \setminus \mathcal{S}_{a,b}} \frac{\Psi_b(\mathcal{C}_b)}{\mu^{\text{old}}_{a,b}(\mathcal{S}_{a,b})} \mu^{\text{old}}_{a,b}(\mathcal{S}_{a,b}), \tag{3.27}$$

where the superscript ‘old’ indicates that the message was computed in the previous iteration of message passing. We then substitute in the definition of a message as given by Equation 3.16 to get

$$\mu^{\text{new}}_{a,b}(\mathcal{S}_{a,b}) \mu^{\text{old}}_{b,a}(\mathcal{S}_{a,b}) = \mu^{\text{new}}_{b,a}(\mathcal{S}_{a,b}) \mu^{\text{old}}_{a,b}(\mathcal{S}_{a,b}). \tag{3.28}$$

If during message passing the messages do not change (message passing has converged), then we know that

$$\mu_{b,a}^{\text{new}}(\mathcal{S}_{a,b}) = \mu_{b,a}^{\text{old}}(\mathcal{S}_{a,b}),$$
$$\mu_{b,a}^{\text{new}}(\mathcal{S}_{a,b}) = \mu_{b,a}^{\text{old}}(\mathcal{S}_{a,b}),$$

and therefore Equations 3.28 and 3.26 are satisfied.

We follow a similar process for conditional message passing. The equation describing the calculation of the conditional sepset belief (Equation 3.23) is rewritten as a message equation by substituting it into the definition of the sepset belief (Equation 3.18) as follows:

$$\begin{aligned}\psi_{a,b}(\mathcal{S}_{a,b}) &= \mu_{a,b}(\mathcal{S}_{a,b})\mu_{b,a}(\mathcal{S}_{a,b}) \qquad (3.29)\\ &= \frac{\sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \Psi_a(\mathcal{C}_a)}{\Theta_{a,b}(\mathcal{Q}_{a,b})},\\ \mu_{a,b}(\mathcal{S}_{a,b}) &= \frac{\sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \Psi_a(\mathcal{C}_a)}{\Theta_{a,b}(\mathcal{Q}_{a,b})\mu_{b,a}(\mathcal{S}_{a,b})}. \qquad (3.30)\end{aligned}$$

Substituting Equation 3.29 into Equation 3.27 results in

$$\Theta_{a,b}^{\text{new}}(\mathcal{Q}_{a,b})\mu_{a,b}^{\text{new}}(\mathcal{S}_{a,b})\mu_{b,a}^{\text{old}}(\mathcal{S}_{a,b}) = \Theta_{b,a}^{\text{new}}(\mathcal{Q}_{a,b})\mu_{b,a}^{\text{new}}(\mathcal{S}_{a,b})\mu_{a,b}^{\text{old}}(\mathcal{S}_{a,b}). \qquad (3.31)$$

For the above condition to be satisfied at convergence, $\Theta_{a,b}^{\text{new}}(\mathcal{Q}_{a,b})$ must equal $\Theta_{b,a}^{\text{new}}(\mathcal{Q}_{a,b})$ since $\mu_{a,b}^{\text{new}}(\mathcal{S}_{a,b})\mu_{b,a}^{\text{old}}(\mathcal{S}_{a,b})$ will equal $\mu_{b,a}^{\text{new}}(\mathcal{S}_{a,b})\mu_{a,b}^{\text{old}}(\mathcal{S}_{a,b})$. To prove that this is indeed what happens, consider a general cluster graph with clusters $\boldsymbol{C}$ and edges $\boldsymbol{S}$ that has exactly *one* conditional message labelled $\mu_{1,2}$ between clusters C_1 and C_2. The conditioning factors consist of marginal distributions constructed from the conditioning sets $\boldsymbol{\mathcal{U}}_{1,2}$, which are calculated from the running intersections between C_1 and C_2 (see Subsection 3.4.2). These running intersections, labelled $\boldsymbol{\mathcal{R}}_{1,2} = \{\mathcal{R}_{1,2}^1 \dots \mathcal{R}_{1,2}^N\}$, are calculated from paths $\boldsymbol{p}_{1,2} = \{p_{1,2}^1 \dots p_{1,2}^N\}$ that contain *no* conditional messages. Since we have just shown Equation 3.26 holds at convergence for standard messages, we know that at convergence each pair of clusters within the different paths satisfy

$$\sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \Psi_a(\mathcal{C}_a) = \sum_{\mathcal{C}_b \setminus \mathcal{S}_{a,b}} \Psi_b(\mathcal{C}_b), \qquad \forall\, S_{a,b} \in p_{1,2}^i,\ p_{1,2}^i \in \boldsymbol{p}_{1,2}. \qquad (3.32)$$

Note that if two distributions are the same, then their marginal distributions will also be the same. This allows us to rewrite Equation 3.32 as

$$\sum_{\mathcal{C}_a \setminus \mathcal{R}_{1,2}^i} \sum_{\mathcal{C}_a \setminus \mathcal{S}_{a,b}} \Psi_a(\mathcal{C}_a) = \sum_{\mathcal{C}_b \setminus \mathcal{R}_{1,2}^i} \sum_{\mathcal{C}_b \setminus \mathcal{S}_{a,b}} \Psi_b(\mathcal{C}_b), \qquad \forall\, S_{a,b} \in p_{1,2}^i,\ p_{1,2}^i \in \boldsymbol{p}_{1,2}. \qquad (3.33)$$

We also know that the running intersection of a path is a subset of every sepset in that path

$$\mathcal{R}^i_{1,2} \subseteq \mathcal{S}_{a,b} \qquad \forall\ S_{a,b} \in p^i_{1,2},\ p^i_{1,2} \in \boldsymbol{p}_{1,2}, \tag{3.34}$$

which allows us to simplify Equation 3.33 to be

$$\sum_{\mathcal{C}_a \setminus \mathcal{R}^i_{1,2}} \Psi_a(\mathcal{C}_a) = \sum_{\mathcal{C}_b \setminus \mathcal{R}^i_{1,2}} \Psi_b(\mathcal{C}_b), \qquad \forall\ S_{a,b} \in p^i_{1,2},\ p^i_{1,2} \in \boldsymbol{p}_{1,2}. \tag{3.35}$$

Since every cluster in a path between C_1 and C_2 agree on the marginal distribution over the running intersection of that path, it follows that C_1 and C_2 also agree. In fact, since the conditioning sets $\boldsymbol{\mathcal{U}}_{1,2}$ are determined from the intersections of the sets in $\boldsymbol{\mathcal{R}}_{1,2}$ and are therefore subsets of the running intersections, C_1 and C_2 agree on all the conditioning sets

$$\sum_{\mathcal{C}_1 \setminus \mathcal{U}^i_{1,2}} \Psi_1(\mathcal{C}_1) = \sum_{\mathcal{C}_2 \setminus \mathcal{U}^i_{1,2}} \Psi_2(\mathcal{C}_2), \qquad \forall\, \mathcal{U}^i_{1,2} \in \boldsymbol{\mathcal{U}}_{1,2}. \tag{3.36}$$

Finally, since the conditioning factors $\Theta^{\text{new}}_{1,2}(\mathcal{Q}_{1,2})$ and $\Theta^{\text{new}}_{2,1}(\mathcal{Q}_{1,2})$ are constructed from the above marginal distributions (see Equation 3.14), which are all the same for C_1 and C_2 at convergence, it follows that $\Theta^{\text{new}}_{1,2}(\mathcal{Q}_{1,2})$ equals $\Theta^{\text{new}}_{2,1}(\mathcal{Q}_{1,2})$ at convergence.

To summarize the proof for a graph containing a single conditional connection, the other paths in the graph, which do not contain conditional messages, allow for the two clusters to agree on the marginal distributions from which the conditioning factors are constructed. This results in Equation 3.31 being satisfied, since at convergence the new and old conditional messages stay the same.

This proof is extended to graphs with multiple conditional messages by repeating the steps above, each time adding one additional conditional message. At each subsequent iteration we know that every other cluster pair in the graph (connected by standard or conditional messages) satisfies Equation 3.26. This iterative approach is exactly the same principle used by Algorithm 1 to determine the conditioning sets and counting numbers. Therefore, if the graph is constructed using Algorithm 1, then at convergence Equation 3.26 holds for every cluster pair in the graph. As a result, conditional messages constrain the cluster beliefs in exactly the same way as standard message passing does, but allows for additional constraints which should result in better approximations.

3.8. Summary

This chapter defined new properties, namely the MRIP and the TRIP, that a cluster graph must satisfy in order for message passing to produce good approximate marginals. They were motivated by the fact that they allow for more information to be exchanged between clusters. However, these larger sepsets allow for severe feedback loops to exist where the same random variables are present throughout the loop. To limit the influence of these specific type of loops, the novel conditional message-passing algorithm was developed. Both belief propagation and belief update variants were presented along with an algorithm for automatically determining all the components required for conditional message passing. It is difficult to analytically predict the performance of message passing algorithms due to their dynamic nature. In the next chapter, extensive tests will be performed in order to determine whether conditional message passing does indeed improve upon standard message passing on cluster graphs.

Chapter 4

Experimental Investigation

This chapter compares the newly defined conditional message passing against standard message passing for cluster graphs, as well as against the parent-to-child algorithm for region graphs. The approximate marginal distributions produced by each method is compared against the true marginal distributions using randomized tests, where the algorithms are applied to randomly generated factorized joint probability distributions. The methods are also applied on the Ising model, which is often used as a benchmark when testing inference algorithms [2, 5].

The following section will discuss the general framework used for generating the randomized tests. In subsequent sections the specific tests will be described and their results provided.

4.1. Methodology

In order to thoroughly compare the different inference methods, they need to be applied on a wide variety of joint distributions. This is done by generating a wide variety of factorized joint probability distributions and using them to compare the different message-passing algorithms. This section describes the method used for generating the factorized distributions.

4.1.1. Generating Factor Scopes

The first step of generating the factorized probability distribution is to generate its factor scopes. The factor scopes are then used to create the factor distributions as well as the cluster graphs and region graphs.

For a given set of variables $\mathcal{X}$, number of factors N and number of variables per factor k_n, a random set of factor scopes is generated by uniformly sampling random variables from $\mathcal{X}$ to form part of a factor scope. The variables for a specific factor scope are sampled from $\mathcal{X}$ without replacement, but every factor is sampled from the same original $\mathcal{X}$ so that they may share random variables. The dimensionality of a set or vector will be indicated as $\mathcal{X}^{(k)}$, where k is the number of elements in the set $\mathcal{X}$. After choosing the factor scope sizes and sampling the random variables we have

$$\mathcal{X} = \bigcup_{n}^{N} \mathcal{X}_n^{(k_n)}. \tag{4.1}$$

The individual size of each factor scope k_i can also be randomized. For our tests, the number of factors N is kept constant, and the factor scope sizes k_n are sampled from a uniform distribution. Since the number of factors N is fixed, the number of variables in $\mathcal{X}$ will affect how many dependencies exist between the random variables. Fewer random variables will, on average, result in more factors sharing variables, leading to more densely connected graphs. Varying the number of random variables in $\mathcal{X}$, along with the random factor scope sizes, results in a large variety of different factorizations.

When comparing the different methods, it would be helpful to have a way of quantifying how difficult the set of factor scopes that have been generated are to handle. Therefore, the *cluster variation depth* is introduced as an indication of how difficult a set of factor scopes are. The cluster variation depth is equal to how many times the function calcRecursiveIntersections recursively calls itself when given the factor scopes as input. It is therefore the number of times the intersections between the factor scopes can be taken.

4.1.2. Generating Factor Distributions from Factor Scopes

After generating a set of N factor scopes $\{\mathcal{X}_i : i = 1, 2 \dots N\}$, the factor distributions are generated by sampling the required factor parameters from some probability distribution. It is useful to approach the problem of generating the parameters from the Bayesian point of view. In Bayesian statistics the parameters can also be random variables, described by their own probability distributions that in turn have their own parameters, also known as *hyperparameters* [2, p 733].

Consider the conditional probability distribution $P(\mathcal{X}|\mathcal{Y})$. The random variables to the right of the conditioning bar act like parameters, since different states result in different probability distributions over the variables on the left of the conditioning bar. Following the Bayesian interpretation, there is no differentiation between random variables and parameters, where both are interpreted as random variables. For example, let the parameters of a distribution be the set of random variables $\boldsymbol{\theta}$. Since there is no differentiation between random variables and parameters, a distribution with known parameters, represented by $\boldsymbol{t}$, can be written as $P(\mathcal{X}|\boldsymbol{\theta} = \boldsymbol{t})$. The joint distribution over $\mathcal{X}$ and $\boldsymbol{\theta}$ is $P(\mathcal{X}, \boldsymbol{\theta}|\boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ is the set of hyperparameters. This joint distribution can be viewed as a distribution over all possible factors with scope $\mathcal{X}$. Using the definition of conditional probability, and the fact that $\mathcal{X}$ is independent of $\boldsymbol{\alpha}$ given $\boldsymbol{\theta}$, the joint distribution is written as

$$P(\mathcal{X}, \boldsymbol{\theta}|\boldsymbol{\alpha}) = P(\mathcal{X}|\boldsymbol{\theta})P(\boldsymbol{\theta}|\boldsymbol{\alpha}). \tag{4.2}$$

For a given choice of hyperparameters $\boldsymbol{a}$, a factor distribution $\phi(\mathcal{X})$ is 'sampled' from $P(\mathcal{X}, \boldsymbol{\theta}|\boldsymbol{\alpha} = \boldsymbol{a})$ by sampling its parameters as follows:

$$\boldsymbol{t} \sim P(\boldsymbol{\theta}|\boldsymbol{\alpha} = \boldsymbol{a}), \tag{4.3}$$

$$\phi(\mathcal{X}) = P(\mathcal{X}|\boldsymbol{\theta} = \boldsymbol{t}). \tag{4.4}$$

The distribution $P(\boldsymbol{\theta}|\boldsymbol{\alpha})$ will be chosen to be the *conjugate prior* of the factor distribution $\phi(\mathcal{X}_i)$ with unknown parameters. The prior distribution $P(\boldsymbol{\theta}|\boldsymbol{\alpha} = \boldsymbol{a})$ is conjugate to $P(\mathcal{X}|\boldsymbol{\theta})$ if the posterior distribution $P(\boldsymbol{\theta}|\mathcal{X} = \boldsymbol{x}, \boldsymbol{\alpha} = \boldsymbol{a})$, following Bayes' rule has the same functional form as the prior [2, p 739], which can be written as

$$\begin{aligned} P(\boldsymbol{\theta}|\mathcal{X} = \boldsymbol{x}, \boldsymbol{\alpha} = \boldsymbol{a}) &= \frac{1}{Z} P(\mathcal{X} = \boldsymbol{x}|\boldsymbol{\theta}) P(\boldsymbol{\theta}|\boldsymbol{\alpha} = \boldsymbol{a}) \\ &= P(\boldsymbol{\theta}|\boldsymbol{\alpha} = \boldsymbol{a}'). \end{aligned} \tag{4.5}$$

The choice of prior distribution for our intents and purposes is largely arbitrary. Any method of generating random factor parameters would suffice in adding randomness to the tests. However, by using *conjugate* priors, existing distributions can be used that are known and have readily available implementations for sampling the necessary factor parameters. These distributions and their hyperparameters have been extensively used and studied, which means there is an existing knowledge base from which to reason about the factor distributions that are generated for different values of the hyperparameters. The specific conjugate priors will be discussed in their respective subsections.

This framework from Bayesian statistics is used for tests presented in this chapter. The set of factors Φ of the factorized joint distribution P_Φ is given by

$$\Phi = \{\phi_i(\mathcal{X}_i|\boldsymbol{\theta}_i = \boldsymbol{t_i}), \boldsymbol{t_i} \sim P(\boldsymbol{\theta}_i|\boldsymbol{\alpha}_i = \boldsymbol{a}_i), i = 1, 2, \ldots, N\}. \tag{4.6}$$

The hyperparameters give us control over the factors that are generated, which is useful during testing. The specific choice of hyperparameters will be discussed in the respective results sections for the discrete and continuous tests.

4.1.3. Comparing Approximate Marginal Distributions

After the factors of the factorized distribution $P_\Phi(\mathcal{X})$ have been determined, equivalent graphs are generated for the different message-passing schemes. The loopy belief update message passing variant for both standard and conditional message passing on cluster graphs will be used. Belief update is more efficient than belief propagation on densely connected graphs, which is the type of graph being investigated. For standard loopy belief update (LBU), the LTRIP algorithm is used to automatically generate a cluster graph that satisfies the RIP [13]. Algorithm 1 is used to construct a graph that satisfies the TRIP and calculates the necessary sets for use in conditional loopy belief update (CLBU). The cluster variation method is used to generate the region graph for the parent-to-child message-passing scheme (PC) [5].

Message passing is performed until the algorithm either converges or some maximum number of messages have been sent. A graph is said to have converged when the cluster beliefs of two connected clusters are the same for the variables in the sepset that connect them (see Subsection 2.4.3 on approximate inference). When this state is reached, a message will not change from one

iteration to the next, since the message equations are all already satisfied and a stationery point has been reached. To determine if the message-passing algorithm has converged, the degree to which a message changes from one iteration to the next is measured by using the Kullback-Leibler divergence as follows:

$$\Delta_{\mathrm{KL}} = D(\mu_{a,b}^{\mathrm{new}} || \mu_{a,b}^{\mathrm{old}}). \tag{4.7}$$

If the change measured is below a certain threshold for every message in the graph, it is assumed that convergence has been reached and messages passing is stopped. It is necessary to use a threshold greater than zero, since in practice μ^{new} will never exactly equal μ^{old}. Sometimes, message passing will oscillate indefinitely. For these non-convergent tests, message passing is stopped after some threshold number of messages have been sent and use the cluster beliefs at that instant. This is done with the assumption that the system is oscillating around the stationary point. Stopping message passing might therefore provide cluster beliefs that are in the vicinity of the stationary point. However, the marginal distributions produced from non-convergent message passing are worse compared to message passing that converges since the beliefs produced do not satisfy the constraints imposed by the message equations. For this reason a differentiation will be made between convergent and non-convergent cases.

The true marginal distributions are calculated by standard message passing on an equivalent junction tree. For each method, the Kullback-Leibler divergence between the approximate and true marginal distribution is calculated for every factor scope. For cluster graphs, this results in the cluster beliefs being compared to the true marginal distributions. When comparing region graphs, the top level region beliefs (regions that have no parents) are compared to their true marginal distributions. This is because the top level regions of our region graphs and the clusters in our cluster graphs are the same. Each value is then added together to form a cumulative Kullback-Leibler divergence as

$$d = \sum_{i \in \mathcal{I}_\Phi} D\left(\Psi_i(\mathcal{X}_i) || \sum_{\mathcal{X} \setminus \mathcal{X}_i} P_\Phi(\mathcal{X}) \right). \tag{4.8}$$

The reason for calculating a cumulative Kullback-Leibler divergence instead of comparing the approximation produced by a single cluster or region, is because one part of a graph might converge to a good approximation while another part of the same graph might not. In order to get an overall error estimate, it is therefore necessary to compare multiple parts of the same graph. Furthermore, the reason for choosing the factor scopes as the scopes of the marginal distributions to be compared is because they are the largest marginal distributions estimated by the different message passing algorithms. Therefore, if a cluster or region belief is a good approximate marginal distribution, then all approximate marginal distributions with smaller scopes, including the univariate marginals, will also be good approximations. The inverse of this is not true. Even if all the approximate univariate cluster beliefs are close to the true univariate marginal distributions, it is not necessarily the case that the joint cluster belief is also close to the

true joint marginal distribution.

4.1.4. Message Scheduling

The order with which messages are passed within a graph can greatly affect how fast a message-passing algorithm converges, or even if it converges [2, p 408]. For this reason a method of scheduling messages known as residual belief propagation is used [14]. It is extremely flexible and simple to implement for LBU, CLBU and PC. Residual error propagation avoids sending unnecessary messages, and focuses on the parts of the network for which the constraints imposed by the message equations are the furthest from being satisfied. This can result in faster convergence, or even allow a message-passing algorithm to converge where it otherwise would not [14].

In residual error propagation, the messages are scheduled according to the amount by which a message will change when it is updated. The Kullback-Leibler divergence Δ_{KL} of Equation 4.7 is used as an estimate of this change. When a message is sent, it requires other messages to be recalculated and sent in turn. These messages are added to a queue, which is sorted from high to low priority. The priority of a message is determined from the Kullback-Leibler divergence Δ_{KL} of the messages it depends on. In every iteration, the top message in the queue is sent and removed from the queue. If the Kullback-Leibler divergence Δ_{KL} between the updated message and its previous self is above the threshold dictating convergence, the messages it influences are either added to the queue or their priorities are updated.

There is some special consideration when scheduling messages for PC and CLBU. For the parent-to-child algorithm it is important that when sending a message $\mu_{p,c}$, the messages in the denominator of Equation 2.34 are up-to-date and therefore not in the queue when calculating $\mu_{p,c}$. If they are in the queue, they need to be sent first. This requirement holds for every message.

Standard residual error propagation works well for CLBU; however, certain cluster graphs can have improved convergence if an adjustment, similar to that of PC, is made to message scheduling. When sending a conditional message, the conditioning factor divides out information that can propagate through other paths. Before sending the conditional message, the messages along these paths are sent first, starting at the cluster from which the conditional message is sent and ending at the receiving cluster. These paths are determined during the graph search where the running intersections are calculated.

4.1.5. Implementation

CLBU and PC were efficiently implemented in C++ using the resources provided by the PGM library known as EMDW, which also provided existing routines for LBU. Both CLBU and PC were implemented using the same classes provided by EMDW for constructing message passing algorithms, such as classes responsible for scheduling the messages and performing factor operations. Both conditional cluster graphs and region graphs were implemented such that

they use the same data structures for representing the graph and for storing message or cluster distributions.

The tests were performed on a 64bit Linux system (Pop! OS version 21.04) with an Intel Core i5-6200U CPU and 8 gigabytes of RAM. No multiprocessing or GPU enhancement was employed.

4.2. Discrete Experiments

In this section, the message-passing schemes are tested on discrete joint distributions. First the specific discrete distribution is introduced, namely the *categorical probability distribution*, as well as its conjugate prior. After the distributions have been introduced, the tests are described and their results presented.

4.2.1. The Categorical Distribution

A categorical distribution is a special case of the multinomial distribution and is sometimes referred to by that name. In a categorical distribution every discrete state $\mathcal{X} = \boldsymbol{x}_i$ has a corresponding probability θ_i associated with it [3]. This can be written as

$$P(\mathcal{X}|\boldsymbol{\theta}) = \prod_{i \in \mathcal{I}_{\boldsymbol{\theta}}} \theta_i^{[\mathcal{X}=\boldsymbol{x}_i]}, \tag{4.9}$$

$$\sum_{i \in \mathcal{I}_{\boldsymbol{\theta}}} \theta_i = 1, \tag{4.10}$$

where $\boldsymbol{\theta}$ is a set of parameters $\boldsymbol{\theta} = \{\theta_i : i \in \mathcal{I}_{\boldsymbol{\theta}}\}$ and $\mathcal{X} = \boldsymbol{x}$ is used to indicate a specific state of a set of random variables, similar to how $X = x$ is used to indicate a specific state of a single random variable. The distribution in Table 2.1 is an example of a categorical distribution. We can explicitly state its parameter values by writing it as $P(R, M|\boldsymbol{\theta} = \{\theta_1 = 0.28, \theta_2 = 0.52, \theta_3 = 0.04, \theta_4 = 0.16\})$.

4.2.2. The Dirichlet Distribution

Subsection 4.1.2 stated that factors are generated by sampling their parameters from a conjugate prior. The conjugate prior of the categorical distribution with unknown parameters is the *Dirichlet distribution* [3], which will be used for generating the parameters of our discrete factors. The Dirichlet distribution is given as

$$\text{Dir}(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{1}{Z(\boldsymbol{\alpha})} \prod_{i \in \mathcal{I}_{\boldsymbol{\theta}}} \theta_i^{\alpha_i - 1}, \tag{4.11}$$

$$\sum_{i \in \mathcal{I}_{\boldsymbol{\theta}}} \theta_i = 1. \tag{4.12}$$

For our purposes, all hyperparameters $\boldsymbol{\alpha}$ are set equal to one another. This will be indicated as $\boldsymbol{\alpha}_\circ = \{\alpha_i = \alpha_\circ : i \in \mathcal{I}_{\boldsymbol{\alpha}}\}$. The value chosen for $\alpha_\circ$ controls the *sparsity* of $\boldsymbol{\theta}$ sampled as shown in Figure 4.1. A probability distribution is said to be *sparse* if only a small portion of possible states are probable, while most states are assigned vanishingly small probabilities. Sparse factor distributions are generally more difficult for message passing algorithms to deal with. Sparser probabilities equate to stronger correlations between random variables. Clusters with sparse probability distributions which disagree over how probable a certain state is can prevent a message-passing algorithm from converging or create poor local optima. Such a collection of clusters are referred to as being *frustrated*. By decreasing $\boldsymbol{\alpha}_\circ$ the probability of generating a frustrated set of cluster distributions is increased.

x	y	z	P(x,y,z;α=10.0)	x	y	z	P(x,y,z;α=1.0)	x	y	z	P(x,y,z;α=0.5)	x	y	z	P(x,y,z;α=0.1)
0	0	0	1.11E-01	0	0	0	5.34E-02	0	0	0	5.34E-03	0	0	0	3.02E-08
0	0	1	1.47E-01	0	0	1	9.04E-02	0	0	1	3.40E-01	0	0	1	1.91E-01
0	1	0	1.53E-01	0	1	0	1.02E-01	0	1	0	1.92E-03	0	1	0	9.92E-14
0	1	1	8.69E-02	0	1	1	7.58E-02	0	1	1	3.28E-02	0	1	1	2.76E-02
1	0	0	9.83E-02	1	0	0	5.01E-02	1	0	0	2.75E-02	1	0	0	6.86E-02
1	0	1	1.23E-01	1	0	1	1.93E-01	1	0	1	1.45E-01	1	0	1	8.95E-06
1	1	0	1.35E-01	1	1	0	1.22E-01	1	1	0	3.58E-02	1	1	0	6.57E-01
1	1	1	1.45E-01	1	1	1	3.13E-01	1	1	1	4.12E-01	1	1	1	5.60E-02

Figure 4.1: Examples of categorical probability distributions created by sampling their parameters from a Dirichlet distribution for different α values. Sparse distributions are more likely for α values smaller than one and less likely for values greater than one. When α is equal to one the Dirichlet forms a uniform distribution over all possible $\boldsymbol{\theta}$. A darker element shade indicates a larger value, this helps to visualize the sparsity of the different distributions.

4.2.3. Results

For every test a set of random variables $\mathcal{X}$ is chosen of size k_{total}. Every discrete random variable in $\mathcal{X}$ is chosen to have m possible states. After choosing a value for the hyperparameter $\alpha_\circ$ and randomly generating the factor scopes, the Dirichlet distribution is used to generate N factors

$$\Phi = \{\phi_i(\mathcal{X}_i^{(k_i)}|\boldsymbol{\theta}_i^{(m^{k_i})} = \mathbf{t_i}) : \mathbf{t_i} \sim Dir(\boldsymbol{\alpha}^{(m^{k_i})} = \alpha_\circ), i = 1, 2, \ldots, N\}. \tag{4.13}$$

All tests were executed with the following choice of parameters:

- $N = 15$
- $m = 2$
- $\alpha_\circ \in [0.1, 0.4, 0.7, 1.0, 1.3]$
- $\mathcal{X} = \{X_i : i \leq k_{\text{total}}, i \in \mathbb{N}\}$

- $\mathcal{X}_i = \{X_j : X_j \in \mathcal{X}, j \leq k_i, j \in \mathbb{N}\}$
- $k_i \in [i : 3 \leq i \leq 10, i \in \mathbb{N}]$
- $k_{\text{total}} \in [18, 20, 30, 62, 70]$

These parameters were chosen to allow for a wide variety of joint distributions for which the exact marginal distributions can be calculated within a reasonable amount of time. The result of the randomly generated tests are shown in Figure 4.2, where the wide variety of joint distributions lead to a large variation in approximation errors. The figure indicates that CLBU and PC produce remarkably similar results, both of which have tests where the cumulative Kullback-Leibler divergence is less than that of any test performed by LBU. However, Figure 4.2 also shows that CLBU and PC converge for fewer tests than LBU, a problem that will be addressed in Section 4.5.

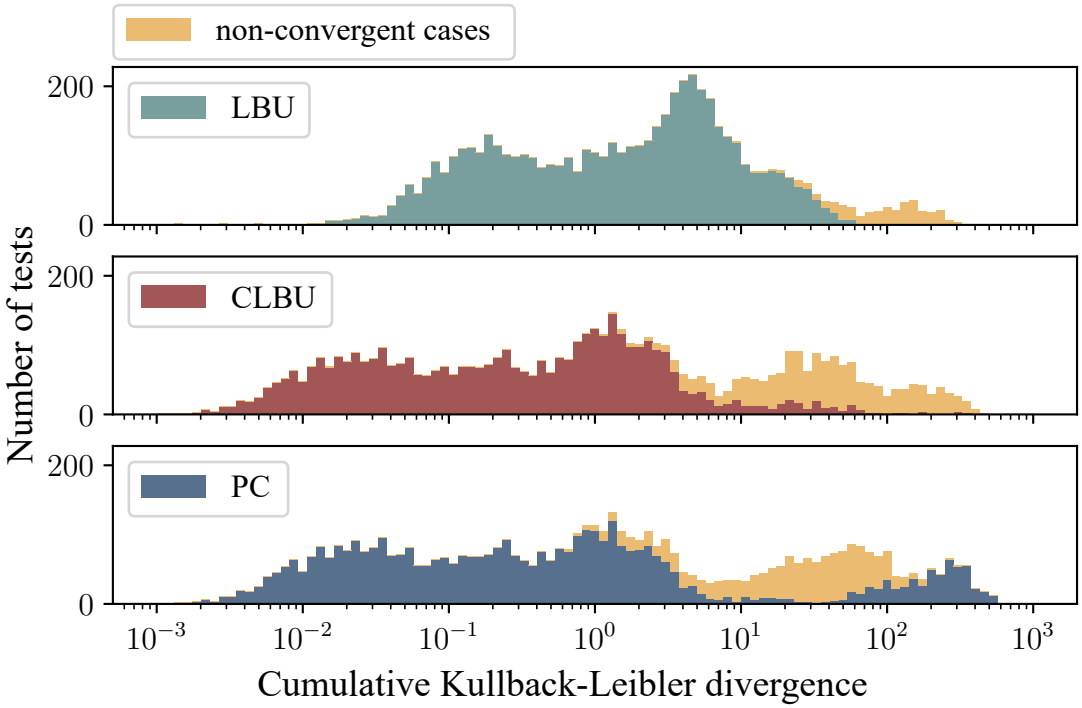


Figure 4.2: Comparison between the approximation error of the different message-passing schemes. The plot shows the difference between the approximated joint marginal distributions and the true marginal distributions is smaller for CLBU than for LBU. CLBU and PC produce a remarkably similar improvement over LBU for the cases where they converge.

Figure 4.3 shows how the cumulative Kullback-Leibler divergence changes when inference is performed using CLBU or PC compared to when inference is performed using LBU. Again we see that PC and CLBU produce similar results. For the majority of tests, CLBU and PC have better approximate marginals compared to LBU, with the peak in Figure 4.3 indicating that many

tests have a cumulative Kullback-Leibler divergence around 3 times smaller when performed using PC or CLBU compared to LBU.

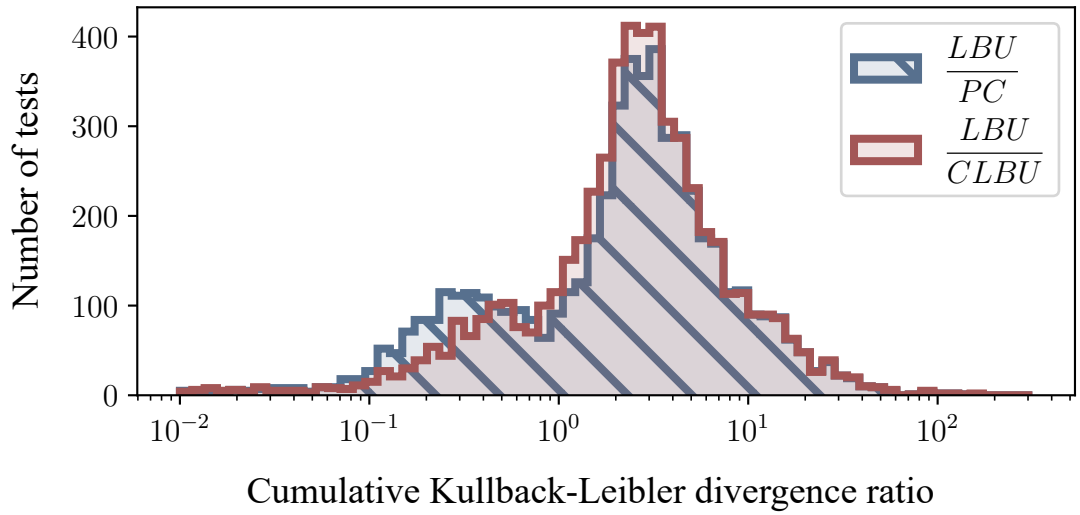


Figure 4.3: Direct comparison between the approximate marginals produced by the different message-passing algorithms for all tests, including non-convergent cases. A ratio greater than one (i.e. to the right of 10^0) indicates an improvement over LBU. This shows that CLBU and PC can produce a significant improvement over LBU, decreasing the Kullback-Leibler divergence ten-fold or more for certain tests.

The quality of approximation decreases as the value of $\alpha_\circ$ decreases as shown in Figure 4.4. As mentioned in Subsection 4.2.2, the factors are more frustrated when their parameters are sampled from a Dirichlet distribution with lower $\alpha_\circ$, which makes inference more difficult leading to an increase in the cumulative Kullback-Leibler divergence. Similarly, the Kullback-Leibler divergence increases as the cluster variation depth increases. A high cluster variation depth suggests that the scopes of the factors overlap with multiple other factors to create unique intersections. A higher cluster variation depth relates to a higher number of levels in a region graph, more conditional messages in a cluster graph satisfying the TRIP and more sepsets omitting a variable in a cluster graph satisfying the RIP. Note that in Figure 4.4 only the tests where every method converged are included. That is to say, only the tests where LBU, CLBU *and* PC converged. This is done to remove the obvious decrease in approximation quality which occurs when a message-passing algorithm does not converge, which unsurprisingly also correlates with $\alpha_\circ$ and the cluster variation depth as shown in figure 4.5.

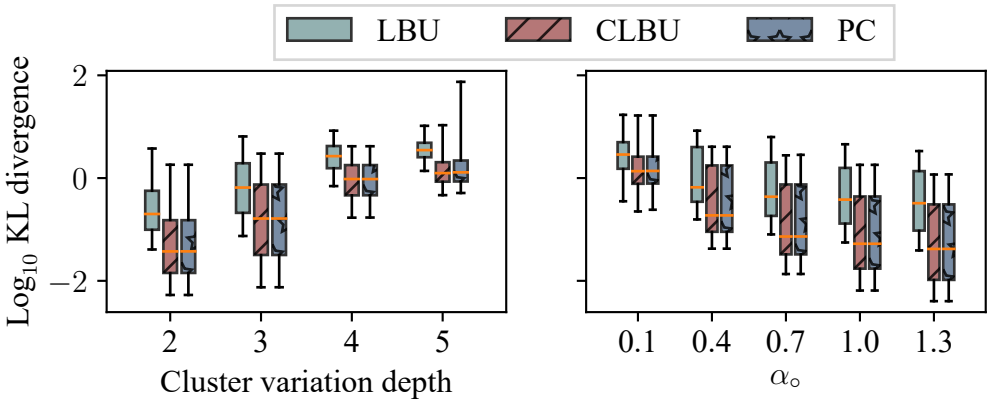


Figure 4.4: The 5%, 25%, 50%, 75% and 95% quantiles of the cumulative Kullback-Leibler divergence for the tests where LBU, CLBU and PC converged for every test. If PC and CLBU converge, they produce a similar improvement over LBU.

It is clear that CLBU and PC produce improved marginals over LBU if they converge. Unfortunately both CLBU and PC are less convergent than standard LBU as shown in Figure 4.5. Extremely densely connected graphs are especially problematic as shown by the sharp decrease in convergence as the cluster variation depth increases, where the cluster variation depth gives a rough indication of how densely connected the graph is. Luckily we have control over this density, since both cluster graphs and region graphs do not have to include all intersections dictated by the cluster variation method (see Section 4.5). For extremely difficult tests, PC has an increase in convergence as shown in Figure 4.5, however for many of those cases the approximation produced has a large KL divergence despite converging. Those cases can be seen in Figure 4.2, where there is an increase in convergent tests with high KL divergence.

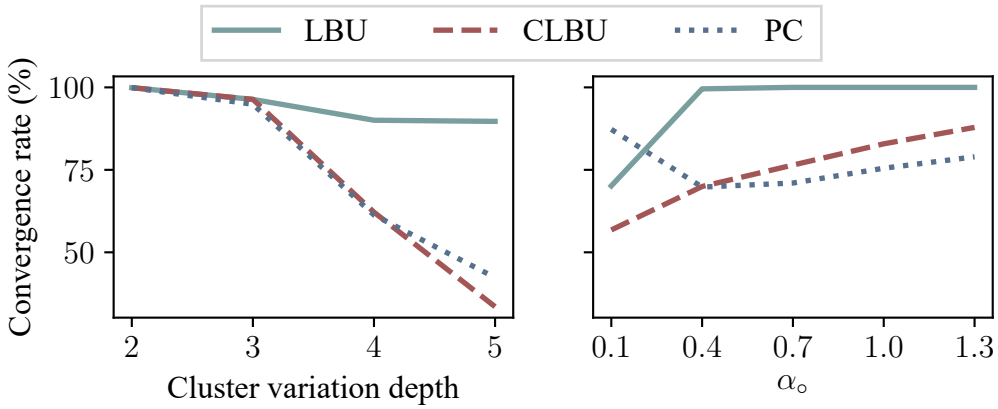


Figure 4.5: Plotting convergence rates against two properties that influence convergence. For the tests where the cluster variation depth is larger than 3, both CLBU and PC struggle to converge. For extremely difficult tests, such as those with $\alpha_\circ = 0.1$, PC has higher convergence, but results in an approximation with a high KL divergence.

The logarithm of the execution times are shown in Figure 4.6. As with Figure 4.4, the tests for which the methods do not converge are omitted since, theoretically, a non-convergent test has an infinite execution time. For graphs with a cluster variation depth of 3 and below, CLBU

has a faster execution time than LBU despite having messages that are more computationally expensive. Fast execution times for CLBU are more common for graphs that have lower cluster variation depth, which corresponds with a higher convergence rate as shown in Figure 4.5.

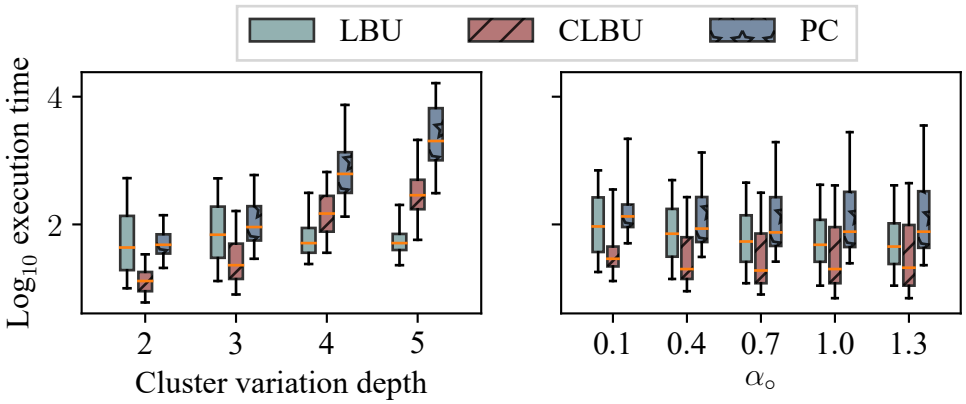


Figure 4.6: The 5%, 25%, 50%, 75% and 95% quantiles of the logarithm of the execution time in milliseconds for the tests where LBU, CLBU and PC converged.

In conclusion, of the 5600 tests executed, CLBU produced improved approximations over LBU for 78.75% of tests. For the 74.7% of tests that converged, CLBU produced improved approximations over LBU for 97% of tests. Similarly, PC improved upon LBU for 72% of the total tests, and 90.7% for the 76.5% of tests that converged.

When comparing the different execution times for the tests that converged, we see that CLBU averages an execution time that is 2 times faster than LBU and 4 times faster than PC. This is largely due the speed increase loopy belief update provides over loopy belief propagation.

4.3. Continuous Experiments

This section compares standard loopy belief update (LBU) and conditional loopy belief update (CLBU) where the joint probability distribution and its factors are Gaussian distributions. The parent-to-child method (PC) will not be considered due to difficulty with negative definite information matrices during message passing. For an explanation on the problem of negative definite information matrices see Subsection 4.3.3. These problems are exacerbated by the fact that the code base, used to implement the message-passing schemes, uses triangular matrices that cannot represent negative definite matrices. CLBU is also susceptible to negative definite information matrices that appear during message passing, but they are resolved by carefully selecting the order of operations. Doing this for the PC algorithm is more difficult than for CLBU. Since the main focus of this thesis is to improve message passing on cluster graphs, PC for Gaussian distributions will not be included.

4.3.1. The Gaussian Distribution

The Gaussian distribution, also known as the normal distribution, is a widely used continuous probability distribution. It has many desirable properties and is frequently observed in practice [3]. The geometric form of the Gaussian distribution is given as

$$\mathcal{N}(\mathbf{X}|\boldsymbol{\mu}, \mathbf{K}^{-1}) = \frac{1}{Z(\mathbf{K})} \exp\left(-\frac{1}{2}(\mathbf{X}-\boldsymbol{\mu})^{\mathrm{T}}\mathbf{K}(\mathbf{X}-\boldsymbol{\mu})\right), \tag{4.14}$$

where $\mathbf{X}$ is a D-dimentional vector of random variables, $\boldsymbol{\mu}$ is the D-dimentional mean vector, and $\mathbf{K}$ is the $D \times D$ information matrix. The information matrix is the inverse of the covariance matrix.

4.3.2. The Gaussian-Wishart Distribution

To generate Gaussian factors for the tests, the Gaussian parameters are treated as random variables. The conjugate prior for the Gaussian distribution, for when the mean and covariance variables are unknown, is the Gaussian-Wishart distribution [3] given by

$$P(\boldsymbol{\mu}, \mathbf{K}|\boldsymbol{\mu}_0, \mathbf{W}, \beta, n) = \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_0, \beta^{-1}\mathbf{K}^{-1})\mathcal{W}(\mathbf{K}|\mathbf{W}, n), \tag{4.15}$$

where $\mathcal{W}$ is the Wishart distribution

$$\mathcal{W}(\mathbf{K}|\mathbf{W}, n) = \frac{1}{Z(\mathbf{W}, n)}|\mathbf{K}|^{\frac{n-D-1}{2}} \exp\left(-\frac{1}{2}\mathrm{Tr}(\mathbf{WK})\right), \qquad n \geq D. \tag{4.16}$$

The function $\mathrm{Tr}(\mathbf{A})$ returns the trace of the square matrix $\mathbf{A}$ and is equal to the sum of its diagonal entries. The determinant of a matrix is indicated by $|\mathbf{A}|$. We can sample the parameters from the Gaussian-Wishart distribution by first sampling the information matrix from the Wishart distribution. Then the mean vector is sampled from a Gaussian distribution.

The hyperparameters $\boldsymbol{\mu}_0, \mathbf{W}, \beta$ and n can be used to alter the probability of generating a certain Gaussian factor. As in the discrete case, we would like to be able to have control over the strength of the dependencies between the random variables. In a Gaussian distribution, this is determined by the information matrix, specifically the *relative* values of the elements in this matrix. This is one of the strengths of the Gaussian distribution because the first and second order moments, the mean and variance, are parameters of the distribution. Generating information matrices from a Wishart distribution that has large off-diagonal entries, relative to the diagonal, will result in strong dependencies between the random variables.

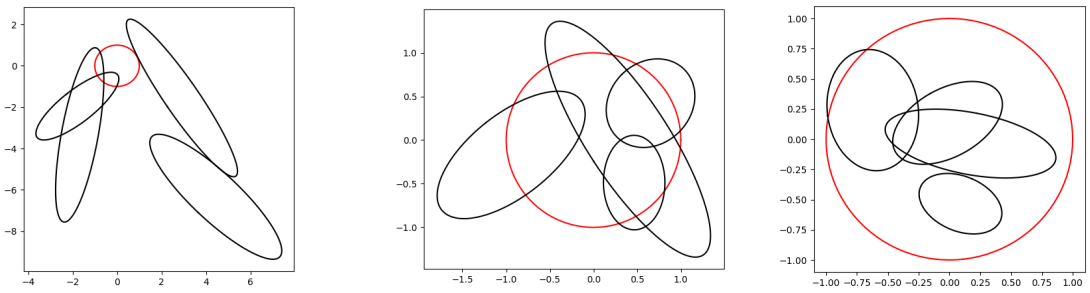
The Gaussian-Wishart distribution is more complicated than the Dirichlet distribution and so it is not obvious what parameters we want to vary. We can gain some insight into the Wishart

distribution by looking at the mean and variance of the elements in the matrix [15]

$$E[\mathbf{K}_{i,j}] = n\mathbf{W}_{i,j}, \tag{4.17}$$

$$\text{Var}[\mathbf{K}_{i,j}] = n(\mathbf{W}_{i,j}^2 + \mathbf{W}_{i,i}\mathbf{W}_{j,j}), \tag{4.18}$$

where $\mathbf{A}_{i,j}$ indicates the element at row i and column j of the matrix $\mathbf{A}$. The parameter $\mathbf{W}$ determines the average 'shape' of the resultant Gaussian distribution, which is determined by parameter $\mathbf{K}$, since n only scales the mean of the elements in $\mathbf{K}$. When considering the parameter n, we see that it scales the mean of $\mathbf{K}_{i,j}$ more than it scales the standard deviation of $\mathbf{K}_{i,j}$ from the mean. Therefore, larger n will result in the *relative* values of the elements in the sampled information matrix to be closer to the *relative* values of the elements in $\mathbf{W}$. Since the relative values determine the shape, changing n will determine how close the shape of our sampled Gaussian distributions are to that of $\mathcal{N}(\boldsymbol{\mu}, \mathbf{W})$. This is demonstrated in Figure 4.7.



(a) $n = 2$ **(b)** $n = 4$ **(c)** $n = 8$

Figure 4.7: Contour plots of two-dimensional multivariate Gaussian distributions with parameters sampled from a Gaussian-Wishart distribution for different values of n. The contours are one standard deviation away from the mean. The hyperparameter $\mathbf{W}$ is equal to the identity matrix $\mathbf{I}$, $\boldsymbol{\mu}_0 = [0, 0]^T$ and $\beta = 1$. Larger values of n result in a higher likelihood of the shape of the sampled distributions (black) to be close to the shape of $\mathcal{N}(\boldsymbol{\mu}_0, \mathbf{W})$ (red).

4.3.3. Negative Definite Information Matrices during Message Passing

Dividing one Gaussian distribution by another can create a distribution that is not a valid probability or factor distribution. Figure 4.8 shows an example of such a case, where the result has infinite area. This happens when the resulting Gaussian has a negative definite information matrix $\mathbf{K}$, resulting in an invalid distribution that cannot be normalized or marginalized since both operations involve calculating an area or volume under the distribution, which will be infinite.

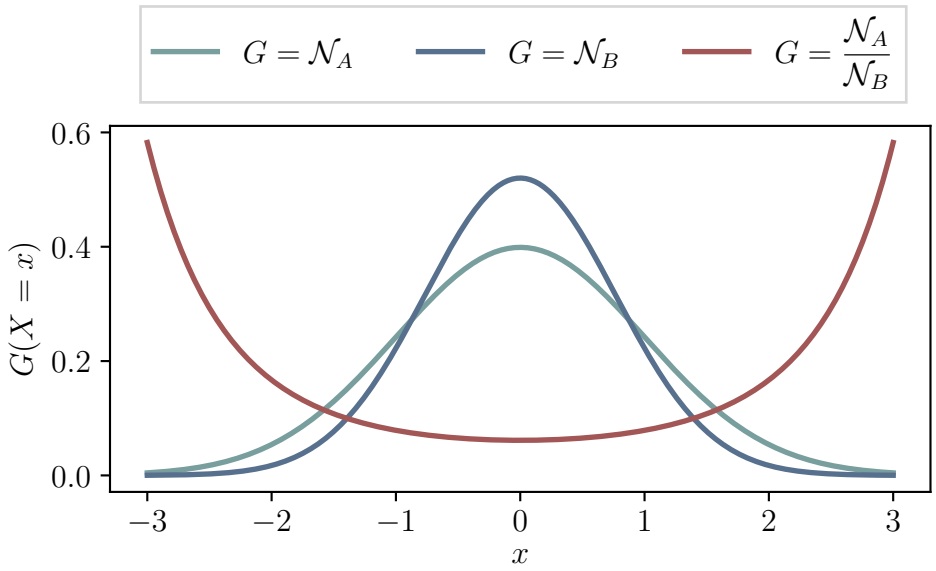


Figure 4.8: An example of an invalid Gaussian distribution with a negative definite information matrix that resulted from the division of two Gaussian distributions.

A message-passing algorithm that requires division of one distribution by another, such as LBU, CLBU and PC, can create invalid Gaussian distributions with negative definite matrices. This is fine as long as the marginalization operation is not performed on the invalid distribution. For example, a message that has a negative definite information matrix will be multiplied with other messages, which can result in a valid Gaussian distribution that is then marginalized without a problem.

The EMDW library used to implement the message-passing algorithms uses the triangular form of the information matrix. The triangular form is numerically stable and allows for certain matrix operations to be performed faster than traditional matrices but is *incapable* of representing negative definite matrices. Therefore, at *no* point during the process of passing a message may a negative definite information matrix occur. This is a problem for CLBU, where the conditional sepset belief itself is constructed from multiplying and *dividing* by distributions and will therefore often require a negative definite information matrix to represent. Therefore, a conditional cluster belief cannot always be explicitly calculated. To get around this, the operations necessary to update the cluster belief are performed in such a way that the conditional cluster belief is never explicitly calculated, similar to how a message is never explicitly calculated during belief update. The standard, unconditioned sepset belief can be stored, since it contains all the information required to implicitly construct the conditional cluster belief as shown in Section 3.6.

4.3.4. Results

As with the discrete case, a random set of factors is generated by first generating the factor scopes and then sampling the parameters. The parameters of the Gaussian factors are sampled as

$$\begin{aligned}\Phi = \{\phi_i(\mathcal{X}_i^{(k_i)}|\mathbf{u}_i^{(k_i)}, \mathbf{K}_i^{(k_i \times k_i)}) : \mathbf{K}_i &\sim \mathcal{W}(\mathbf{W}, n), \\ \mathbf{u}_i &\sim \mathcal{N}(\boldsymbol{\mu}_0, \beta\mathbf{K}_i^{-1}), \\ n &\geq k_i, \\ i &= 1, 2, \ldots, N\}.\end{aligned} \tag{4.19}$$

For the tests the hyperparameters β and $\boldsymbol{\mu}_0$ are chosen to be $\beta = 1$ and $\boldsymbol{\mu}_0 = [0, 0, \ldots, 0]^T$ since they only affect the mean of the random variables and not the dependencies between them. The random variables in a Gaussian distribution are independent of one another if the of-diagonal entries in the information matrix are all zero. Setting $\mathbf{W}$ equal to the identity matrix causes the random variables to be independent mean of the Gaussian-Wishart distributions. For low values of n, the sampled information matrix will deviate from the mean, resulting in strong dependencies between the random variables as seen in Figure 4.7a. As n increases, the likelihood of generating a factor with weak dependencies increases as shown in Figure 4.7c.

The specific choice for the parameters is similar to that of the discrete tests:

- $N = 15$
- $\boldsymbol{\mu}_0 = [0, \ldots, 0]^T, \beta = 1, \mathbf{W} = \mathbf{I}$.
- $n \in [8, 16, 32, 64]$
- $\mathcal{X} = \{X_i : X_i \leq k_{\text{total}}, i \in \mathbb{N}\}$
- $\mathcal{X}_i = \{X_j : X_j \in \mathcal{X}, j \leq k_i, j \in \mathbb{N}\}$
- $k_i \in [i : 3 \leq i \leq 10, i \in \mathbb{N}]$
- $k_{\text{total}} \in [18, 20, 30, 62, 70]$

The difference between the true marginal distribution and the approximation is measured using the cumulative Kullback-Leibler divergence (Equation 4.8) as well as two other metrics: the Euclidean distance between the true mean and the approximate mean as well as the distance between the true covariance and the approximate covariance. The latter is calculated by packing the matrices into vectors and then calculating the Euclidean distance between the two vectors. Since the covariance matrix is symmetric, only the top half is used in the calculation.

There were 47000 continuous tests executed with both CLBU and LBU having converged for virtually all of the tests. The results will be presented similarly to those of the discrete

tests in Section 4.2.3. Figure 4.9 shows that CLBU has many tests for which the cumulative Kullback-Leibler divergence is less than that of LBU.

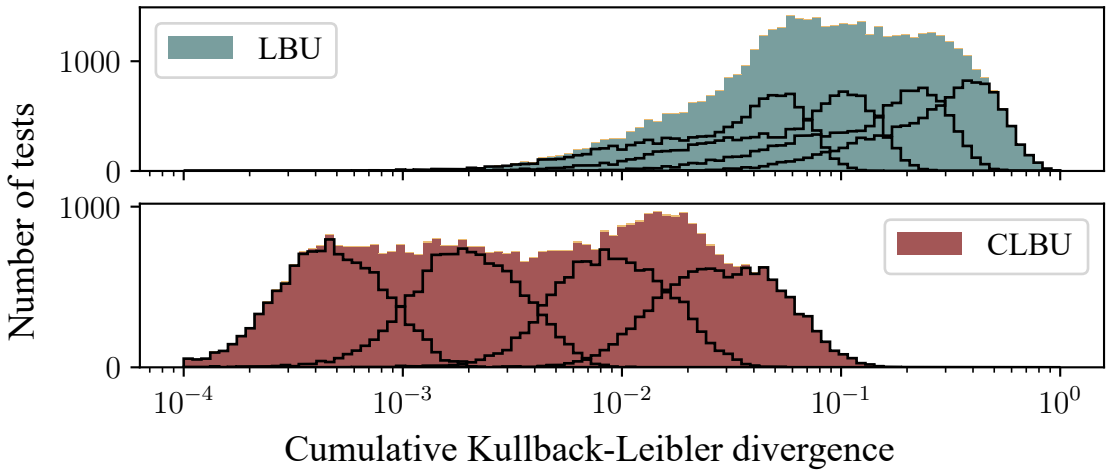


Figure 4.9: Comparison between the approximation error produced by LBU and CLBU as measured by the cumulative Kullback-Leibler divergence. The figure excludes the 1% of tests for which CLBU did not converge. The figure shows that CLBU significantly improves the approximation by having marginal distributions that are closer to the true marginal distributions. The black, Gaussian-shaped lines within the histograms show the performance for different values of n, where a higher n corresponds with a lower error.

Figure 4.10 shows the same results as Figure 4.9, but separates the approximation quality of the mean and the approximation quality of the covariance matrix. Again we see that there are tests for which the approximation error is much smaller when using CLBU compared to LBU. The distance to the true mean for both LBU and CLBU is much smaller than the distance to the true covariance due to a property that message passing with Gaussian distributions have, where, if they converge, they converge to the true means [16]. Despite this property, the approximate mean does not equal the true mean because in practice convergence is never exactly achieved. This combined with numerical errors which accumulate during message passing results in the error not equalling zero.

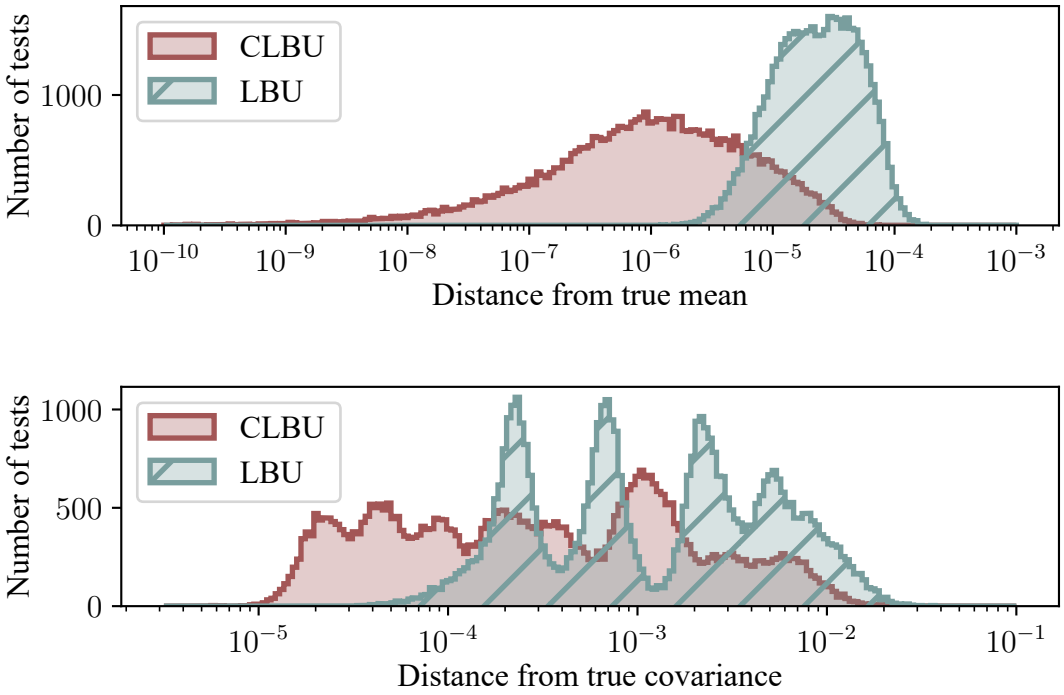


Figure 4.10: Comparison between the distance to the true mean and true covariance of the approximate marginal distributions estimated by CLBU and LBU. CLBU shows an improvement in both the approximate mean and covariance.

The quality of approximation decreases as the value of the hyperparameter n decreases as shown in 4.11. This result is similar to that of the discrete tests for different $\alpha_\circ$ values, and for similar reasons. Lower n lead to stronger correlations between the random variables that can result in more frustrated graphs as well as stronger feedback loops. Inference using CLBU on Gaussian networks was not affected as much by the cluster variation depth compared to the discrete tests as seen in Figure 4.11. Note that Figure 4.11 only includes the tests for which every method converged.

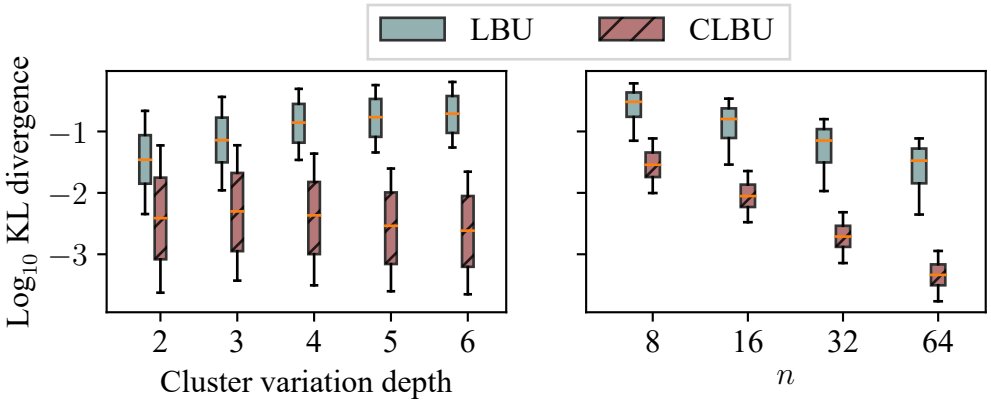


Figure 4.11: The 5%, 25%, 50%, 75% and 95% quantiles of the cumulative Kullback-Leibler divergence for the tests where LBU and CLBU converged.

Both LBU and CLBU have good convergence when used for inference on the randomly

generated Gaussian networks compared to the discrete tests as shown in Figure 4.12.

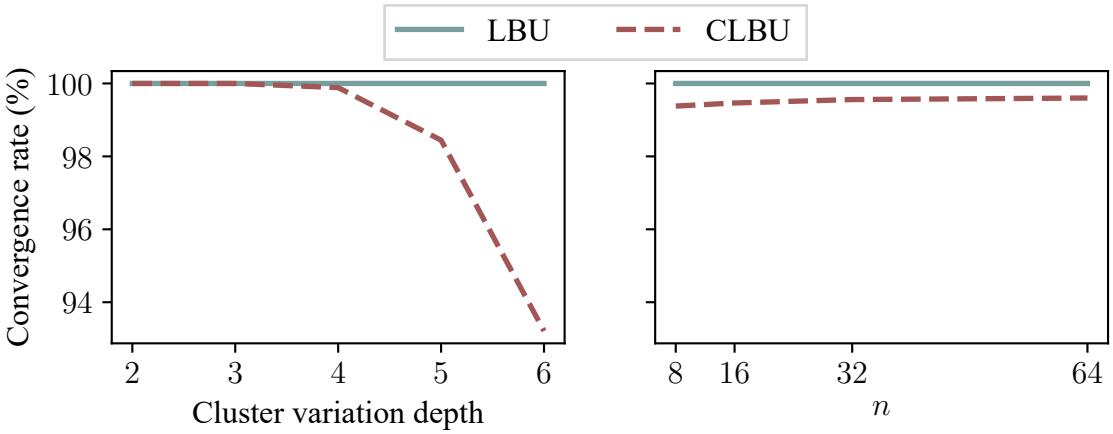


Figure 4.12: Convergence rates for CLBU and LBU on Gaussian networks. As with the discrete case, convergence decreases with higher cluster variation depth. The Gaussian networks have better convergence than the discrete networks.

CLBU for Gaussian networks is made slow due to the steps taken to prevent negative definite covariance matrices from ever occurring as shown in Figure 4.13. An implementation that allows for negative definite matrices during the intermediary steps, which do not require positive definite matrices, could potentially provide a significant speed increase. This is not possible with the current code base since the triangular form of the information matrices are used (see Subsection 4.3.3).

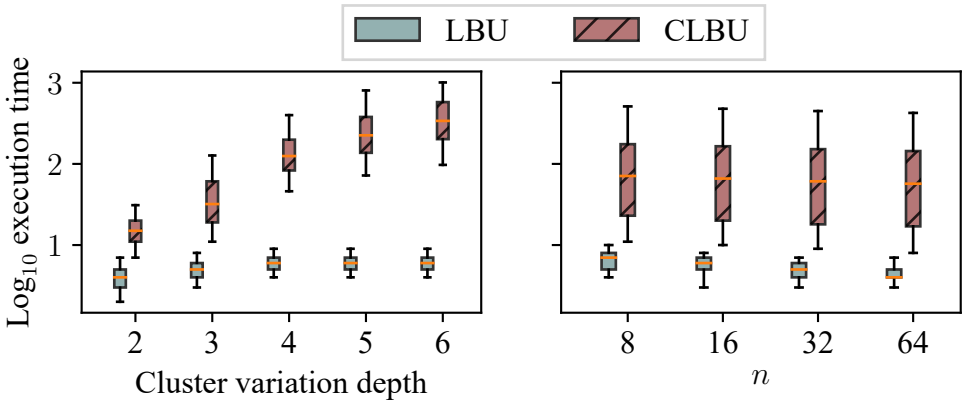


Figure 4.13: Execution time comparison for LBU and CLBU on Gaussian networks. CLBU is slower than LBU due inefficient operations required to avoid negative definite matrices.

In conclusion, CLBU produces improved approximate marginal distributions compared to LBU, improving the approximation for 99% of tests executed. Both methods have high convergence rates with LBU converging for all tests and CLBU for 99% of tests.

4.4. The Ising Model

This section compares the performance of the message-passing schemes on the Ising model. The Ising model statistically models a mesh of interacting atoms. It applies to ferromagnetic materials where the direction of spin of each atom in the material is represented by a binary random variable $X_i \in \{-1, +1\}$. The atoms are generally in a lattice structure where neighbouring atoms interact with each other. A Markov network can be used to model this lattice, where the edges between nodes represent the interaction between the atoms. The probability distribution describing the interaction of two neighbouring atoms is given as

$$P(X_a = x_a, X_b = x_b | w_{a,b}) = \frac{1}{Z} e^{w_{a,b} x_a x_b}, \qquad x_a, x_b \in -1, 1, \tag{4.20}$$

where $w_{a,b}$ determines how strongly the two atoms interact [2, p 126]. If $w_{a,b} > 0$ the two atoms want to agree on their spin and disagree if $w_{a,b} < 0$.

The factor scopes $\{\mathcal{X}_i : i \in \mathcal{I}_\Phi\}$ are created by clustering neighbouring variables in the Markov network into clusters of 4 as shown in Figure 4.14. These factor scopes result in a cluster graph that is densely connected where the running intersection property results in many sepsets omitting a variable (see Figure A.1 in Appendix A).

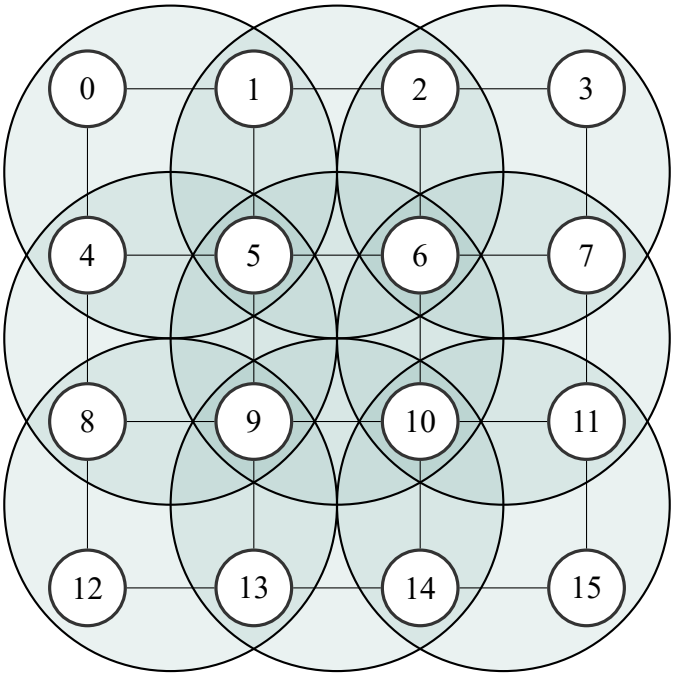


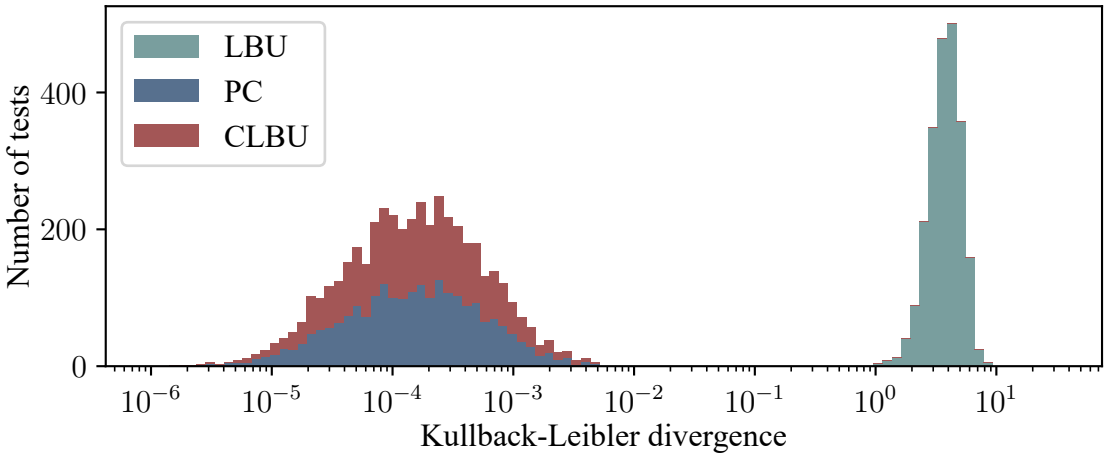
Figure 4.14: A 4×4 Markov network. Each node in the Markov network has a number indicating which random variable it is associated with. For our Ising model these random variables are grouped together in intersecting sets of 4, also known as cliques, that form the factor scopes. These factor scopes are indicated by the large circles, with the random variables associated with the encircled vertices forming a single factor scope.

The factor probability distributions in Φ are generated from the atom interaction distribution described in Equation 4.20. The parameter $w_{a,b}$ for each variable pair in the Markov network is

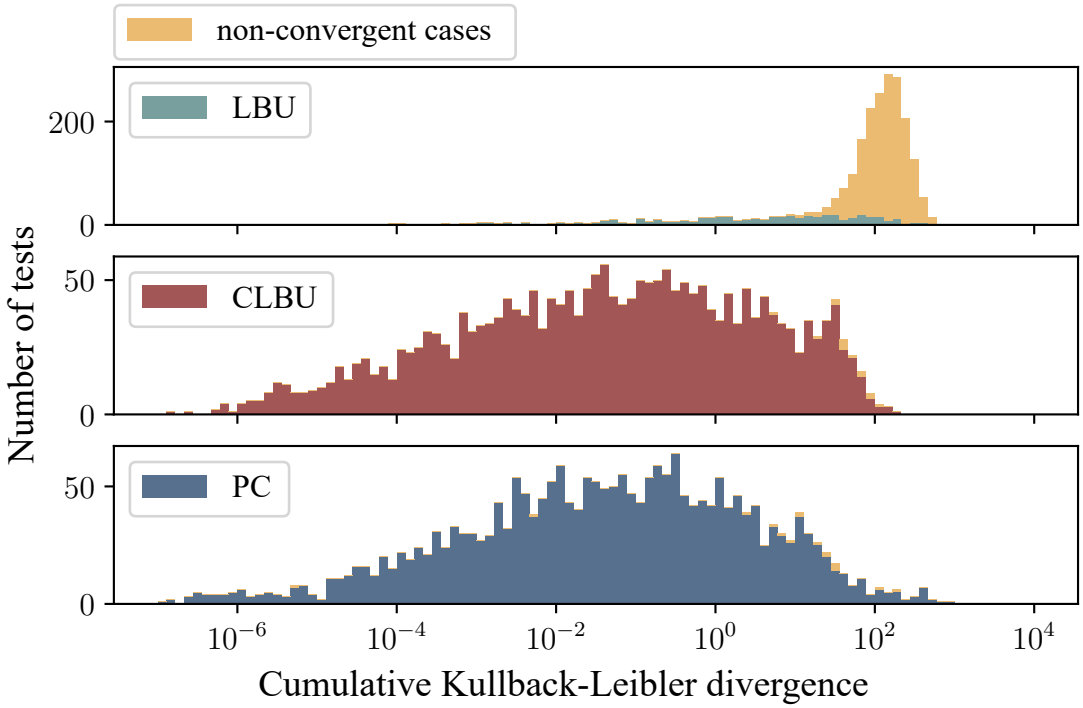
sampled from a uniform distribution $w_{a,b} \sim U(-C, C)$. The preferred state of each atom is also randomized by creating a categorical probability distribution with random potentials $\boldsymbol{\theta}$ describing the atom state. This creates a system where each atom is biased to have a different spin, and each atom pair have different attraction/repulsion interactions. Higher values of C create stronger interactions. A factor ϕ has 4 atoms and 4 atom interaction pairs as shown in Figure 4.14. Let $\mathcal{E}$ be the set of all edges in the Markov network, then the factor distributions $\phi_i(\mathcal{X}_i)$ are calculated as

$$\phi_i(\mathcal{X}_i) = \frac{1}{Z} \prod_{E_{a,b} \in \mathcal{E}} P(X_a, X_b | w_{a,b})^{[X_a, X_b \in \mathcal{X}_i]} \prod_{X_a \in \mathcal{X}_i} P(X_a | \boldsymbol{\theta}_a), \qquad \forall i \in \mathcal{I}_\Phi, \tag{4.21}$$
$$w_{a,b} \sim U(-C, C),$$
$$\boldsymbol{\theta}_a \sim \text{Dir}(\boldsymbol{\alpha} = [1, 1]).$$

There were 4500 tests executed by generating random sets of factors as described in Equation 4.21, to form the joint distributions of the Ising model. The Markov network for all tests was a 7-by-7 grid, which is a larger version of the one shown in Figure 4.14. LBU, CLBU and PC were used to estimate the marginal distributions of the Ising model and compared with the correct marginal distributions, which are computed by performing inference on an equivalent junction tree. As with the discrete and continuous tests, the error is measured using the cumulative Kullback-Leibler divergence as shown in Equation 4.8. Figure 4.15 shows that CLBU and PC produce significantly better results than standard LBU. CLBU produces approximate marginal distributions that are closer to the true marginal distributions while also having better convergence than standard LBU. The execution time of CLBU is also much less than LBU and PC as shown in Figure 4.16.



(a) Tests with C equal to 1, all tests converged. The histogram displays CLBU stacked on top of PC.



(b) Tests with C equal to 10.

Figure 4.15: Comparison between the message-passing schemes applied on a 7-by-7 Ising model Markov network for different atom interaction strengths. Higher values of C correspond with stronger correlations between the random variables. Figures 4.15a and 4.15b show that the distance of the approximate marginal distributions to the true marginal distributions of the CLBU and PC is far less than that of standard LBU. For high values of C, CLBU and PC have much better convergence than LBU.

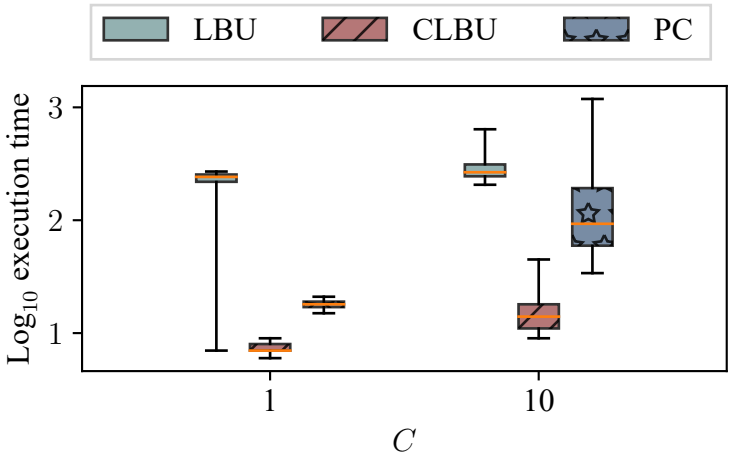


Figure 4.16: Execution time comparison with CLBU having the fastest execution time among the three methods.

For the Ising model, both PC and CLBU provide improvements over LBU that are much greater than what was seen for the discrete tests of Section 4.2, especially when it comes to convergence. The Ising model has two main properties that contribute to the effectiveness of CLBU and PC. The first is that the density of connections in the graph is not high compared to the previous discrete tests. The cluster variation depth for the factor scopes of the Ising model is 2, which is an indicator for good convergence as shown in Figure 4.5. The second reason is that the factors can be factorized further. For example, the factor $\phi(0, 1, 4, 5)$ is made up of simpler pair-wise factors $\phi(0, 1), \phi(1, 5), \phi(4, 5)$ and $\phi(0, 4)$. Higher-order interactions are therefore not as important in the Ising model as they are for the tests of Section 4.2. Using CLBU or PC, clusters can efficiently communicate these pairwise interactions, which is not the case for LBU since certain messages must omit a random variable in order to satisfy the RIP.

4.5. Improving Convergence

Thus far we have seen that extremely densely connected graphs result in CLBU having low convergence rates. In this section, we drastically improve convergence by requiring that the cluster graph satisfies the MRIP, which is a weaker constraint than the TRIP. This allows us to simplify the cluster graph by omitting certain conditional messages.

When analysing the tests from Section 4.2.3 to 4.4 a negative correlation between convergence and the number of a certain type of message, refer to as *purely conditional*, was identified. A purely conditional message conveys no marginal information about any specific random variable, only on the correlations between variables. In a purely conditional message, the scope of the conditioning factor is equal to the sepset. The messages between C_1 and C_2 for the example graph in Section 3.4 are purely conditional messages. An extremely simple and effective way of improving convergence is to refrain from adding any purely conditional messages to the graph. The resulting cluster graph is guaranteed to satisfy the MRIP, but not the TRIP.

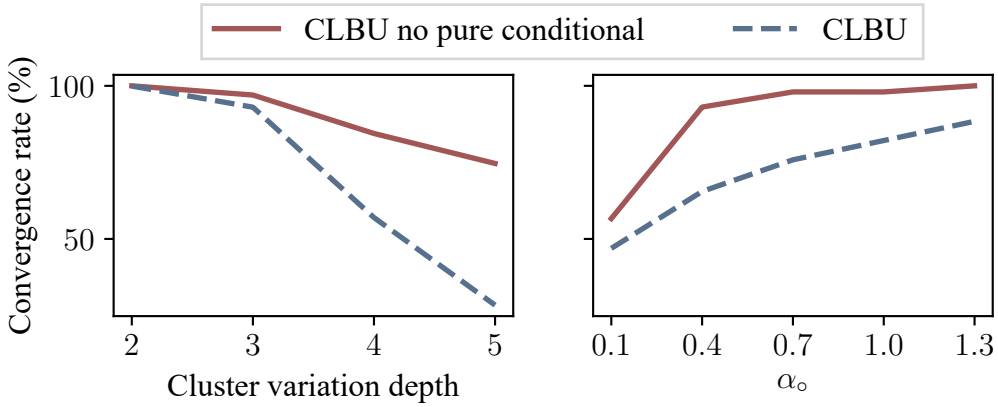


Figure 4.17: CLBU has better convergence when not including purely conditional messages.

The discrete tests of Section 4.2 are repeated here, however, this time a connection between two clusters is not added if the resulting conditional messages are purely conditional. Figure 4.17 shows the drastic increase in convergence when excluding purely conditional messages. This improvement comes at a reduction in the potential improvement accuracy CLBU can have over LBU as shown in Figure 4.18. Note that this is not the limit on improving convergence, since other conditional messages can be chosen to be standard messages with reduced scope as mentioned in Section 3.4.3. In fact, for a specific set of clusters there exists a spectrum of graphs, all of which satisfy the MRIP, ranging from graphs with no conditional edges (the graph then satisfies the RIP) continuing on with more and more conditional edges until the graph satisfies the TRIP. Knowing which graph in this range will produce the best results, along with which conditional messages to include and which not is far from obvious and could be investigated further in future. Therefore, removing purely conditional messages is only one possible guideline for improving the chances of CLBU to converge.

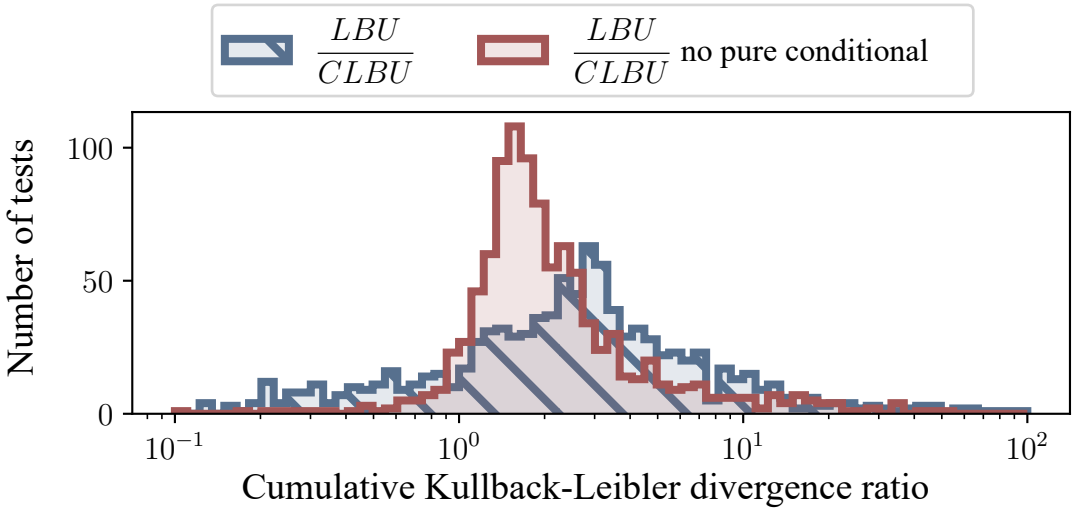


Figure 4.18: Not including purely conditional messages improves the approximations for the cases where CLBU would otherwise not have converged, but also reduces the improvement for the cases where it would have converged despite having purely conditional messages. This plot includes non-convergent tests, the majority of which are tests with a ratio less than one.

We have seen that reducing the number of purely conditional messages improves convergence. This raises the question: can the number of purely conditional messages be reduced, not by omitting them from the graph, but choosing a different starting graph and order in which messages are added? It turns out that this is indeed possible as shown in Appendix A. Unfortunately, constructing graphs that minimizes the amount of purely conditional messages is harder to do than simply not including them.

4.6. Summary

From the extensive tests conducted on randomly generated discrete, continuous and Ising distributions, it was empirically showed that, when conditional message passing converges, it produces approximate marginal distributions that are closer to the true marginal distributions than the approximations produced by standard message passing on cluster graphs. For certain distributions, such as those of the Ising model, CLBU significantly improves upon the approximate marginal distributions, execution time and convergence. However, for other distributions, CLBU can have difficulty converging. For the cases where it does not converge we, have identified a simple and effective way of increasing convergence.

When compared to PC, CLBU produces remarkably similar results. They both seem to minimize the same approximate energy functional as given by the cluster variation method. The largest difference is in execution time, where CLBU outperforms PC. This is largely due to conditional message passing having been developed as a belief update algorithm.

Chapter 5

Conclusion and Future Work

5.1. Conclusion

This thesis developed conditional message passing, which is a novel message-passing scheme for use on cluster graphs with the aim of producing improved approximate marginal distributions compared to standard message passing. To achieve this improvement, larger sepsets were included in the cluster graphs than was previously allowed by the running intersection property, with the motivation that it would allow clusters to exchange more information. To allow for these larger messages, conditional message passing includes loop correction in the form of conditioning factors. The region graph method also allows for larger messages, but where region graph methods use multiple messages to correct for the overcounting of certain beliefs, conditional message passing concentrates the correction into single messages. This allows for simpler and more intuitive graphs, compared to the region graph method, by removing the need for multiple levels of nodes, which each have their own connections and messages (see Appendix A, Figures A.4 and A.5).

The experiments conducted in Section 4 show that conditional message passing does indeed produce improved approximate marginal distributions when it converges, which includes the approximate marginals of randomly generated discrete distributions, continuous distributions and Ising model distributions. Compared to the region graph method, conditional message passing behaves remarkably similarly. Both produce similar approximations and both converge for a similar number of tests. For densely connected discrete networks, both conditional loopy belief update (CLBU) and the parent-to-child (PC) message passing schemes struggle to converge. A simple and flexible method for drastically increasing the number of tests for which CLBU converges was identified (see Section 4.5). While this method can help remove problematic connections, it does not help with constructing optimal cluster graphs, which is a much harder problem that requires future investigation similar to the work performed by Max Welling on choosing optimal regions for region graphs [17].

In conclusion, conditional message passing provides an alternative to the region graph method, even though it was derived from a completely different point of view. It allows for a richer family of cluster graphs that thus far have been limited by the RIP. The alternative and intuitive approach explored in this thesis, namely loop correction on cluster graphs, might lead to even

further improvements in message-passing algorithms in the future.

5.2. Future Work

Thus far only loops where the same variable occurs through the entire loop have been considered. Conditional message passing then limits information from looping in these loops, but does not compensate for general loops.

This section briefly explores the possibility of generalizing conditional message passing to include all types of loops. In other words, possibly also compensating for loops where the same variable does not occur throughout the loop. To this end, future work might investigate a more general conditioning factor that should ideally include the current definition as a special case.

One such definition is as follows: Consider a message $\mu_{a,b}(\mathcal{S})$. We want to remove any information that will loop back through *any* other path. Let $\boldsymbol{D}$ contain $\mathcal{S}$ and all possible subsets of $\mathcal{S}$. For each set in $\boldsymbol{D}$ we define a coefficient σ_i that indicates, approximately, how the information of the message $\mu_{a,b}(\mathcal{D}_i)$ will loop back. Using these parameters, we can compute counting numbers c_i for each set in $\boldsymbol{D}$ as

$$c_i = \sigma_i - \sum_{j \in \mathcal{I}_{\boldsymbol{D}}} c_j [\mathcal{D}_i \subset \mathcal{D}_j]. \tag{5.1}$$

The more general conditioning factor can be defined as in Equation 3.14,

$$\Theta_{a,b}(\mathcal{S}_{a,b}) = \prod_{i \in \mathcal{I}_{\boldsymbol{D}}} \left(\sum_{\mathcal{C}_a \setminus \mathcal{D}_i} \Psi_a(\mathcal{C}_a) \right)^{c_i}. \tag{5.2}$$

This definition of the conditioning factor is more general than the one defined in Subsection 3.4.2. Here the sets in $\mathcal{D}$ are not only all possible intersections of running intersections, but all possible subsets of the sepset. Also, the counting numbers are not required to add up to one.

During the process of iteratively adding sepsets, if we assume that σ_i equals 1 if there exists a path such that $\mathcal{D}_i$ is present in all sepsets along a path and 0 otherwise, we get the same conditioning factors as defined in Subsection 3.4.2. In other words, if we assume that there is no information loop along a path if the running intersection is empty (which is not the case), then this more general definition reduces to conditional message passing investigated in this thesis.

Future work could determine whether a generalized definition of a conditioning factor is indeed a valid loop correction method and, if it is, how to determine the σ parameters for each message in the graph. Determining the optimal σ parameters might be as expensive to compute as exact inference. Any estimate of the σ parameters will likely also change during message passing.

Bibliography

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, ser. Intelligent Robotics and Autonomous Agents. Cambridge, Mass: MIT Press, 2005.

[2] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. New York, NY: Springer Science + Business Media, 2006.

[4] J. Pearl, *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*, ser. Morgan Kaufmann Series in Representation and Reasoning. San Mateo, Calif: Kaufmann, 1988.

[5] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2282–2312, 2005.

[6] P. Z. Peebles Jr. and B. E. Shi, *Probability, Random Variables and Random Signal Principles*. McGraw-Hill Education, 2015.

[7] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[8] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[9] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[10] G. R. Shafer and P. P. Shenoy, "Probability propagation," *Annals of Mathematics and Artificial Intelligence*, vol. 2, pp. 327–351, 1990.

[11] A. Pelizzola, "Cluster variation method in statistical physics and probabilistic graphical models," *Journal of Physics A: Mathematical and General*, vol. 38, no. 33, aug 2005.

[12] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 50, no. 2, pp. 157–194, 1988.

[13] S. Streicher and J. du Preez, "Graph coloring: Comparing cluster graphs to factor graphs," in *Proc. SAWACMMM '17*. New York, NY, USA: Association for Computing Machinery, 2017, p. 35–42.

[14] G. Elidan, I. McGraw, and D. Koller, "Residual belief propagation: Informed scheduling for asynchronous message passing," in *Proc. 22nd Conference on Uncertainty in Artificial Intelligence*, Cambridge, MA, USA, 2006, p. 165–173.

[15] S. W. Nydick, "The wishart and inverse wishart distributions," *Electronic Journal of Statistics*, vol. 6, no. 1-19, 2012.

[16] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in gaussian graphical models of arbitrary topology," *Neural Computation*, vol. 13, no. 10, pp. 2173–2200, 2001.

[17] M. Welling, "On the choice of regions for generalized belief propagation," in *Proc. 20th Conference on Uncertainty in Artificial Intelligence*, Arlington, Virginia, USA, 2004, p. 585–592.

Appendix A

Ising Model Graphs

This appendix provides some example graphs for a 4 by 4 Ising grid, showing how the same set of clusters can be connected in multiple different ways. The notation $\frac{i,j}{\{i\}\{j\}}$ is used to indicate how the conditional message is constructed,

$$\begin{aligned}\psi_{a,b}(\mathcal{S}_{a,b}) &= \frac{\Psi_a(\mathcal{S}_{a,b})}{\Theta_{a,b}(\mathcal{Q}_{a,b})}, \\ &= \frac{\Psi_a(X_i, X_j)}{\Psi_a(X_i)\Psi_a(X_j)}.\end{aligned}$$

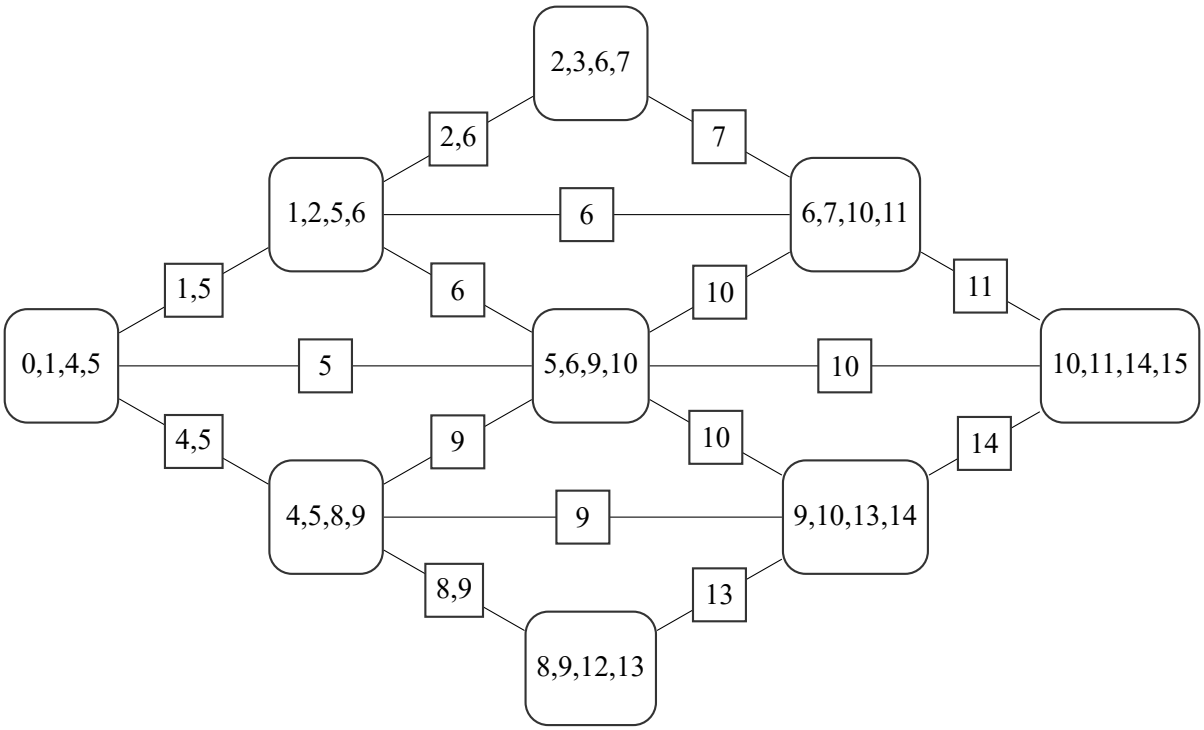


Figure A.1: Example of a cluster graph satisfying the RIP for a 4×4 Ising grid.

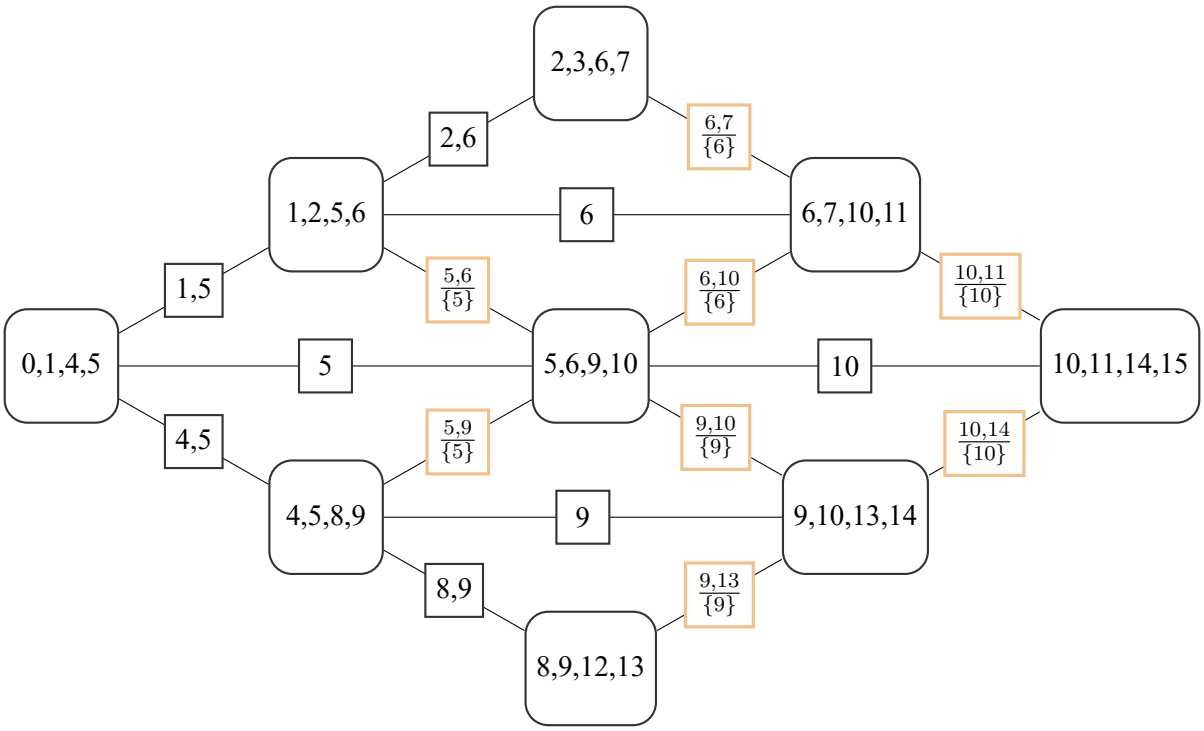


Figure A.2: Example of a cluster graph that satisfies the TRIP for a 4×4 Ising grid. This graph has no purely conditional messages.

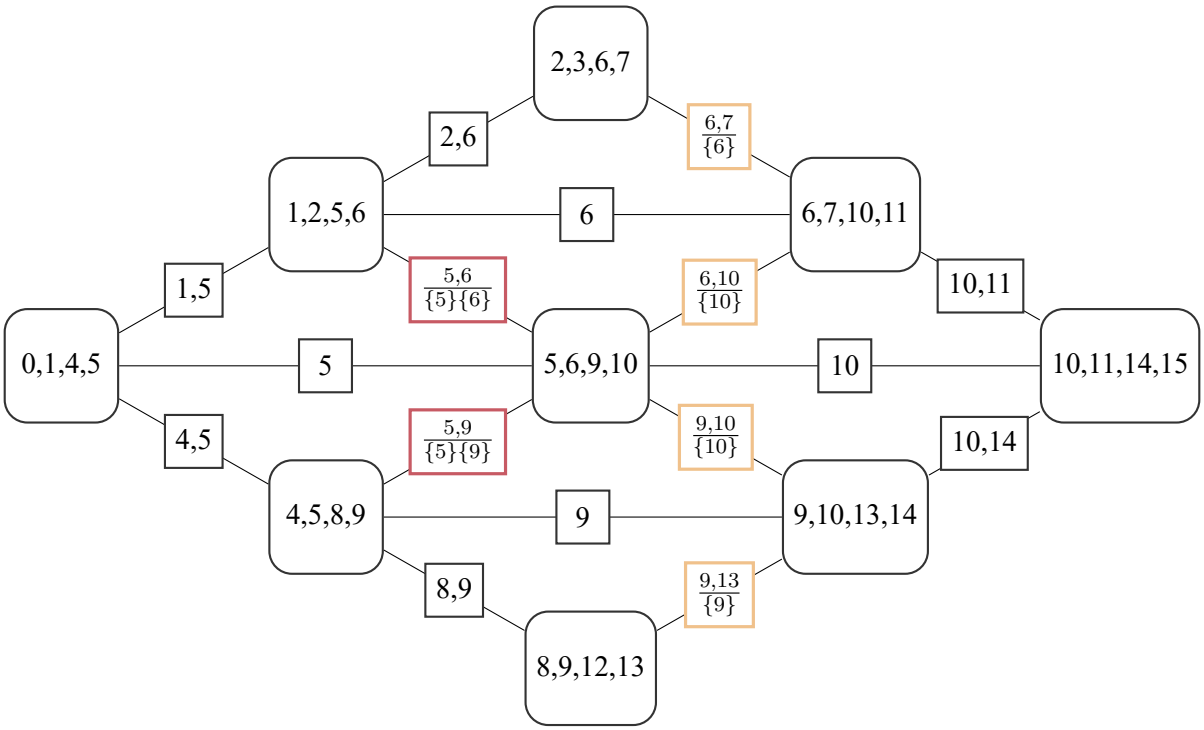


Figure A.3: Example of a cluster graph that satisfies the TRIP for a 4×4 Ising grid. This graph does include purely conditional messages (red).

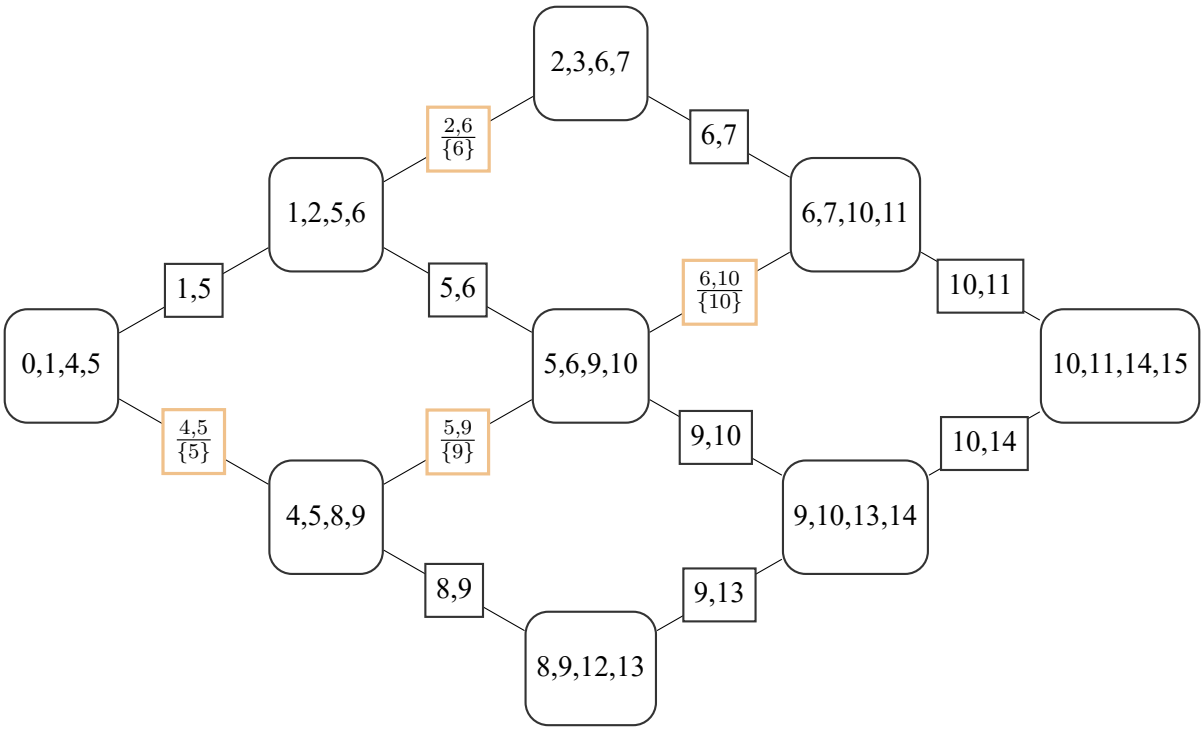


Figure A.4: Example of a cluster graph that satisfies the TRIP for a 4×4 Ising grid. This example has the minimum amount of edges between the clusters. Algorithm 1 will automatically produce the optimal graph for the Ising model when only given the cluster scopes (no starting graph is provided).

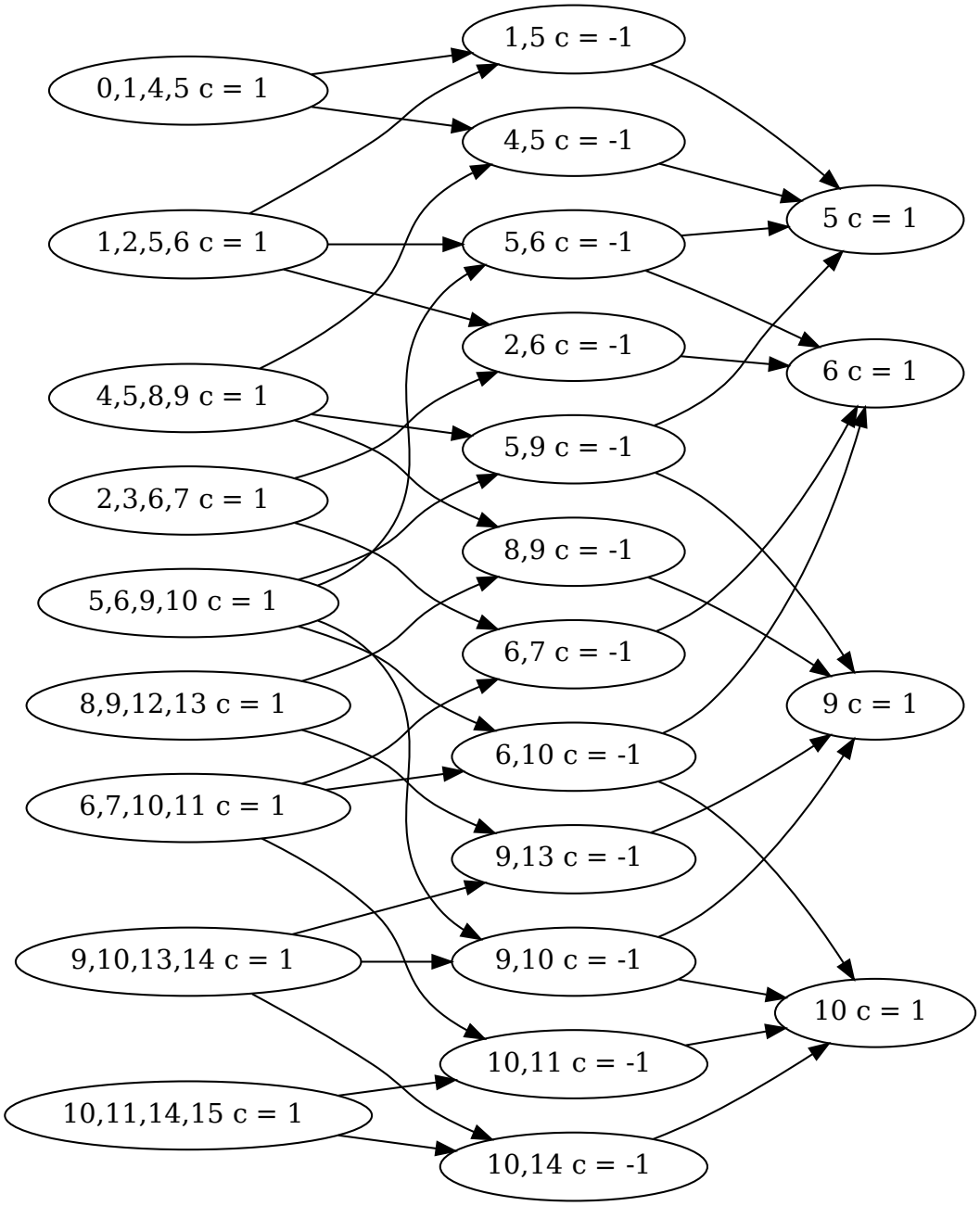


Figure A.5: A region graph for a 4×4 Ising grid with regions from the cluster variation method and that satisfies the region graph condition. The counting numbers c are also included