Brown University

Doctoral Dissertation

Message Passing Dynamics of Belief Propagation Algorithms

Author:
Anna Grim

Advisor:
Pedro Felzenszwalb

A dissertation submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Division of Applied Mathematics

May 2022

This dissertation by Anna Grim is accepted in its present form by the Division of Applied Mathematics as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date____________________ ____

Pedro Felzenszwalb, Ph.D., Advisor

Recommended to the Graduate Council

Date____________________ ____

Johnny Guzman, Ph.D., Reader

Date____________________ ____

Matthew Harrison, Ph.D., Reader

Approved by the Graduate Council

Date____________________ ____

Andrew G. Campbell, Dean of the Graduate School

CV

Education

Brown University *September 2016 - May 2022*

Doctor of Philosophy in Applied Mathematics

Masters in Applied Mathematics

University of St. Thomas *September 2012 - May 2016*

Bachelor of Science, Mathematics

Minors in Computer Science

Publications

1. Grim, A. and Felzenszwalb, P. Improving Belief Propagation with Numerical Homotopy Continuation, *In preparation.*

2. Grim, A. and Felzenszwalb, P. Convex Combination Belief Propagation on Factors Graphs with Numerical Homotopy Continuation, *In preparation.*

3. Grim, A. Local Stability of Belief Propagation on the QMR Network, *In preparation.*

4. Grim, A. and Felzenszwalb, P. Convex Combination Belief Propagation Algorithms. (submitted, 2021).

Teaching

Brown University

Teaching Assistant, Methods of Applied Mathematics I	*Fall 2017, Spring 2018*
Teaching Assistant, Computational Probability and Statistics	*Fall 2020*
Teaching Assistant, Recent Applications of Prob. and Stat.	*Spring 2021*
Teaching Assistant, Statistical Inference I	*Fall 2021*
Teaching Assistant, Graphs and Networks	*Spring 2022*

Abstract of *Message Passing Dynamics of Belief Propagation Algorithms,*

by Anna Grim, Ph.D., Brown University, May 2022

Our work is motivated by the classical belief propagation algorithms which are well-known for obtaining state of the art results in certain settings, but often fail to converge when the underlying graph has complex topology. The main objective of this thesis is to present a two-part solution to this classical problem. First, we describe an algorithm referred to as *convex combination* belief propagation. This method was developed by modifying the message passing operator in loopy belief propagation so that it's more robust to graphs with cycles. The main advantage of this algorithm is that it's guaranteed to converge on graphs with arbitrary topology. Although this algorithm has good theoretical properties, one disadvantage is that it is generally less accurate than the classical algorithm when both converge.

Second, we build upon this work by incorporating a homotopy operator that gradually deforms the message passing operator from convex combination belief propagation into the operator used in loopy belief propagation. Under this framework, convex combination belief propagation obtains an initial solution. Then we improve the accuracy of the solution by using the homotopy operator in a continuation scheme. The outcome of this work is an approximate inference algorithm that is significantly more accurate than convex combination belief propagation, while also converging at a much higher rate than loopy belief propagation.

Lastly, we demonstrate the usefulness of our approximate inference algorithms by applying them to real-world problems. We discuss medical diagnostic inference on the QMR network which models causal relations between a set of diseases and findings. We use our homotopy continuation algorithm to perform inference in this problem. Our results show that this algorithm always converges and obtains exceptional results on this task.

Acknowledgements

First, I would like to thank my research advisor Pedro. His guidance and support over the last few years have been crucial to this research. I very much admire his approach to research along with his balance between theory and meaningful applications. I hope to continue to pursue interesting and challenging problems with the same creativity and practicality.

I also would like to thank Matt Harrison and Johnny Guzman for taking the time to serve on my thesis committee and providing helpful feedback. I greatly appreciate the insightful discussion and suggestions of future research directions.

I am incredibly grateful to my undergrad advisor Cheri Shakiban. I would like to thank her leading so many undergraduate research projects and encouraging me to pursue a PhD. She has been amazing mentor over the years and opened a door to many opportunities. I would also like to thank Peter Olver for his guidance and support. His passion for mathematics is inspiring and I hope to approach research with the same curiosity and vigor.

I am very thankful for the support of friends who have been a source of encouragement and entertainment. Thank you to the previous generations of APMA graduate students who have continued to pass down a collaborative and welcoming culture. I'd like to thank Ernesto for your friendship over the years along with the countless hours of working on problem sets and playing pool at the GCB. I would also like to thank Angelina for being a good tennis partner and even better friend. Thank you to the rock climbing community across the pure and applied math departments. I would also like to thank my office mates for lively and insightful discussions. A special thanks to Becky for so many good conversations and for having a good sense of humor. I am very thankful for Patrick who has been incredibly supportive and a constant source of inspiration. His love for problem solving is contagious and I hope to continue to collaborate on projects.

Finally, I would like to thank my family for always being a phone call away, I cannot imagine going through graduate school without their continual support. I also greatly appreciate their encouragement to work hard and persevere when challenges arise. A special thanks to Cheerio for all of the nice walks and adding so much joy to my life.

Contents

List of Figures

CHAPTER 1

Introduction

The focus of this chapter is to discuss the motivations behind this thesis and outline our main contributions to the scientific community. Section 1.1 broadly introduces the technical domain of this work, then briefly describes the fundamental problems that motivated the developments in this thesis. Section 1.2 describes our general approach and then summarizes our main contributions. We conclude by providing a concise overview of the content and organization of this thesis in Section 1.3.

1.1 Motivations

Graphical models are a well-established framework for probabilistic modeling due to their ability to express complex relationships between variables in a system. The main components of this model are a joint probability distribution that governs the system and a graph which captures certain statistical properties. Each node in the graph represents a variable and edges encode statistical relationships. Thus, the graph provides a global representation of the structure of the system in terms of local relationships.

and reflects local dependencies between variables. The expressive power and flexible framework of graphical models makes them a natural tool for solving problems in computer vision, machine learning, bioinformatics, and engineering (see, e.g., [13], [24], [54], [56], [92]).

In these types of applications, a central computational objective is to perform probabilistic inference. We use the term *probabilistic inference* to refer to the following inference tasks: *marginal* inference and *maximum a posterior (MAP)* inference. Marginal inference involves computing marginal probability distributions, while MAP inference involves determining the optimal configuration of a system that maximizes the joint distribution. Probabilistic inference is an important problem because the outcome of this calculation forms the basis behind machine-based learning and decision making. Exact inference in arbitrary graphical models is computationally intractable because this problem has been proven to be NP-hard. This well-known fact has driven a large number of research efforts to develop efficient approximation algorithms.

Belief propagation is a widely used local message passing algorithm that leverages the structure of the graph to perform inference in graphical models. This algorithm was first introduced by Judea Pearl in the 1980s as a method that performs exact inference in polynomial time when the underlying graph is tree-structured [64]. When the graph contains cycles (i.e. loops), the method can be adapted into a fixed point iteration scheme referred to as *loopy* belief propagation. In this setting, the algorithm performs approximate inference by utilizing an operator that facilitates local message passing between all neighboring nodes in the graph.

Loopy belief propagation has gained popularity within the artificial intelligence community because it obtains state of the art results in certain settings such as error correcting codes and image analysis (see, e.g., [11], [19], [21], [25], [74]). However, loopy belief propagation is also notorious for being unreliable when the underlying graph has complex topology. This is problematic because many real world networks fall into this category. Although there are alternative methods that can perform approximate inference in these settings, loopy belief propagation is significantly more efficient and accurate in certain settings.

1.2 Main Contributions

We are both motivated and intrigued by the challenge to develop efficient algorithms that perform approximate inference in graphical models with cycles. Although there is a large body of work on this subject, many of these methods are computationally expensive, fail to converge on challenging problems, or difficult to use in practice. We present a solution that differentiates itself from other approaches by being simple to implement and designed to converge on the most difficult inference problems.

In this thesis, we address four key aspects of performing inference in graphical models with cycles: 1) development of efficient **approximation algorithms** that address both types of inference problems, 2) **theoretical analysis** that specifies when the algorithms converge and characterizes the solution, 3) **experimental evaluation** that compares the performance of our algorithms to other methods, 4) using our approximation algorithms to solve difficult **real-world applications** where conventional methods fail. In Figure 1.1, we provide a visual outline of both the organization and contributions in this thesis.

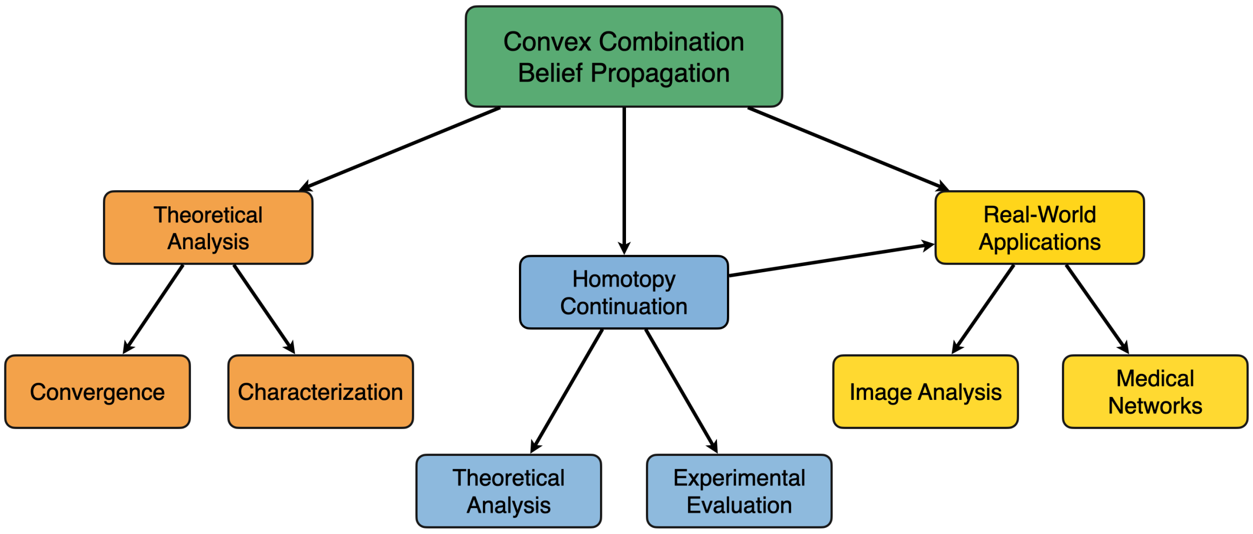


FIGURE 1.1: Overview of main contributions in this thesis

1.2.1 Approximation Algorithms

We present *convex combination* belief propagation which is a convergent alternative to loopy belief propagation. This algorithm was developed by modifying the message passing operator in loopy belief propagation in way that makes it more robust to graphs with cycles. The result is a class of algorithms that are designed to converge on the most difficult inference problems. In addition, our algorithms are fast and simple to incorporate within any existing implementation of belief propagation, which is a major advantage compared to other methods. Convex combination belief propagation is also foundational to subsequent contributions in this thesis. We emphasize this point in Figure 1.1 by showing several research directions that branch out from this initial development.

In fact, convex combination belief propagation is our initial solution to performing inference in graphical models with cycles. One limitation of this algorithm is that it tends to be less accurate than loopy belief propagation when both algorithms converge. We build upon this work by incorporating a homotopy operator that gradually deforms the message passing operator from convex combination belief propagation into the traditional operator used in loopy belief propagation. Under this framework, convex combination belief propagation obtains an initial solution. Then we improve the accuracy of the solution by using the homotopy operator in a continuation scheme. The outcome of this work is an approximate inference algorithm that is significantly more accurate than convex combination belief propagation, while also converging at a much higher rate than loopy belief propagation.

1.2.2 Theoretical Analysis

A central objective of this thesis is to accompany algorithm development with theoretical analysis. Our general approach is to consider belief propagation as a discrete-time map, then utilize ideas from the theory of dynamical systems to understand the behavior of this algorithm. In this thesis, we refer to a discrete-time map $F : \mathbb{R}^m \to \mathbb{R}^m$ as a function that acts on a vector $x \in \mathbb{R}^m$ according to

$$x^{(n+1)} := Fx^{(n)},$$

where n is a discrete time index. It is natural to analyze belief propagation from a dynamical systems perspective because both convex combination and loopy belief propagation are fixed point iteration algorithms.

In Chapters 4 and 5, we analyze the theoretical properties of convex combination belief propagation. One important result is that this algorithm is guaranteed to converge to a unique solution (fixed point) independent of the graph's topology and initialization of the fixed point iteration scheme. We prove this result by showing that the message passing operator in convex combination belief propagation is a contraction and then invoke Banach's fixed point theorem. In addition, we also provide a characterization of the solution obtained with convex belief propagation for certain classes of graphs.

Chapter 6 focuses on belief propagation with homotopy continuation. The main idea behind this method is to trace a path of fixed points that is parametrized by time t, where the path is formed by gradually varying t from 0 to 1. This path originates at the unique fixed point of convex combination belief propagation and terminates at a fixed point of loopy belief propagation. In order for any continuation algorithm to be feasible, this path must be continuous with respect to time so that it can be numerically traced as this parameter is gradually varied. We provide a rigorous argument that uses Brouwer's fixed point theorem and a generalization of the implicit function theorem to prove that this property holds for arbitrary graphical models.

Lastly, Chapter 7 focuses on the local stability of belief propagation in the application of performing approximate inference in a medical diagnostic network called the Quick Medical Reference (QMR) network. In our experimental evaluation, we observe that belief propagation with homotopy continuation converges on every problem in this application. These results raise the question of whether the QMR network is a special case where our continuation algorithm always converges. One possible explanation is that the fixed points of belief propagation are locally stable in the case of the QMR network. Under this assumption, numerical homotopy continuation would converge (in theory) because this implies the existence of a small enough time step such that the current approximate fixed point lies in the basin of attraction of the next fixed point.

From a dynamical systems perspective, a classical approach to this problem is to analyze the eigenvalues of the Jacobian of the message passing operator. Although this approach is generally intractable when the Jacobian is very high dimensional, we use the unique structure of the QMR network to drastically simplify the Jacobian and computation of its eigenvalues. Our analysis draws a connection between the topology of the graph and structure of the Jacobian. We prove that the Jacobin consists of nested triangular matrices, then leverage this fact to reduce the complexity of computing its eigenvalues. In addition, we also analyze how the parameters of this model affect the local stability of belief propagation fixed points.

1.2.3 Experimental Evaluation

Another important objective of this thesis is to show that our inference algorithms have both strong theoretical properties and good performance in applications. In particular, we aim to develop inference algorithms that reliably outperform loopy belief propagation. We use the spin glass models[1] from statistical physics as a baseline inference problem to compare the performance of our algorithms against loopy belief propagation. Spin glass models are often used for this purpose because this model poses a difficult inference problem where loopy belief propagation tends to fail.

The performance of our algorithm is determined by analyzing the runtime, accuracy, and convergence rate. Experimental evaluation is a crucial part of algorithm development because it reveals both the advantages and limitations of an algorithm. Our initial experiments involving convex combination belief propagation reveal that loopy belief propagation tends to be more accurate when both algorithms converge. This realization led to the development of a belief propagation algorithm that utilizes homotopy continuation. We show that this improved algorithm (on average) outperforms loopy belief propagation across all performance metrics.

1.2.4 Real-World Applications

Although we perform an experimental evaluation of our inference algorithms on spin glass models, this problem is only a toy example. Thus, we demonstrate the usefulness of our approximate inference algorithms by applying them to real-world problems. In Chapter 4, we use convex combination belief propagation to perform image restoration. The objective of this image analysis problem is to estimate a clean (original) picture given a corrupted version of it. The restoration problem is formulated as "pixel labeling" where inference involves choosing an optimal label for each pixel.

In Chapter 7, we discuss medical diagnostic inference and the QMR network. Diagnostic inference is both a difficult and important problem which has attracted much attention in the last few decades. The ultimate goal is to design an intelligent system that assists in diagnosing a patient who exhibits some observed findings (or symptoms). One classical approach to this problem is to develop a probabilistic model that utilizes a network which describes the relationship between diseases and findings. The QMR network is a directed bipartite graph that models causal relations

[1]We formally introduce spin glass models in Chapter 2

between a set of diseases and findings. There have been many attempts to develop algorithms that perform approximate inference on this network, including loopy belief propagation. We use our homotopy continuation algorithm to perform inference in this problem. Our results show that this algorithm always converges and obtains exceptional results on this task.

1.3 Thesis Overview

The outline below specifies the organization of this thesis:

- Chapter 2: Background material on graph theory, graphical models, and probabilistic inference.
- Chapter 3: Introduction to the loopy belief propagation algorithms.
- Chapter 4: Convex combination belief propagation for pairwise models
- Chapter 5: Convex combination belief propagation for factor graphs
- Chapter 6: Belief propagation with homotopy continuation
- Chapter 7: Medical diagnostic inference with belief propagation and homotopy continuation

CHAPTER 2

Background

This chapter presents background material necessary for subsequent developments in this thesis. The main focus is to provide a self-contained introduction to graphical models and probabilistic inference. In Section 2.2, we provide a brief overview of graph theory and establish some basic notation. We broadly introduce graphical models in Section 2.3, and also describe a few special types. In Section 2.4, we present the inference task and discuss some important approaches to this problem. Lastly, we describe some relevant applications in Sections 2.5, which also provides context for the inference problem.

2.1 Introduction

Graphical models are a well-established probabilistic model due to their ability to express complex relationships between variables in a system. The main components of this model are a probability distribution and a graph representing conditional independences (Markov properties) of the system. The graph provides both a global representation of the structure of the system and reflects local dependencies among variables. Graphical models are not only a natural model for complex systems, they also provide a computational framework that is supported by a suite of algorithms. Many of these algorithms are derived by utilizing the graphical structure of the model to simplify certain calculations.

The origins of graphical models can be traced back to independent work in statistical physics, genetics, and probability theory. One of the earliest uses dates back to Gibbs in the late 1800s who used this framework to model physical systems of interacting particles [30]. He used a lattice to model the global structure of these systems, where neighboring nodes represent interacting particles. Later in the 1920s, Wright created an approach called “path analysis” which uses directed graphs to model how genetic traits are generated in a population [84]. His analysis uses causal modeling to express dependencies between variables. Markov developed a theory of stochastic processes in which complex systems can be explained via a simple chain of dependencies [59]. His work established important notions about the relationship between conditional independence and graphical structures which would later become part of the theoretical foundation of graphical models.

It wasn’t until the 1980s that graphical models finally became mainstream. This development was largely driven by research in artificial intelligence to design intelligent systems that rely on probabilistic reasoning. The theoretical foundation was pioneered by Judea Pearl who introduced mathematical tools for modeling causal relationships and performing probabilistic inference (see [64], [65]). Over the course of 100 years, this general framework has found applications in a wide variety of disciplines including bioinformatics, neuroscience, computer vision, statistical physics, economics, and computer science (see, e.g., [10], [22], [27], [47], [61]). Graphical models have become an important area of research as the need to model complex information continues to grow.

2.2 Graph Theory

A graph $G = (V, E)$ consists of a set vertices (or nodes) $V = \{1, \dots, n\}$ and a set of edges E which connect pairs of vertices. The edge set consists of either directed or undirected edges. Let $\{i, j\} \in E$ denote an undirected edge, while $(i, j) \in E$ denotes a directed edge. In this thesis, we mostly focuses on graphical models which involve undirected graphs. Thus, assume that the graph G is undirected for the remainder of this section. In addition, we also assume that the graph does not contain self loops (i.e. $\{i, i\} \notin E$).

Let $\mathcal{N}(i) = \{\, j : \{i, j\} \in E \,\}$ be the set of neighbors of node i. The degree $d(i)$ of node $i \in V$ is given by $d(i) = |\mathcal{N}(i)|$, where $|\cdot|$ denotes the cardinality of a set. A path between two nodes $i, j \in V$ is a sequence of distinct neighboring vertices that starts at i and ends at j. Let $E(i, j)$ be the set of all sequences of edges that form a path between nodes i and j. A graph is called connected if there exists a path between any pair of vertices. A cycle is a path from a node $i \in V$ back to itself, where the path consists of a sequence of distinct edges.

A clique is any fully connected subset of vertices (see Figure 2.1). A clique is maximal if it is not properly contained within any other clique. Let $C(G)$ be the set of all cliques in the graph G. The notion of a clique is especially useful in the study of graphical models because the structure of the distribution is closely related to the set $C(G)$.

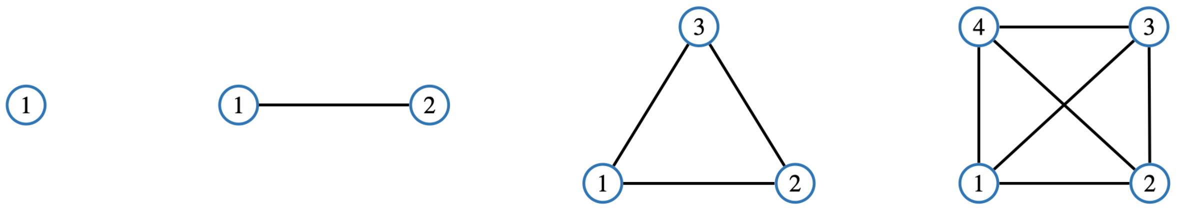


FIGURE 2.1: Cliques from size 1 to 4

Next, we define several graphs that are extensively used through out this thesis. There are several other types of graphs discussed in this thesis, including simple cycles, grid graphs, and random graphs. However, the graphs in Definitions 2.2.1-2.2.3 play a more central role throughout this work.

Definition 2.2.1. *A tree-structured graph is an undirected graph in which every pair of distinct vertices are connected by exactly one path.*

Definition 2.2.2. *A complete graph is an undirected graph in which every distinct pair of vertices is connected by an edge.*

Definition 2.2.3. *A bipartite graph is an undirected graph $G = (V \cup U, E)$ whose vertices can be divided into two disjoint sets (i.e. $V \cap U = \emptyset$). The edge set consists of pairs of vertices $\{i, j\} \in E$ such that $i \in V$ and $j \in U$.*

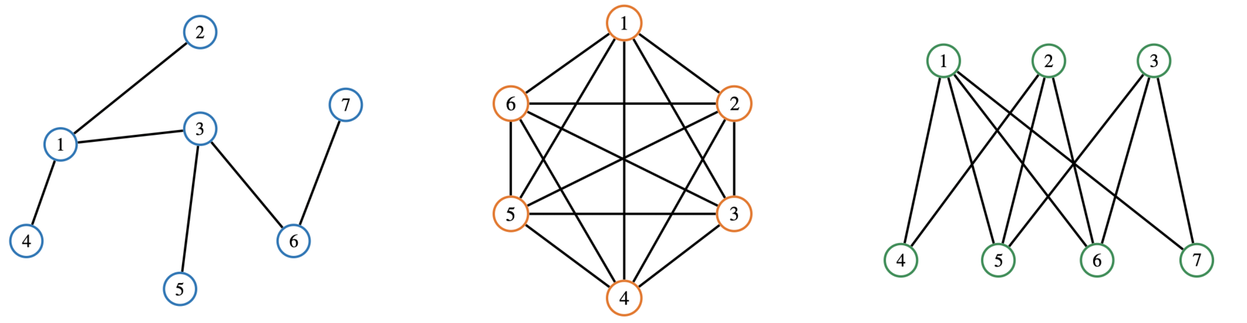


FIGURE 2.2: Here we see a tree-structured graph on the left, a complete graph in the center, and a bipartite graph on the right.

2.3 Graphical Models

Graphical models provide a computational framework for problems that are probabilistic in nature and have an underlying graphical structure. In this section, we see that another important component of this model is that the global properties are governed by local interactions. The joint distribution is written as a product of local terms (i.e. potentials) that correspond to cliques in the graph. Each potential models interactions between variables corresponding to the clique. Given that this model was originally used to understand interacting particle systems, we use this as a canonical example to explain the intuition behind the rigorous definition.

Let $G = (V, E)$ be a graph with the vertex set $V = \{1, \ldots, n\}$, where each node represents a random variable in the system. Let $X = (X_1, \ldots, X_n)$ be a random vector such that each $X_i \in \Omega$ is a random variable defined over the set of possible outcomes $\Omega = \{1, \ldots, m\}$. Intuitively, each random variable represents a particle and the random vector represents the system. A configuration of the random vector is given by $x = (x_1, \ldots, x_n) \in \Omega^n$, where $x_i \in \Omega$ is the state of the i-th random variable. Let x_S denote a configuration of a subset of random variables such that $S \subset V$. The probability of a configuration of the random vector X is given by the joint distribution

$$\mathbb{P}(X = x) = \frac{1}{Z} \prod_{\mathcal{C} \in C} \psi_{\mathcal{C}}(x_{\mathcal{C}}), \tag{2.1}$$

where Z is a normalization constant and $C \subset C(G)$ is a set of cliques. Each $\psi_{\mathcal{C}} : \Omega^{|\mathcal{C}|} \to \mathbb{R}$ is a non-negative function referred to as a potential. The potentials encode dependencies among random variables and model local interactions between particles.

The expressive power of graphical models makes them a natural framework for modeling problems in computer vision, machine learning, bioinformatics, and robotics (see, e.g., [13], [24], [54], [56], [92]). Some applications have a unique structure which is better suited for certain types of graphical models. In this thesis, we discuss three distinct types of graphical models, namely Bayesian networks, Markov random fields, and factor graph. The joint distribution in these special types of graphical models has the same form as Equation 2.1, but the structure of the graph and cliques differ. The remainder of this section reviews the three types of graphical models discussed in this thesis.

2.3.1 Bayesian Networks

Bayesian networks gained popularity in the 80s and 90s as research in artificial intelligence shifted towards designing intelligent systems that integrate expert knowledge and common sense into a systematic framework. They became the standard model for building expert systems due to their natural ability to encode causal relationships. Pearl was very influential in this paradigm shift with his work on causality [65]. In this work, he identifies several different types of causal relationships which are most naturally represented with a directed graph (see Figure 2.3) [70].

Let $G = (V, E)$ be a directed graph such that the edge set is a collection of ordered pairs of vertices $(i, j) \in E$. In a directed graph, each node $i \in V$ has a set children nodes denoted by $Ch(i) = \{\, j \,:\, (i, j) \in E \,\}$ and parent nodes $Pa(i) = \{\, k \,:\, (k, i) \in E \,\}$.

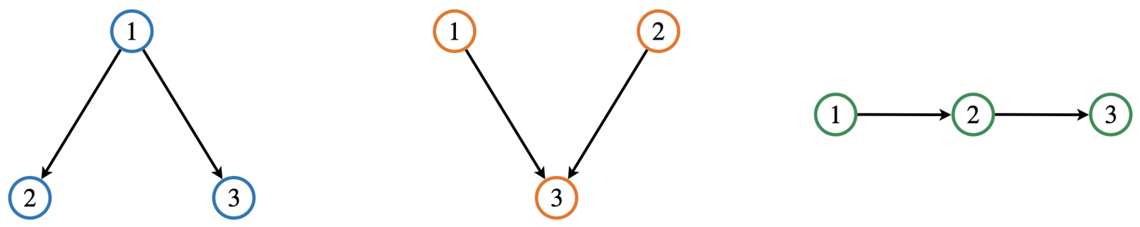


FIGURE 2.3: Fundamental types of causal relationships

Each node $i \in V$ in the graph represents a random variable X_i. A directed edge $(i, j) \in E$ indicates a dependency between the random variables X_i and X_j. In this type of model, dependencies have a causal structure in the sense that the state of X_i is dependent upon the states of its parents. Causal relationships between random variables are captured by conditional distributions, which act as the potentials in the model. The likelihood of a configuration of the random vector X is given by

$$\mathbb{P}(X = x) = \frac{1}{Z} \prod_{i \in V} \mathbb{P}\big(X_i = x_i | X_{Pa(i)}\big).$$

Next we provide a concrete example of a Bayesian network and more closely describe the relationship between the topology of the graph and structure of the joint distribution. Note that we also discuss an example of an expert system built upon a Bayesian network in Section 2.5.3.

Example 2.3.1. *Let $G = (V, E)$ be a directed graph with $V = \{1, \dots, 6\}$ as shown in Figure 2.4. Let $X = (X_1, \dots, X_6)$ be a random vector and let joint distribution be*

$$\mathbb{P}(X = x) = \frac{1}{Z} \mathbb{P}(X_1)\, \mathbb{P}(X_2|X_1)\, \mathbb{P}(X_3|X_2, X_5)\, \mathbb{P}(X_4|X_3)\, \mathbb{P}(X_5)\, \mathbb{P}(X_6|X_5).$$

Each random variable in the model corresponds to a conditional distribution. One common practice is to define a conditional probability table (CPT) which stores all of these probabilities.

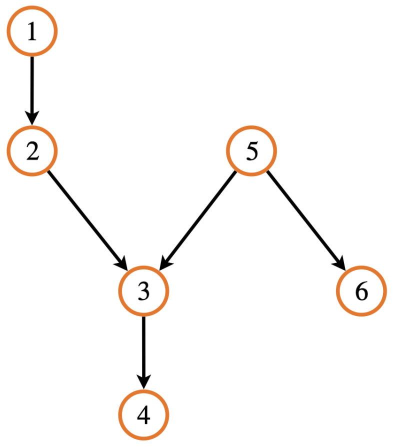


FIGURE 2.4: Directed graph from Example 2.3.1

2.3.2 Markov Random Fields

Markov random fields are the most popular type of graphical model and have been used in a wide variety of applications, especially computer vision and image analysis (see, e.g., [15], [57], [58], [91]). This type of graphical modes captures non-causal relationships between random variables on an undirected graph. This allows them to express certain dependencies that a Bayesian network cannot. In particular, Markov random fields are well-suited to expressing marginal dependence and conditional independence [23].

The joint distribution in a Markov random field has the same general form as Equation 2.1. One key difference is that the potentials must be strictly positive by Hammersly-Clifford Theorem [34]. One distinctive feature of this model is that it satisfies the local Markov property. This property describes a certain conditional independence property in terms of the topology of the graph. The main idea is that the conditional distribution of a random variable X_i given the states of all other variables is equivalent to the conditional distribution of X_i given its neighbors.

Definition 2.3.2. *A random vector X is a Markov random field with respect to the undirected graph $G = (V, E)$ if for all vertices $i \in V$,*

$$\mathbb{P}(X_i | X_{V \setminus i}) = \mathbb{P}(X_i | X_{\mathcal{N}(i)}).$$

Example 2.3.3. *Let $G = (V, E)$ be an undirected graph with $V = \{1, \ldots, 7\}$ as shown in Figure 2.5. Let $X = (X_1, \ldots, X_7)$ be a random vector and let joint distribution be*

$$\mathbb{P}(X = x) = \frac{1}{Z} \Psi_{12}(x_1, x_2)\, \Psi_{27}(x_2, x_7)\, \Psi_{2345}(x_2, x_3, x_4, x_5)\, \Psi_{456}(x_4, x_5, x_6).$$

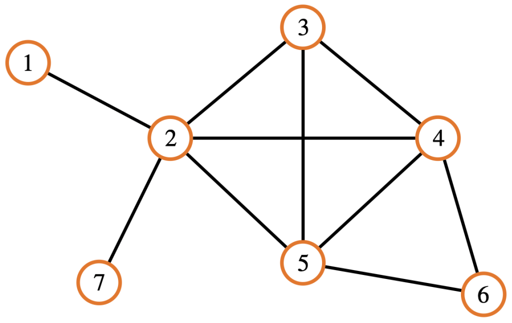


Figure 2.5: Undirected graph from Example 2.3.3

2.3.3 Energy-Based Models

Energy-based models are often used to model physical systems whose dynamics are governed by some underlying energy principles. The main components of this model are an undirected graph and energy defined with respect to the graph. The energy is a sum of cost functions (or potentials) that both model local interactions and incorporate prior information. In addition, the cost functions correspond to cliques in a graph, which makes this model closely related to Markov random fields. The discussion in this section describes energy-based models in the context of graph labelling.

Let $G = (V, E)$ be an undirected graph with the vertex set $V = \{1, \ldots, n\}$ and let $\Omega = \{1, \ldots, m\}$ be a set of labels. A labelling of the vertices in the graph is given by $x = (x_1 \ldots, x_n) \in \Omega^n$, where $x_i \in \Omega$ denotes the label of node $i \in V$. We use x_S to denote a labeling of a subset of nodes such that $S \subset V$. The cost (or energy) of a labeling x is given by the Gibbs energy

$$E(x) = \sum_{i \in V} g_i(x_i) + \sum_{\{i,j\} \in E} h_{ij}(x_i, x_j), \tag{2.2}$$

where g_i and h_{ij} are non-negative functions. The function $g_i : \Omega \to \mathbb{R}$ captures local information such as an observation of the label of a node i. The value of $g_i(x_i)$ is a cost for assigning label x_i to node i. The function $h_{ij} : \Omega \times \Omega \to \mathbb{R}$ captures pairwise relationships and influences neighboring nodes to have compatible labels.

Energy-based models provide a computational framework for performing either probabilistic or non-probabilistic inference. Non-probabilistic inference involves determining the labeling that minimizes the energy. Equivalently, one can define the joint distribution

$$\mathbb{P}(X = x) = \frac{1}{Z} e^{-E(x)}, \tag{2.3}$$

where probabilistic inference involves obtaining the labeling that maximizes the distribution. One important realization is that the probabilistic version of this model is a pairwise Markov random field,

$$\begin{aligned}
\mathbb{P}(X = x) &= \frac{1}{Z} e^{-E(x)} \\
&= \frac{1}{Z} \exp\Big(- \sum_{i \in V} g_i(x_i) - \sum_{\{i,j\} \in E} h_{ij}(x_i, x_j) \Big) \\
&= \frac{1}{Z} \prod_{i \in V} e^{-g_i(x_i)} \prod_{\{i,j\} \in E} e^{-h_{ij}(x_i, x_j)} \\
&= \frac{1}{Z} \prod_{i \in V} \phi_i(x_i) \prod_{\{i,j\} \in E} \psi_{ij}(x_i, x_j)
\end{aligned}$$

where $\phi_i(x_i) = \exp\big(- g_i(x_i) \big)$ and $\psi_{ij}(x_i, x_j) = \exp\big(- h_{ij}(x_i, x_j) \big)$. This is a useful observation because it can be leveraged to derive efficient algorithms that perform non-probabilistic inference in energy-based models.

2.3.4 Factor Graphs

Factor graphs are the most general class of graphical models since they explicitly represent arbitrary factorizations of distributions. This also makes them the most expressive graphical model since they incorporate higher order dependencies among variables. In this model, the joint distribution factors into a product of potential functions which encode statistical dependencies among arbitrary subsets of random variables. The structure of this distribution is represented by a bipartite graph referred to as a factor graph.

Let $G = (V \cup F, E)$ be an undirected bipartite graph with a set of variable nodes $V = \{1, \ldots, n\}$ and factor nodes $F = \{1, \ldots, k\}$. Variable nodes represent random variables, while factor nodes represent potentials. Given that the graph is bipartite, then $\mathcal{N}(i) \subset F$ when $i \in V$ and $\mathcal{N}(f) \subset V$ when $f \in F$. In this work, we use the convention that the letters i, j, k denote a variable node whereas f, g, h denote a factor node.

Each factor node $f \in F$ is connected to some subset of variable nodes (i.e. $\mathcal{N}(f) \subset V$). The factor captures statistical dependencies among these random variables, which are encoded in the potential function $\Psi_f : \Omega^{d(f)} \to \mathbb{R}$. Given that each factor corresponds to a potential, the joint distribution is written as a product over the factor nodes

$$\mathbb{P}(X = x) = \frac{1}{Z} \prod_{f \in F} \Psi_f(x_{\mathcal{N}(f)}).$$

Example 2.3.4. *Let $G = (V \cup F, E)$ be a factor graph with the variable nodes $V = \{1, 2, 3, 4, 5\}$ and factor nodes $F = \{a, b, c, d\}$ as shown in Figure 2.6. Let $X = (X_1, \ldots, X_5)$ be a random vector and let the joint distribution be*

$$\mathbb{P}(X = x) = \frac{1}{Z} \Psi_a(x_1)\Psi_b(x_2, x_3)\Psi_c(x_1, x_2, x_4)\Psi_d(x_2, x_4, x_5).$$

Each potential corresponds to a factor node and it is a function of the random variables corresponding to the neighbors of the factor node.

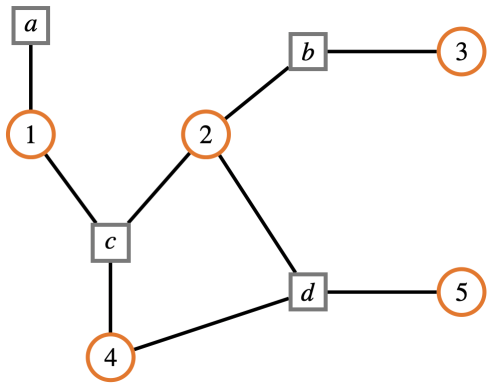


FIGURE 2.6: Factor graph from Example 2.3.4

Factor graphs are the most expressive type of graphical model and particularly well-suited for certain computations. Both Markov random fields and Bayesian networks can be easily transformed into an equivalent factor graph. In fact, this is a standard practice when performing probabilistic inference. Given a Markov random field, each clique is identified with a factor node. Then vertices belonging to the clique are connected to the factor node (see Figure 2.4).

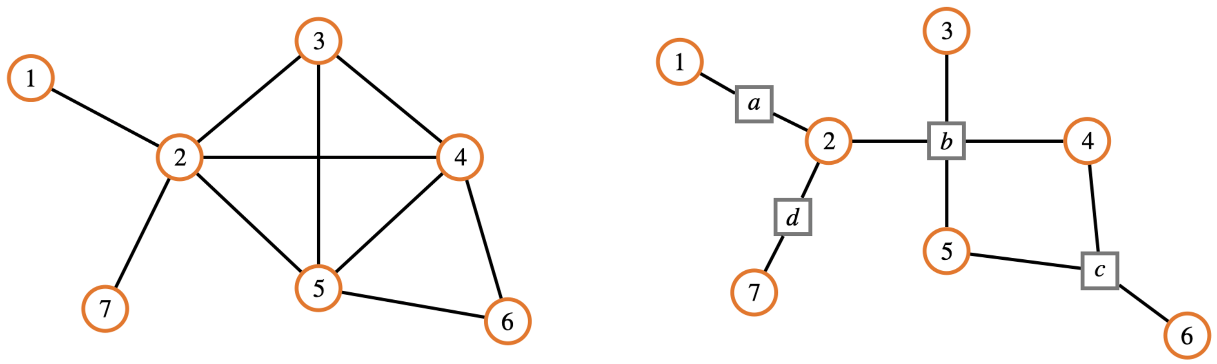


FIGURE 2.7: Factor graph representation of Markov random field from Example 2.3.3

Bayesian networks can also be easily transformed into a factor graph. Given a Bayesian network, let $i' \in F$ be a factor node which is identified as a copy of node $i \in V$. Then node i' is connected to i and the parents of this node (see Figure 2.8).

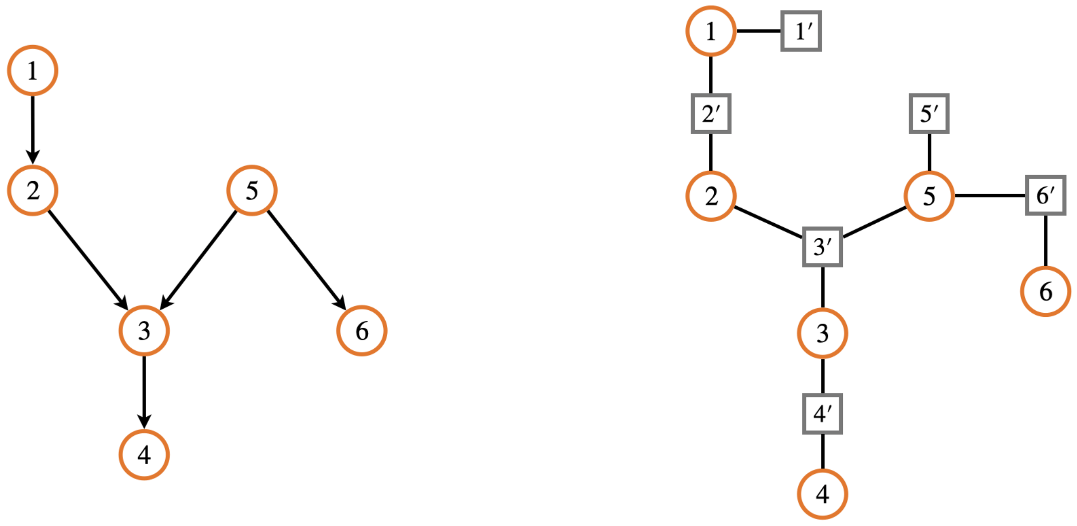


FIGURE 2.8: Factor graph representation of Bayesian network from Example 2.3.1

2.4 Probabilistic Inference

Probabilistic inference is a central computational challenge in many applications. Exact inference is generally intractable for arbitrary distributions, especially when the joint is defined over a large number of random variables. A major advantage of graphical models is that the structure of the joint distribution can be exploited to derive more efficient algorithms. As a result, graphical models also provide a computational framework which is supported by a suite of inference algorithms.

In this thesis, we focus on the two closely related inference tasks which frequently appear in applications. The first involves computing marginal distributions, while the second task is to determine the optimal configuration of the system.

1. *Marginal Inference.* Compute the marginal distribution of each random variable $i \in V$,

$$\mathbb{P}(X_i = \tau) = \sum_{x_{V \setminus i}} \mathbb{P}(X = x).$$

2. *Maximum a Posteriori (MAP) Inference.* Determine the globally optimal state of the random vector,

$$\hat{x}_{MAP} = \arg\max_{x \in \Omega^n} \mathbb{P}(X = x).$$

It is well-known that exact inference is NP-hard in both of these problems [14]. The reason being that the first involves summing over an exponential number of terms, while the latter involves optimizing over an exponential number of states. There are certain classes of graphical models in which these computations can be performed in polynomial time. For general models, it is essential to use an approximation algorithm.

Next we provide a literature review of important approaches to the inference problem. First, we describe several exact algorithms which are only applicable in certain settings. Then we discuss various algorithms that perform approximate inference which is a very active area of research. In Figure 2.9, we summarize the algorithms which are covered in this section.

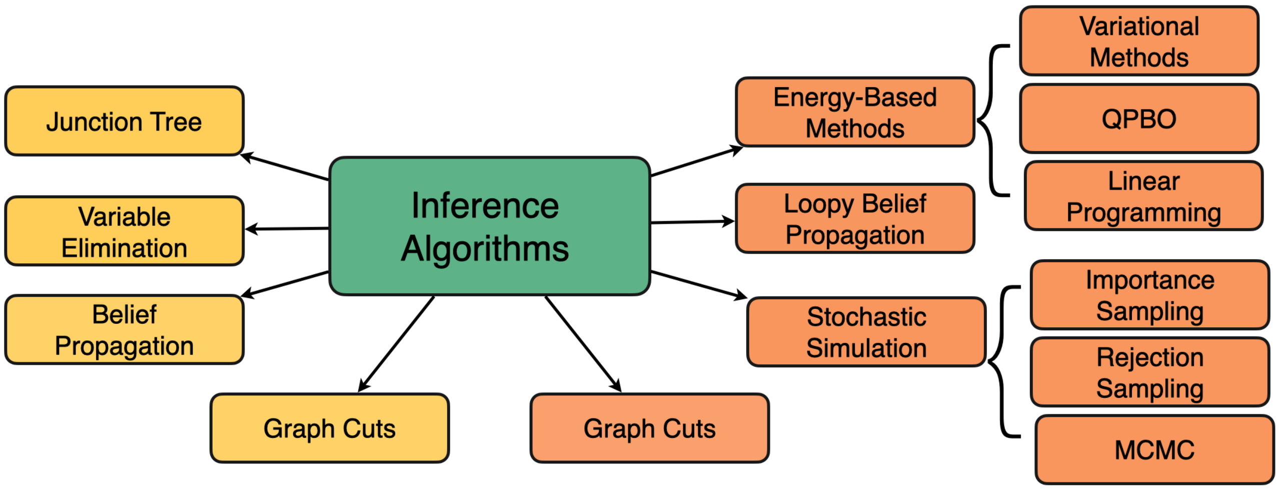


FIGURE 2.9: Inference algorithms. On the left, we see exact algorithms in yellow boxes. On the right, we see approximate inference algorithms in the orange boxes.

2.4.1 Exact Algorithms

The discussion in this section focuses on algorithms that perform exact marginal inference. There are analogous versions of these algorithms that compute the MAP solution. In general, this version can be easily recovered by replacing each “sum” with a “max”. Since these algorithms are so closely related, we simplify the discussion by focusing on marginal inference.

Exact Inference

The main idea behind these exact inference algorithms is to reduce the computational complexity by directly manipulating the joint distribution. As an example, let $X = (X_1, \dots, X_4)$ be a random vector with the joint distribution

$$\mathbb{P}(X = x) = \frac{1}{Z}\psi_1(x_1)\psi_1(x_1, x_2)\psi_2(x_2, x_3)\psi_3(x_3, x_4),$$

where the underlying graph is a chain with 4 nodes. The marginals are computed by fixing the value of one random variable, then summing over all possible states. The number of operations is exponential in the number of variables if this calculation is carried out naively. However, the computational complexity can be reduced to polynomial time,

$$\begin{aligned}\mathbb{P}(X_4 = \tau) &= \frac{1}{Z}\sum_{x_1}\sum_{x_2}\sum_{x_3}\psi_1(x_1)\psi_1(x_1,x_2)\psi_2(x_2,x_3)\psi_3(x_3,x_4)\\ &= \frac{1}{Z}\sum_{x_3}\psi_3(x_3,x_4)\sum_{x_2}\psi_2(x_2,x_3)\sum_{x_1}\psi_1(x_1)\psi_1(x_1,x_2).\end{aligned}$$

Exact inference methods are derived by leveraging this observation to improve the computational complexity.

Variable Elimination

The variable elimination algorithm performs exact inference by iteratively marginalizing (or eliminating) variables. The computation is carried by eliminating the innermost sum, then multiplying the result with the new innermost sum. This process is repeated until every variable has been marginalized. The main advantage of variable elimination is that it can be applied to arbitrary graphical models. In many cases, the algorithm can drastically reduce the computational complexity of exact inference. However, the run time is highly dependent upon the ordering of the sums. There are some efficient algorithms that find good orderings, but these methods are only applicable to low tree-width graphs. In general, determining the optimal visitation schedule is NP-hard [68].

Belief Propagation

Although belief propagation shares some similarities with variable elimination, this algorithm carries out the calculations in a more elegant manner. The main idea is to use dynamic programming to break up the sum into subproblems. This results in a local message passing algorithm, where information is propagated throughout the graph by sending messages between neighboring nodes. Each message incorporates local information from the potential and messages from neighboring nodes (excluding the neighbor receiving the message). It is well-known that belief propagation performs exact inference when the graph is tree-structured. This turns out to be a major limitation of the algorithm because many real work networks contain cycles (i.e. loopy graphs).

Junction Trees

The junction tree algorithm can be used to transform a loopy graph into a tree-structured graph. In a junction tree, each node corresponds to a subset of random variables as shown in Figure 2.10. The purpose of this construction is to eliminate cycles by clustering them into a single node. The details of this algorithm are a bit involved, but the main idea is to marginalize out certain variables within each cluster. If these cluster-level problems can be solved exactly, then belief propagation can be used to compute the exact marginals. This algorithm is very effective in certain settings, but it does not generalize well to arbitrary graphical models. One issue is that the complexity is highly dependent upon how the nodes are clustered. Although there are some algorithms that find good junction trees, the runtime has only been proven to be polynomial when the size of the largest clique is $\mathcal{O}(\log n)$ with n being the number of vertices. In general, determining the optimal junction tree is NP-hard [3].

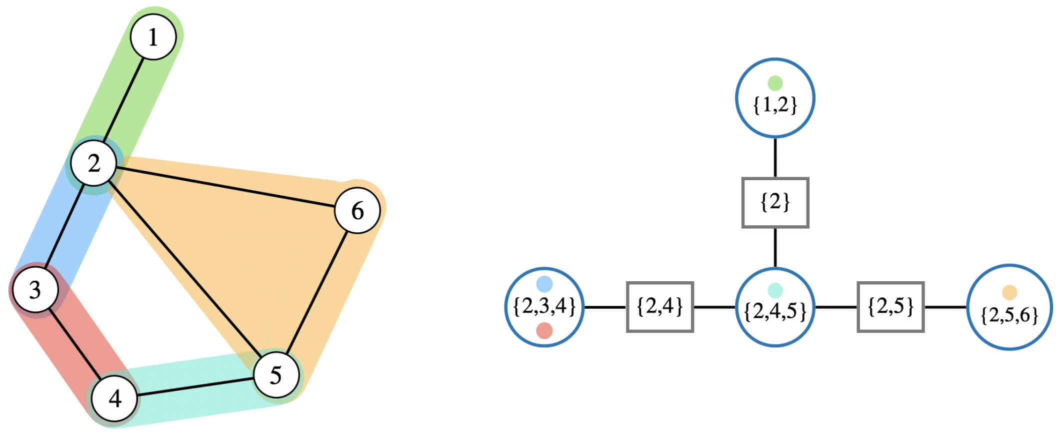


FIGURE 2.10: Here we see a graph on left and its junction tree on the right. In the junction tree, circles represent node clusters and rectangles are separating sets, which are sets of variables shared by neighboring clusters[1].

Graph Cuts

Graph cuts are a general class of algorithms that perform both exact and approximate MAP inference. The main idea behind this method is to define an energy-based model, then minimize the energy by solving a max flow problem. Although this algorithm is very popular in computer

[1]Note: this figure was originally created by Mark Paskin

vision, it is only guaranteed to perform exact inference in certain settings. Greig et al. (1989) showed that "binary" problems can be solved exactly with a single graph cut computation. Their algorithm involves maximizing the flow through an associated network after the introduction of a source and sink [32]. Ishikawa (2003) generalized this algorithm to first order Markov random fields with prior terms that are convex in terms of a linearly ordered label set [44].

2.4.2 Approximate Algorithms

Due to the computational complexity of performing exact inference, most efforts have been centered around developing approximation algorithms. There have been numerous approaches to this problem which broadly includes stochastic simulation, energy-based methods, and loopy belief propagation. In this section, we provide a brief description of each class of algorithms along with some advantages and limitations of each method.

Stochastic Simulation

Stochastic simulation is a rich area of research which has been very active since the 1940s. This class of algorithms uses that statistical properties of a large number of random samples are deterministic when certain conditions are met. Sampling directly from an arbitrary distribution is often intractable, instead a common practice is to sample from a simpler *proposal* distribution. This technique leads to an exact solution as long as the proposal satisfies certain conditions. One approach is to use rejection sampling where samples are generated from the proposal (e.g. uniform distribution), then rejected according to a probability which depends upon how well the proposal approximates the target distribution. One issue is that this algorithm tends to have a high rejection rate when the distribution is high dimensional, which makes the method very inefficient. Importance sampling is one alternative where samples are drawn from a proposal that is roughly proportional to the target, then samples are reweighed in a principled way. One improvement is to use *adaptive* importance sampling in which statistical properties of the samples are used to iteratively update the proposal [1]. This general approach tends to work better than rejection sampling and is more efficient in the sense that all samples are used. However, one limitation is that choosing a good proposal is not straight forward and can be very difficult in some cases. In addition, the rate of convergence is highly dependent upon the approximation quality of the proposal. In the context of graphical models, one disadvantage of sampling procedures is they are general purpose schemes that do no utilize the underlying graphical structure of the model.

Markov Chain Monte Carlo (MCMC) methods are a class of algorithms that similarly aim to generate samples from a target distribution. One important advantage of these algorithms is that they are better suited for sampling from high dimensional distributions. The main idea is to construct a Markov chain in which the stationary distribution is the target distribution. Then an approximation is obtained by using random walks to generate sequences of samples. As more samples are produced, the values more closely approximate the target distribution. In the Metropolis-Hastings algorithm, a current sample is used to generate a *candidate* sample which is accepted according to some probability. This probability is defined in such a way that samples are more likely to stay in high-density regions. Gibbs sampling is a special case of the Metropolis-Hastings algorithm in which samples are generated from the conditional distribution of each random variable [27]. This method is particularly well-suited for Markov random fields because each conditional is relatively simple due to the local Markov property. It can be shown that the sequence of samples constitutes a Markov chain, and the stationary distribution is exactly the target distribution [26]. Although these sampling algorithms are very popular, they are also notorious for being slow to converge. In addition, it can be difficult to determine how many iterations are necessary to obtain a good approximation.

Energy-Based Methods

Variational methods are another general class of algorithms that perform approximate inference [46]. The main idea is to reformulate inference as an optimization problem, then use gradient descent to obtain the approximation [41]. In order to make inference tractable, the approximation is restricted to a more manageable class of distributions. The objective is to minimize the KL-divergence over this class, which is equivalently done by maximizing a lower bound known as the *evidence lower bound* (ELBO). One of the main advantages of variational inference is that it scales better and is more amenable to techniques like parallelization and GPU optimization. Since most implementations rely on stochastic gradient descent, variational methods are susceptible to common pitfalls of this algorithm such as local minima, sensitivity to the initialization, and determining an optimal step size. Another challenge is that the approximation is dependent upon the chosen class of distributions. In general, more expressive families of distributions tend to result in more difficult optimization problems.

Graph cuts perform MAP inference by optimizing an objective function. There are certain problem instances where graph cuts compute the exact MAP solution, which were discussed in the previous section. In general, this class of algorithms is used to obtain an approximation of the MAP solution. One common approach is to initialize some solution, then iteratively perform *moves* that improve the quality of the solution. In the context of image analysis, a move refers to changing a pixel's label in order to decrease the energy. Boykov et al. (2001) use large moves referred to as α-expansions and $\alpha\beta$-swaps which simultaneously change the labels of arbitrarily large sets of pixels [9]. Their expansion algorithm finds a labeling within a known factor of the global minimum, while the swap algorithm handles more general energy functions.

Quadratic pseduo-Boolean optimization (QPBO) is a combinatorial optimization method which is closely related to graph cuts. When an energy is a sum of unary and binary cost functions which are submodular and defined with respect to binary variables, the MAP solution can be obtained with a graph cut optimization. This optimization problem is known to be NP-hard when the energy is not sub-modular. One alternative is to replace non-submodular terms with a submodular approximation, then use exact methods to obtain an approximation. However, this generally produces sub-optimal results, especially when there are a large number of non-submodular terms [53]. Instead a more favorable alternative is to use the QPBO algorithm. This algorithm involves building an extended graph and introducing a set of auxiliary variables, then the approximation is obtained from the min cut of this graph is computed with a max-flow algorithm (see [8],[7]).

Belief Propagation

Belief propagation was originally developed to perform exact inference in tree-structured graphs. However, there are many important problems in which the underlying graph contains cycles. In these applications, a common practice is to adapt the message passing scheme into a fixed point iteration algorithm called *loopy* belief propagation. This algorithm performs approximate inference and there are numerous examples of it obtaining good approximations (see [5],[25]). Loopy belief propagation tends to work well when the underlying graph is locally tree-structured. Since this algorithm relies on fixed point iteration, its susceptible to common issues associated with this numerical scheme. When the underlying graph has many cycles, the message passing operator tends to have multiple fixed points. The local dynamics of each fixed point are often unpredictable. Some fixed points may be attractive, while other are repelling which often causes the scheme to fail. Even if the algorithm does converge, the resulting solution depends on the initialization and may not provide a good approximation.

2.5 Applications

The main objective of this section is to introduce several applications of graphical models and provide context for the inference problem. The applications discussed in this section also play a central role throughout this thesis. In Section 2.5.1, we introduce the Ising model from statistical physics. Section 2.5.2 discusses how graphical models are used to perform image analysis. Lastly, we conclude this chapter by providing an example of a medical diagnostic network in Section 2.5.3.

2.5.1 Ising and Spin Glass Model

The Ising model is a mathematical model of ferromagnetism in statistical mechanics. The model consists of a large number of interacting particles that are spatially arranged in a lattice. Particles have an atomic spin that can be in one of two distinct states (i.e. up or down). The dynamics of this physical system are driven by local interactions between neighboring particles and an external magnetic field. The Ising model aims to understand ferromagnetic interactions in which neighboring particles prefer to be in the same state. These dynamics are often complicated by an external magnetic field which defines a preferred state for each particle. Given that the dynamics are driven by physical interactions, it is most natural to model this system with an energy-based model. In this setting, neighboring particles with compatible spins have a lower energy and the system tends to the lowest energy.

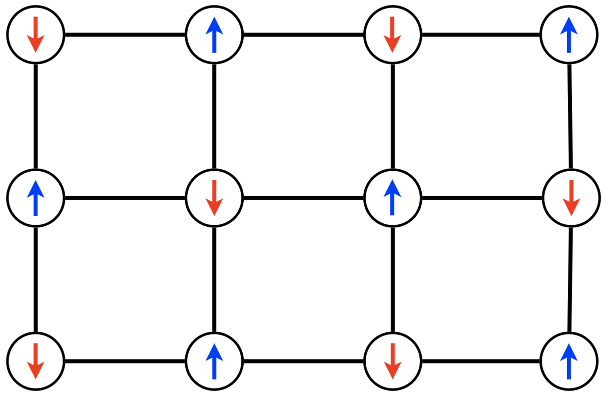


FIGURE 2.11: Simple example of a spin glass model that lacks an external field. Particles are spatially arranged in a 3×4 grid graph and the spin of each particle is denoted by an arrow. Note that neighboring particles have opposite spins.

Let $G_{n,m} = (V, E)$ be an $n \times m$ grid graph with undirected edges. Particles correspond to vertices and edges connect interacting particles. Let $\Omega = \{-1, 1\}$ be the set of possible orientations of a particle's spin. A configuration of the particle system is given by $x = (x_1 \ldots, x_n) \in \Omega^n$, where $x_i \in \Omega$ denotes the state of particle $i \in V$. The energy of a configuration of the system is given by

$$E(x) = -\sum_{i \in V} x_i y_i - \sum_{\{i,j\} \in E} \lambda_{ij} x_i x_j.$$

The configuration $y = (y_1, \ldots, y_n)$ is an external field, where y_i defines a local preference for each particle $i \in V$. The parameter λ_{ij} is a coupling factor which influences interactions between neighboring particles $i, j \in V$. The magnitude of this parameter controls how strongly neighboring states are coupled and the sign determines whether neighbors prefer to be aligned or misaligned. An interaction between two particles is ferromagnetic when $\lambda_{ij} > 0$ and anti-ferromagnetic when $\lambda_{ij} < 0$. In the Ising model, the coupling terms are assumed to be positive so that all interactions are ferromagnetic. There is a more complicated version of this model called a spin glass model, where the coupling factors may be either positive or negative.

An important computational problem is determining the configuration of the system that minimizes the energy. This problem can be equivalently stated as performing MAP inference since minimizing an energy is equivalent to maximizing the corresponding energy-based joint distribution. There are a few special cases in which this problem can be solved exactly. For example, the model was solved exactly in 1-dimension by Ernest Ising in the mid 1920s [66]. Then the

2-dimensional case was solved by Onlager in the mid 1940s [63]. Later, it was discovered that the n-dimensional case can also be solved exactly with graph cuts [2].

In this thesis, we use spin glass models in our experimental evaluation of comparing the performance of various belief propagation algorithms. Spin glass models are a good test case because positive and negative coupling factors instigate both attractive and repulsive dynamics. This often makes approximate inference with message passing very difficult. For this reason, this model is generally used as a test case for approximate inference algorithms.

2.5.2 Image Analysis

Markov random fields are a natural framework for image analysis due to their ability to encode spatial dependencies between pixels and enforce image priors that capture spatial coherence. Images have an inherent graphical structure which is captured by a grid graph. This model has been used in many image analysis tasks including image segmentation, depth estimation, and image restoration (see, e.g., [15], [57], [58], [91]). These applications generally use an energy-based model, where the cost functions are tailored to the particular application and enforce certain image priors. The task is then formulated as finding a labelling of the vertices (pixels) of the graph (image) that minimize the energy. The cost functions are defined in such a way that the optimal labelling provides a good solution to the imaging problem.

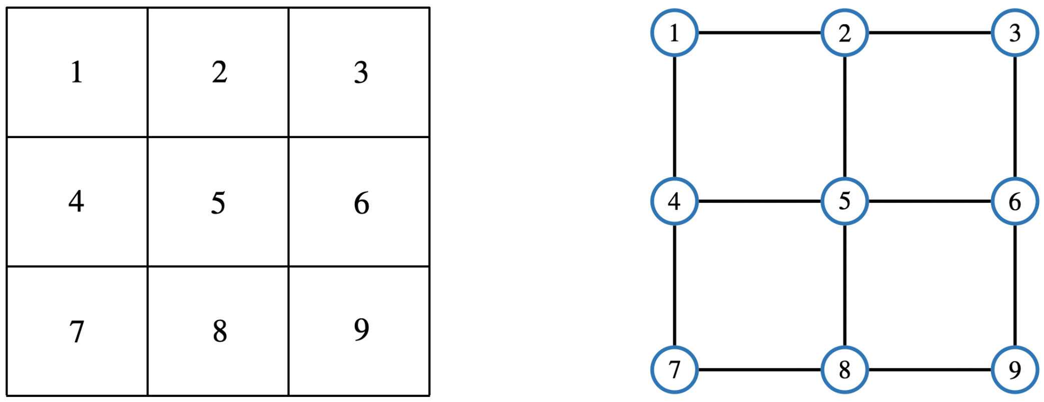


Figure 2.12: Here we see an image grid on the left and its corresponding grid graph on the right. Pixels in the image correspond to nodes in the graph and neighboring nodes correspond to adjacent pixels.

To formalize this approach, let $\mathcal{I}$ be an $n \times m$ image and let $G_{n,m} = (V, E)$ be an $n \times m$ grid graph that provides a graphical representation of the image. Each pixel corresponds to a node and the edges connect neighboring pixels as shown in Figure 2.12. Let $\Omega = \{\ell_1, \ldots, \ell_k\}$ be the set of possible labels, where the contents of this set depend upon the application. For example, binary labels are used in image segmentation and light intensity values from 0 to 255 are used in image restoration. Let $x = (x_1, \ldots, x_{nm}) \in \Omega^{nm}$ denote a labelling of the vertices of the graph and note that a labelling defines an image. The cost (or energy) of a labelling is given by the Gibbs energy

$$E(x) = \sum_{i \in V} g_i(x_i) + \lambda \sum_{\{i,j\} \in E} h_{ij}(x_i, x_j).$$

The unary cost function $g_i : \Omega \to \mathbb{R}$ is generally used to encode information over individual pixels. For example, this function may incorporate a noisy observation of a label or a likelihood that defines a local preference for certain labels. The pairwise function $h_{ij} : \Omega \times \Omega \to \mathbb{R}$ enforces that the optimal labelling is smooth and consists spatially coherent regions. In practice, this function is usually either the ℓ^1- or ℓ^2-distance.

In applications, the main computational objective is to perform MAP inference which involves determining a labelling of the vertices that minimizes the Gibbs energy. Exact inference is computationally intractable because images often have a large number of pixels and the set of labels may be large. Thus, it is essential to use approximate inference algorithms in these applications.

2.5.3 Medical Diagnostic Networks

Diagnostic inference is both a difficult and important problem which has attracted much attention in the last few decades. The objective is to design an intelligent system that assists in diagnosing a patient who exhibits some observed findings (or symptoms). Bayesian networks are a natural model for this task due to their ability to encode causal relationships into a systematic framework. The Quick Medical Reference (QMR) network is an example of a Bayesian network that models causal relationships between diseases and findings. This system uses a bipartite graph where diseases point to findings, which reflects the causal relationship between these variables. Each disease and finding pair has a corresponding conditional probability that represents the probability that a disease causes a finding to be present.

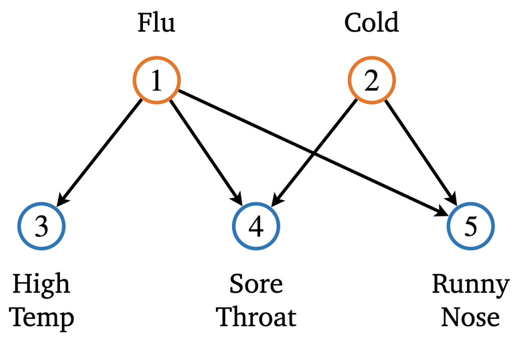


Figure 2.13: Simple example of a QMR network.

The QMR network is a directed bipartite graph $\mathcal{G} = (\mathcal{D} \cup \mathcal{F}, \mathcal{E})$, where $\mathcal{D} = \{1, \dots, n\}$ is a set of disease nodes and $\mathcal{F} = \{n+1, \dots, m\}$ is a set of finding nodes. The set $\mathcal{E}$ is a collection of ordered pairs of vertices $(i, j) \in \mathcal{E}$ such that $i \in \mathcal{D}$ and $j \in \mathcal{F}$. Each pair is a directed edge which models that findings are dependent upon diseases. Let $D = (D_1, \dots, D_n)$ and $F = (F_{n+1}, \dots, F_m)$ be random vectors that represent the set of disease and finding nodes, respectively. Let $\Omega = \{0, 1\}$ be the set of outcomes of each random variable. A configuration of the random vector (D, F) is given by $(d, f) = (d_1, \dots, d_n, f_{n+1}, \dots, f_m) \in \Omega^m$. The likelihood of a configuration is given by the joint distribution

$$\mathbb{P}(F = f, D = d) = \prod_i \mathbb{P}(F_i = f_i |\, \mathrm{Pa}(i)) \prod_j \mathbb{P}(D_i = d_j).$$

There are a number of underlying assumptions in this model that allow us to simplify the joint distribution. We return to this topic in Chapter 7, which provides a detailed overview of the QMR network.

The inference task is to estimate the conditional probability of a individual disease being present given an observation of findings. The outcome of this calculation provides a probabilistic explanation of the cause behind the observed findings. It has been proven that the computational complexity of exact inference is NP-hard [16]. Thus, it is essential to use approximate inference algorithms in these applications.

CHAPTER 3

Belief Propagation Algorithms

This chapter presents an introduction to the traditional belief propagation algorithms. There are two closely related variants of this algorithm which correspond to distinct inference tasks. The focus of Section 3.1 is to introduce the max-product algorithm which performs MAP inference, then Section 3.2 describes an equivalent version of this algorithm called the min-sum algorithm. In Section 3.3, we introduce the sum-product algorithm which performs marginal inference. Lastly, we provide a concise overview of recent developments in the belief propagation literature in Section 3.4.

3.1 Max-Product Algorithm

In this section, we present the max-product algorithm which performs MAP inference in a graphical model. Our objective is to both describe the intuition behind the algorithm, while also communicating its technical aspects. We discuss the MAP inference problem in Section 3.1.1, then provide a simple example in Section 3.1.2 that explains how and why the max-product algorithm works. In Sections 3.1.3 and 3.1.4, we present the max-product algorithm along with some discussion of its limitations.

3.1.1 Exact MAP Inference

Exact MAP inference in graphical models is computationally intractable because this problem is NP-hard. However, when the underlying graph is tree-structured, it is possible to derive an efficient algorithm that performs exact inference in polynomial time. In this special case, the max-product algorithm leverages the structure of both the joint distribution and underlying graph to break up the calculation into subproblems. Then the solution is efficiently computed by recursively solving the subproblems with dynamic programming.

The max-product algorithm computes *max-marginals* of the joint distribution, then obtains the MAP solution by maximizing each individual function. The max-marginal of node $i \in V$ is given by

$$p_i(x_i) = \max_{x_{V \setminus i}} \mathbb{P}(X = x). \tag{3.1}$$

The value $p_i(x_i)$ specifies the maximum probability of a configuration of the random vector when the state of the i-th random variable is x_i. As long as there are no ties, the exact MAP solution $\hat{x}_{\text{MAP}} := \arg\max_{x} \mathbb{P}(X = x)$ can be obtained by computing

$$\hat{x}_i = \arg\max_{\tau \in \Omega} p_i(\tau) \qquad \forall i \in V. \tag{3.2}$$

3.1.2 Simple Derivation of Messages

Let $G = (V, E)$ be the undirected graph in Figure 3.1 and let $V = \{1, \ldots, 5\}$ be the set of vertices. Let $X = (X_1, \ldots, X_5)$ be a random vector and $\Omega = \{1, \ldots, N\}$ be the set of possible outcomes of each random variable. Let the joint distribution of this Markov random field be

$$\mathbb{P}(X = x) = \frac{1}{Z} \prod_{\{i,j\} \in E} \psi_{ij}(x_i, x_j).$$

where Z is a normalization constant.

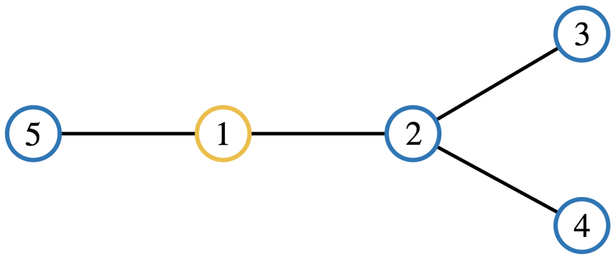


FIGURE 3.1: Simple tree-structured graph. Node 1 is highlighted to emphasize that the main focus is to compute the max-marginal of this node.

The purpose of this simple example is to explain the connection between dynamic programming and message passing. Since the algorithms are derived by analyzing the structure of the calculation, we proceed in this manner by computing the max-marginal of node 1.

$$\begin{aligned} p_1(x_1) &= \max_{x_2} \max_{x_3} \max_{x_4} \max_{x_5} \Big\{ \psi_{12}(x_1, x_2)\, \psi_{15}(x_1, x_5)\, \psi_{23}(x_2, x_3)\, \psi_{24}(x_2, x_4) \Big\} \\ &= \max_{x_5} \Big\{ \psi_{15}(x_1, x_5) \max_{x_2} \Big\{ \psi_{12}(x_1, x_2) \max_{x_3} \Big\{ \psi_{23}(x_2, x_3) \max_{x_4} \psi_{24}(x_2, x_4) \Big\}\Big\}\Big\}. \end{aligned}$$

Since x_1 is a constant and the max over x_5 is disjoint from the rest, this expression can be simplified as a *product* of *maxes*.

$$= \Big(\max_{x_5} \psi_{15}(x_1, x_5) \Big) \Big(\max_{x_2} \Big\{ \psi_{12}(x_1, x_2) \max_{x_3} \Big\{ \psi_{23}(x_2, x_3) \max_{x_4} \psi_{24}(x_2, x_4) \Big\}\Big\}\Big)$$

Similarly, the maximizations over x_3 and x_4 are also disjoint, so the expression within the max over x_2 can also be simplified as a *product* of *maxes*.

$$= \Big(\underbrace{\max_{x_5} \psi_{15}(x_1, x_5)}_{m_{51}(x_1)} \Big) \Big(\underbrace{\max_{x_2} \Big\{ \psi_{12}(x_1, x_2) \Big(\underbrace{\max_{x_3} \psi_{23}(x_2, x_3)}_{m_{32}(x_2)} \Big) \Big(\underbrace{\max_{x_4} \psi_{24}(x_2, x_4)}_{m_{42}(x_2)} \Big) \Big\}}_{m_{21}(x_1)} \Big) \tag{3.3}$$

Computing the max-marginals can be realized as a succession of subproblems $m_{ij}(x_j)$ which can be efficiently solved with dynamic programming. This approach drastically reduces the computational complexity from $\mathcal{O}(N^5)$ if the calculation is carried out naively to $\mathcal{O}(N^2)$.

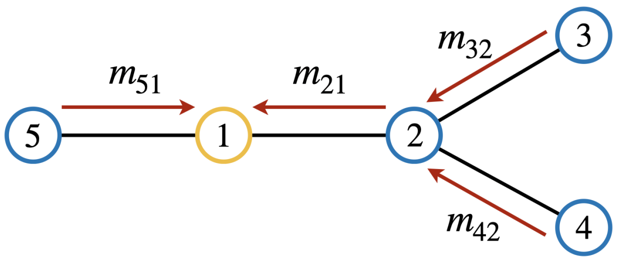


FIGURE 3.2: Local message passing on a graph. Here we emphasize the messages that are used to compute the max-marginal of node 1.

The subproblems are derived by exploiting that some maximizations are disjoint and can be simplified as a *product* of *maxes*. The *max-product* algorithm leverages this pattern to simplify the calculation as a succession of recursive subproblems. In the context of belief propagation, each subproblem is referred to as a *message*. Messages are sent between neighboring nodes, where $m_{ij}(x_i)$ denotes the message sent from node i to j.

The max-marginal of node 1 can be efficiently computed with local message passing as shown in Figure 3.2. The first step is to compute messages sent from the leaves of the graph rooted at node 1. Once a node receives a message, it sends messages to its other neighbors. The process repeats until node 1 receives messages from all of its neighbors. Although this procedure efficiently computes the max-marginal of node 1, the max-marginals of every node must be computed to obtain the MAP solution. If we repeat the same derivation for every max-marginal, some of the exact same messages reappear. Thus, the max-marginals can be more efficiently obtained by computing all messages sent between neighboring nodes in parallel.

3.1.3 Markov Random Fields

Let $X = (X_1, \ldots, X_n)$ be a random vector and $\Omega = \{1, \ldots, N\}$ be the set of possible outcomes for each random variable. Let $G = (V, E)$ be an undirected graph with the set of vertices $V = \{1, \ldots, n\}$, where each node represents a random variable. In a pairwise Markov random field, the joint distribution is

$$\mathbb{P}(X = x) = \frac{1}{Z} \prod_{i \in V} \phi_i(x_i) \prod_{\{i,j\} \in E} \psi_{ij}(x_i, x_j),$$

The unary potential $\phi_i : \Omega \to \mathbb{R}$ captures local information such as an observation of the random variable X_i. The binary potential $\psi_{ij} : \Omega \times \Omega \to \mathbb{R}$ encodes a statistical dependency between the random variables X_i and X_j. Our initial discussion of the max-product algorithm focuses on *pairwise* models, where the joint distribution is comprised of unary and binary potentials.

MAP inference is a central computational challenge in many applications involving graphical models. When the graph is tree-structured, the max-product algorithm computes the max-marginals for each random variable in polynomial time. The main idea behind this algorithm is to use dynamic programming to break up the calculation into subproblems. This results in a local message passing algorithm, where information is propagated throughout the graph via messages passing between neighboring nodes as illustrated in Figure 3.3.

Let $m_{ij} \in \mathbb{R}^N$ be the message sent from node i to j. For arbitrary tree-structured graphs, messages are computed with the max-product equations

$$m_{ij}(x_j) = \max_{x_i} \Big\{ \phi_i(x_i)\, \psi_{ij}(x_i, x_j) \prod_{k \in \mathcal{N}(i) \setminus j} m_{ki}(x_i) \Big\}.$$

This equation is a generalization of the messages from Equation (3.3) in the previous section. Each message incorporates local information from the potentials along with messages received from the other neighbors. Intuitively, the message m_{ij} provides information regarding what state node i thinks X_j should be in, where large values of $m_{ij}(x_j)$ correspond to favorable states. The messages are often normalized for numerical reasons, in which case m_{ij} is a distribution over the set of possible states.

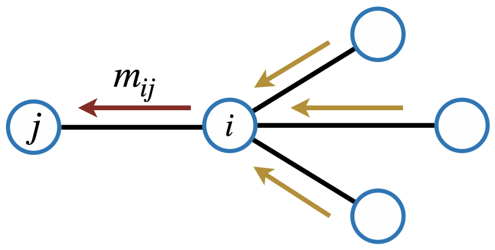


FIGURE 3.3: Illustration of message passing on a graph. We see that the message sent from the node i to j aggregates messages from all other neighbors of i.

After a node receives a message from each of its neighbors, the incoming messages are combined to compute belief functions for each node. Let $b_j : \Omega \to [0,1]$ be the belief function of node j such that

$$b_j(x_j) = \phi_j(x_j) \prod_{i \in \mathcal{N}(j)} m_{ij}(x_j). \tag{3.4}$$

A classical result in the belief propagation literature is that b_j is the exact max-marginal of node $j \in V$ when the graph is tree-structured [64]. In this case, the MAP solution can be obtained by maximizing each belief function individually.

When the graph contains cycles, this algorithm can be adapted into a fixed point iteration scheme referred to as *loopy* belief propagation. In this setting, the message passing equations are used to define an operator that facilitates message passing between all neighboring nodes in the graph. The message passing operator acts on the product space

$$\mathcal{M} := \bigotimes_{i \in V} \bigotimes_{j \in \mathcal{N}(i)} \mathbb{R}_+^{|\Omega|}$$

Let $m \in \mathcal{M}$ be a vector which consists of all messages sent between neighboring nodes in the graph.

Definition 3.1.1. *The message passing operator $P : \mathcal{M} \to \mathcal{M}$ in the max-product algorithm is $P = NQ$. Q computes new messages using the max-product equation and N normalizes each message to sum to one.*

$$\begin{aligned} \big(Qm\big)_{ij}(x_j) &= \max_{x_i} \Big\{ \phi_i(x_i)\, \psi_{ij}(x_i, x_j) \prod_{k \in \mathcal{N}(i) \setminus j} m_{ki}(x_i) \Big\} \\ \big(Nm\big)_{ij}(x_j) &= \frac{m_{ij}(x_j)}{\sum\limits_{\tau \in \Omega} m_{ij}(\tau)} \end{aligned}$$

Note that the message updates are normalized to prevent numerical overflow. Since the messages and beliefs are only scaled by a constant, normalizing simply scales the resulting beliefs. In loopy belief propagation, the messages are initialized as $m^{(0)} \in \mathcal{M}$ and repeatedly updated by using the fixed point iteration scheme:

$$m^{(\ell+1)} = Pm^{(\ell)},$$

where $\ell \in \mathbb{N}$ is the number of iterations. In practice, this scheme is either run for a large number of iterations or stopped once the messages have sufficiently converged. The resulting messages $m^{(\ell)}$ are then used to compute beliefs $b_j^{(\ell)}$ for every node.

Definition 3.1.2. *The belief $b_j^{(\ell)} : \Omega \to [0,1]$ of node $j \in V$ after ℓ iterations is given by*

$$b_j^{(\ell)}(x_j) = \kappa\, \phi_j(x_j) \prod_{i \in \mathcal{N}(j)} m_{ij}^{(\ell)}(x_j), \tag{3.5}$$

where κ is a normalization constant.

The beliefs are an approximation of the exact max-marginals and can be used to obtain an approximate MAP solution $\tilde{x}_{MAP} \in \Omega^n$ such that

$$\tilde{x}_i := \arg\max_{x_i} b_i^{(\ell)}(x_i).$$

Belief propagation has gained popularity within the artificial intelligence community since it obtains a good approximation to the MAP solution in certain settings such as error correcting codes and image analysis (see, e.g., [11],[19],[21],[25],[74]). However, belief propagation is also notorious for failing to converge, being sensitive to the message initialization, and returning an inaccurate estimate of the MAP solution. These problems are the main motivations behind the developments in this thesis as well as many other works over the last few decades.

One general approach is to consider belief propagation as a discrete-time map, then utilize ideas from the theory of dynamical systems to understand the behavior of this algorithm. This is a natural approach to this problem because the messages are updated with the fixed point iteration scheme:

$$m^{(\ell)} := Pm^{(\ell)}$$

where $m^{(0)} \in \mathcal{M}$ is the message initialization. If the messages converge, they provide an approximation of a solution (i.e. fixed point) of the fixed point equation $Pm = m$. It is well-known that this equation always has at least one solution when the potentials are positive [39]. There are a number of works that have studied how the topology of the graph and potentials affect the number of solutions. In general, graphical models with strong couplings between random variables and graphs with more cycles tend to have more fixed points. We return to this topic in Section 3.4, where we go into more detail on the findings from the literature.

Although the fixed point equation $Pm = m$ is guaranteed to have at least one solution, it is often very difficult to obtain any solution with loopy belief propagation. The local dynamics of the fixed points are often unpredictable. Some fixed points may be attractive, while others are repelling which often causes the messages to oscillate. One numerical method that improves the performance of this algorithm is to stabilize the scheme by taking a convex combination of the messages from the previous and current iteration. In the belief propagation literature, this modification is referred to as *damped* belief propagation[1].

Definition 3.1.3. *The damping operator $P_\alpha : \mathcal{M} \to \mathcal{M}$ in the max-product algorithm is*

$$P_\alpha m = (1-\alpha)Pm + \alpha\, m,$$

where $\alpha \in (0,1)$ is the damping factor.

In practice, this scheme often prevents the messages from oscillating and converges to a fixed point faster than loopy belief propagation when both algorithms converge [69]. In Section 3.2, we provide an example where damping effectively prevents the messages from oscillating. However, the theoretical understanding of damped belief propagation is very limited. One general result is that this scheme converges when the operator is non-expansive and defined over a closed, bounded compact subset of a Hilbert space (see [4],[67]). However, the message passing operator in loopy belief propagation is generally expansive with a Lipschitz constant much greater than one. There are many problems where damping is not enough to prevent oscillations. In the next chapter, we provide an example of a graphical model where damped belief propagation does not converge for any damping factor that we tried.

[1] This general scheme is referred to as the Krasnoselskij-Mann iteration scheme in the numerical analysis literature.

3.1.4 Factor Graphs

Next we present the factor graph version of the max-product algorithm. The message passing scheme is similar, but one key difference is that both factor and variable nodes send messages. Although this complicates the analysis, most results on pairwise models extend to this more general case.

Let $G = (V \cup F, E)$ be an undirected bipartite graph with the set of *variable* nodes $V = \{1, \ldots, n\}$ and *factor* nodes $F = \{1, \ldots, m\}$. In this thesis, variable and factor nodes are concisely referred to as *variables* and *factors*, respectively. In addition, we use the convention that the letters i, j, k denote variables whereas f, g, h denote factors. Let $X = (X_1, \ldots, X_n)$ be a random vector and let $\Omega = \{1, \ldots, N\}$ be the set of possible outcomes of each random variable. Now consider the joint distribution

$$\mathbb{P}(X = x) = \frac{1}{Z} \prod_{f \in F} \Psi_f(x_{\mathcal{N}(f)}).$$

Let $\mu_{f \to i} \in \mathbb{R}^N$ be the message sent from factor f to variable i and let $\nu_{i \to f} \in \mathbb{R}^N$ be the message sent from i to f. Note that we use μ as opposed to m to distinguish between messages sent on factor graphs versus Markov random fields. When the factor graph is tree-structured, the messages are computed as

$$\begin{aligned}
\nu_{i \to f}(x_i) &= \prod_{g \in \mathcal{N}(i) \backslash f} \mu_{g \to i}(x_i) \\
\mu_{f \to i}(x_i) &= \max_{x_{\mathcal{N}(f)}} \Big\{ \Psi_f(x_{\mathcal{N}(f)}) \prod_{j \in \mathcal{N}(f) \backslash i} \nu_{j \to f}(x_j) \Big\}.
\end{aligned}$$

The semantic meaning of each message is slightly different in the case of a factor graph. Intuitively, the message $\mu_{f \to i}$ provides information regarding what state the other neighbors of factor f think the random variable X_i should be in, where large values of $\mu_{f \to i}(x_i)$ correspond to favorable states. The message $\nu_{i \to f}$ communicates what state node i thinks that the other neighbors of factor f should be in. The incoming messages at each variable are multiplied together to compute a belief function $b_i : \Omega \to \mathbb{R}_+$ given by

$$b_i(x_i) = \kappa \prod_{f \in \mathcal{N}(i)} \mu_{f \to i}(x_i) \quad \forall i \in V,$$

where κ is a normalization constant. When the factor graph is tree-structured, the beliefs are the exact max-marginals.

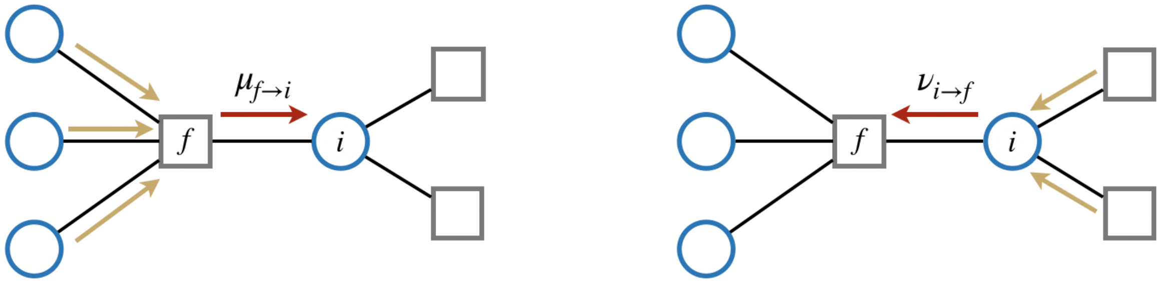


FIGURE 3.4: Illustration of message passing on a factor graph. Circles represent variable nodes and squares represent factor nodes. On the left, we see that the message sent from f to i aggregates messages from all other neighbors of f. On the right, we see that the message sent from i to f also aggregates messages from the other neighbors.

When the graph contains cycles, this algorithm can be adapted into a fixed point iteration scheme. In this setting, the message passing equations are used to define operators that facilitate message passing between all neighboring nodes in the graph. The message passing operators act on the product space

$$\mathcal{M} := \bigotimes_{f \in F} \bigotimes_{i \in \mathcal{N}(f)} \mathbb{R}_+^N$$

Let $\mu \in \mathcal{M}$ be a vector which consists of all messages sent from factors to variables and let $\nu \in \mathcal{M}$ consist of all messages sent from variables to factors.

Definition 3.1.4. *The message passing operator $T : \mathcal{M} \to \mathcal{M}$ in the max-product algorithm that computes messages sent from variables to factors is*

$$(T\mu)_{i \to f}(x_i) = \prod_{g \in \mathcal{N}(i) \setminus f} \mu_{g \to i}(x_i).$$

Definition 3.1.5. *The message passing operator $P : \mathcal{M} \to \mathcal{M}$ in the max-product algorithm that computes messages sent from factors to variables is $P = NQ$. Q computes new messages using the sum-product equation and N normalizes the messages.*

$$
\begin{aligned}
\big(Q\nu\big)_{f\to i}(x_i) &= \max_{x_{\mathcal{N}(f)}} \Psi_f(x_{\mathcal{N}(f)})\Big\{\prod_{j\in\mathcal{N}(f)\setminus i} \nu_{j\to f}(x_j)\Big\}\\
\big(N\nu\big)_{f\to i}(x_i) &= \frac{\nu_{i\to f}(x_i)}{\sum\limits_{\tau\in\Omega} \nu_{i\to f}(\tau)}.
\end{aligned}
$$

Definition 3.1.6. *The message passing operator $D : \mathcal{M}\times\mathcal{M} \to \mathcal{M}\times\mathcal{M}$ updates all messages in parallel,*

$$D(\mu, \nu) = (P\nu,\, T\mu).$$

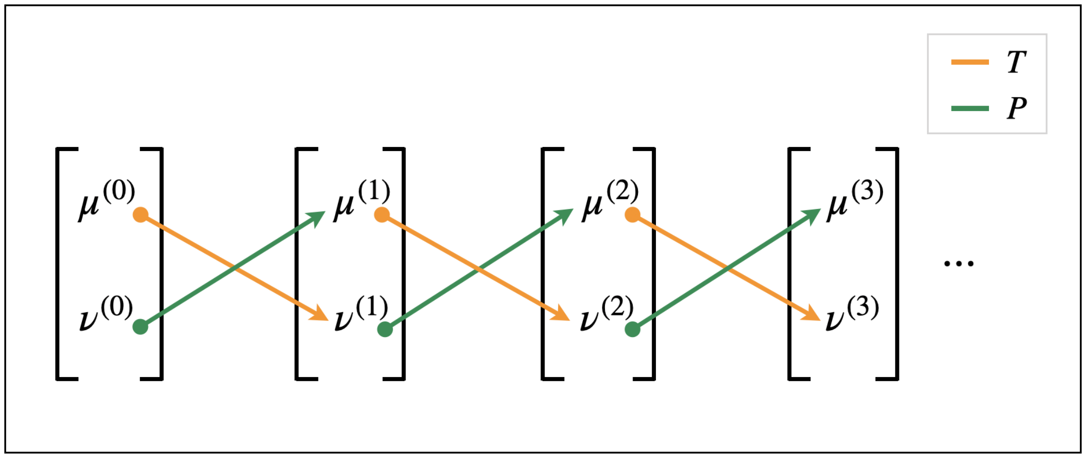


Figure 3.5: Illustration of message updates in loopy belief propagation. Column vectors show the two sets of messages computed on each iteration. Arrows represent either T or P, depending on the color. The initial point on an arrow indicates the input to the respective operator and the end points to the output.

In loopy belief propagation, the messages are initialized with $\mu^{(0)}, \nu^{(0)} \in \mathcal{M}$, then repeatedly updated via the fixed point iteration scheme

$$\big(\mu^{(\ell+1)},\, \nu^{(\ell+1)}\big) := D(\mu^{(\ell)}, \nu^{(\ell)}),$$

where $\ell \in \mathbb{N}$ denotes the number of iterations. The algorithm terminates when the messages converge to a solution of the following system of nonlinear equations

$$\mu = P\nu$$
$$\nu = T\mu.$$

It is important to note that a solution to this system of equations can also be realized as a fixed point of the system $PT\mu = \mu$ (also note that $PT\mu^{(\ell+1)} = \mu^{(\ell)}$). This system of equations is guaranteed to have at least one solution when the potentials are strictly positive [86]. Once the algorithm converges, the messages are used to compute beliefs.

Definition 3.1.7. *Let $b_i^{(\ell)} : \Omega \to (0,1)$ be the belief of $i \in V$ be given by,*

$$b_i^{(\ell)}(x_i) = \kappa \prod_{f \in \mathcal{N}(i)} \mu_{f \to i}^{(\ell)}(x_i)$$

with κ being a normalization constant.

The factor graph version of the max-product algorithm suffers from the same problems discussed in the last section, namely multiple fixed points, instability, and sensitivity to the initialization of the messages. Similarly, damping is often used as an attempt to stabilize the message passing dynamics.

Definition 3.1.8. *The damping operator $D_\alpha : \mathcal{M} \times \mathcal{M} \to \mathcal{M} \times \mathcal{M}$ is*

$$D_\alpha(\mu, \nu) = \Big((1-\alpha)\nu + \alpha T\mu,\ (1-\alpha)\mu + \alpha P\nu \Big)$$

where $\alpha \in (0,1)$ is the damping factor.

3.2 Min-Sum Algorithm

Next we present the min-sum algorithm which also performs MAP inference in graphical models. MAP inference is an interesting problem which differentiates itself from marginal inference by having an equivalent, alternative form. MAP inference can be performed by either maximizing the joint distribution or minimizing a closely related energy function. This alternative form is more natural for certain applications and also provides additional insight on the behavior of belief propagation.

3.2.1 Energy-Based MAP Inference

Let $E(x) = -\log \mathbb{P}(X = x)$ be an energy function. The energy-based formulation of the MAP inference problem is often preferred because it is less sensitive to numerical artifacts.

The minimization problem can be solved in an analogous manner by computing *min*-marginals, then minimizing each individual function. The min-marginal of node $i \in V$ is given by

$$q_i(x_i) = \min_{x_{V\setminus i}} E(x).$$

The value $q_i(x_i)$ specifies the optimal cost of a labeling where the i-th node is labelled as x_i. As long as there are no ties, the MAP solution $\hat{x}_{MAP} \in \Omega^n$ can be obtained as

$$\hat{x}_i = \arg\min_{\tau \in \Omega} q_i(\tau) \qquad \forall i \in V.$$

Computing exact min-marginals in an energy-based model is also NP-hard. However, when the graph is tree-structured, the min-marginals can be computed efficiently by exploiting the relationship between energy-based models and Markov random fields. One corollary of this relationship is that the min-marginals can be written in terms of the corresponding max-marginals,

$$\begin{aligned} q_i(x_i) &= \min_{x_{V\setminus i}} E(x) \\ &= \min_{x_{V\setminus i}} \log \mathbb{P}(X = x) \\ &= -\log \max_{x_{V\setminus i}} \mathbb{P}(X = x) \\ &= -\log p_i(x_i) \end{aligned}$$

where p_i is the max-marginal of node $i \in V$. Note that the normalization constant in the distribution can be ignored since it does not affect the MAP solution. This relationship implies that the max-product algorithm can be adapted into an equivalent algorithm that computes min-marginals.

3.2.2 Message Passing Algorithm

The min-sum algorithm is a local message passing algorithm that computes exact min-marginals when the underlying graph is tree-structured. In this algorithm, messages are computed with the min-sum equation

$$m_{ij}(x_j) = \min_{x_i} \Big\{ g_i(x_i) + h_{ij}(x_i, x_j) + \sum_{k \in N(i) \setminus j} m_{ki}(x_i) \Big\},$$

which is the negative log version the max-product equations. Intuitively, the message m_{ij} provides information regarding what label node i thinks that its neighbor j should be given, where small values of $m_{ij}(x_j)$ correspond to favorable labels.

The incoming messages at each node are summed together to compute a *belief* function $b_j : \Omega \to \mathbb{R}$ given by

$$b_j(x_j) = g_j(x_j) + \sum_{i \in N(j)} m_{ij}(x_j).$$

For tree-structured graphs the definition of messages in terms of other messages leads to a unique solution that can be obtained by starting from the leaves of the graph. In this case, the resulting beliefs are exactly the min-marginals of the energy. Thus, the optimal labelling can be obtained by minimizing each belief function individually.

When the graph contains cycles, the min-sum equations can be used to define a message passing operator that updates all the messages sent between nodes in the graph in parallel (sequential update versions are also commonly used in practice).

Definition 3.2.1. *The message passing operator $Q : \mathcal{M} \to \mathcal{M}$ in the min-sum algorithm is $Q = NP$. P computes new messages using the min-sum equation and N centers each message so the resulting vectors have mean zero.*

$$\big(Pm\big)_{ij}(x_j) = \min_{x_i} \Big\{ g_i(x_i) + h_{ij}(x_i, x_j) + \sum_{k \in \mathcal{N}(i) \setminus j} m_{ki}(x_i) \Big\},$$

$$\left(Nm\right)_{ij}(x_j) = m_{ij}(x_j) - \frac{1}{|\Omega|}\sum_{\tau\in\Omega} m_{ij}(\tau).$$

Although the min-sum algorithm has been successfully used in a variety of applications, the algorithm is not guaranteed to converge. Similarly, damping is often used to stabilize the fixed point iteration scheme. Next we provide a simple example where damping prevents the messages from oscillating.

Example 3.2.2. *Let $G = (V, E)$ be a cycle with four nodes and let $\Omega = \{-1, 1\}$. Suppose that $y = (1, -1, 1, -1)$ is an "external field" that defines a local preference for the label of each vertex. Define the cost functions,*

$$g_i(x_i) = -y_i x_i \qquad \textit{and} \qquad h_{ij}(x_i, x_j) = -x_i x_j.$$

We applied both damped and undamped belief propagation to this example and show the resulting beliefs in Figure 3.6.

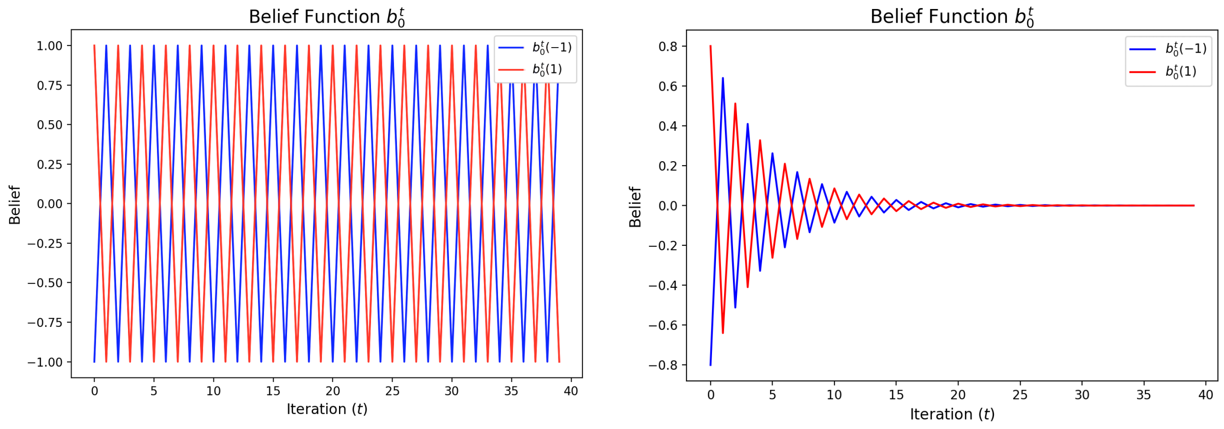


FIGURE 3.6: On the left, we see that the belief oscillates when non-damped belief propagation is applied to this example. On the right, we see that damped belief propagation stabilizes the oscillations and the belief converges.

In general, damped belief propagation appears to converge more often when compared to non-damped belief propagation, but the approach is still not guaranteed to converge. In the next chapter, we show an example where damped belief propagation does not converge for any damping factor that we tried.

3.3 Sum-Product Algorithm

The main objective of this section is to present the sum-product algorithm. In Section 3.3.1, we discuss the marginal inference problem, then provide a simple example that explains how and why the algorithm works. Sections 3.3.2 and 3.3.3 present the Markov random field and factor graph versions of the sum-product algorithm, respectively.

3.3.1 Marginal Inference

In marginal inference, the objective is to compute

$$\mathbb{P}(X_i = x_i) = \frac{1}{Z} \sum_{x_{V \setminus i}} \mathbb{P}(X = x).$$

This objective bears a close resemblance to MAP inference in the sense that they both involve a calculation over every configuration of the random vector. In fact, we can simplify the calculation in the exact same manner by exploiting the unique structure of the joint distribution.

Next we use the same graphical model from Section 3.1.2 to describe how and why the algorithm works. Since the algorithm is derived by analyzing the structure of the calculation, we proceed in this manner by computing the marginal of node 1.

$$\begin{aligned} \mathbb{P}(X_1 = x_1) &= \frac{1}{Z} \sum_{x_5} \sum_{x_4} \sum_{x_3} \sum_{x_2} \psi_{15}(x_1, x_5)\, \psi_{23}(x_2, x_3)\, \psi_{24}(x_2, x_4) \\ &= \frac{1}{Z} \sum_{x_5} \Big(\psi_{15}(x_1, x_5) \sum_{x_2} \Big(\psi_{12}(x_1, x_2) \sum_{x_3} \Big(\psi_{23}(x_2, x_3) \sum_{x_4} \psi_{24}(x_2, x_4) \Big) \Big) \Big), \end{aligned}$$

Several of these sums are disjoint and can be written as a *product* of *sums*,

$$= \frac{1}{Z} \underbrace{\Big(\sum_{x_5} \psi_{15}(x_1, x_5) \Big)}_{m_{51}(x_1)} \underbrace{\Big(\sum_{x_2} \psi_{12}(x_1, x_2) \underbrace{\Big(\sum_{x_3} \psi_{23}(x_2, x_3) \Big)}_{m_{32}(x_2)} \underbrace{\Big(\sum_{x_4} \psi_{24}(x_2, x_4) \Big)}_{m_{42}(x_2)} \Big)}_{m_{21}(x_1)}. \qquad (3.6)$$

Similar to the case of computing max-marginals, the calculation can be reduced to computing a series of messages $m_{ij}(x_j)$. The messages are derived by exploiting that some sums are disjoint and can be simplified as a *product* of *sums*. The *sum-product* algorithm leverages this pattern to simplify the calculation. Next we generalize these message passing equations to the case of an arbitrary graph.

3.3.2 Markov Random Fields

Let $G = (V, E)$ be an undirected graph with the set of vertices $V = \{1, \ldots, n\}$. Let $X = (X_1, \ldots, X_n)$ be a random vector and $\Omega = \{1, \ldots, N\}$ be the set of possible outcomes for each random variable. In a pairwise Markov random field, the joint distribution is

$$\mathbb{P}(X = x) = \frac{1}{Z} \prod_{i \in V} \phi_i(x_i) \prod_{\{i,j\} \in E} \psi_{ij}(x_i, x_j),$$

where Z is a normalization constant.

When the factor graph is tree-structured, the sum-product algorithm can be used to efficiently compute the marginals. In this case, messages are computed with the sum-product equations

$$m_{ij}(x_j) = \sum_{x_i} \phi_i(x_i)\, \psi_{ij}(x_i, x_j) \prod_{k \in \mathcal{N}(i) \setminus j} m_{ki}(x_i).$$

After a node receives a message from each of its neighbors, the incoming messages are combined to compute beliefs by using Equation 3.4. Similar to the case of max-product algorithm, the beliefs are the exact marginals when the graph is tree-structured [64].

When the graph contains cycles, the sum-product equation can be adapted into a message passing operator. The operator used in this version of belief propagation is nearly the same as the max-product algorithm. The only difference is that it includes a “sum” as opposed to a “max”.

Definition 3.3.1. *The sum-product message passing operator $S : \mathcal{M} \to \mathcal{M}$ that computes messages sent from factors to variables is $S = NR$. R computes new messages with the sum-product equation* *and N normalizes messages.*

$$\begin{aligned}
\big(Rm\big)_{ij}(x_j) &= \sum_{x_i} \phi_i(x_i)\, \psi_{ij}(x_i, x_j) \prod_{k \in \mathcal{N}(i)\setminus j} m_{ki}(x_i) \\
\big(Nm\big)_{ij}(x_j) &= \frac{m_{ij}(x_j)}{\sum\limits_{\tau \in \Omega} m_{ij}(\tau)}
\end{aligned}$$

In loopy belief propagation the messages are initialized, then repeatedly updated by using fixed point iteration with the operator S. If the algorithm converges, the messages are used to compute beliefs by using Definition (3.1.2). The beliefs then provide an approximation to the marginals. The sum-product algorithm (even with damping) suffers from the exact same issues as the max-product algorithm, namely the messages often fail to converge, the operator can have multiple fixed points, and the resulting fixed point may depend on the initialization of messages.

3.3.3 Factor Graphs

In this section, we present the factor graph version of the the sum-product algorithm. Most details of this algorithm are very closely related to the factor graph version of the max-product algorithm. In order to avoid being repetitive, we simply define the message passing operators used in this algorithm.

Definition 3.3.2. *The sum-product message passing operator $T : \mathcal{M} \to \mathcal{M}$ that computes messages sent from variables to factors is*

$$\big(T\mu\big)_{i\to f}(x_i) = \prod_{g \in \mathcal{N}(i)\setminus f} \mu_{g\to i}(x_i).$$

Definition 3.3.3. *The sum-product message passing operator $S : \mathcal{M} \to \mathcal{M}$ that computes messages sent from factors to variables is $S = NR$. R computes new messages using the sum-product equation* *and N normalizes messages.*

$$\begin{aligned}
\big(R\nu\big)_{f\to i}(x_i) &= \sum_{x_{\mathcal{N}(f)}} \Psi_f(x_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)\setminus i} \nu_{j\to f}(x_j) \\
\big(N\nu\big)_{f\to i}(x_i) &= \frac{\nu_{i\to f}(x_i)}{\sum\limits_{\tau\in\Omega} \nu_{i\to f}(\tau)}
\end{aligned}$$

The message passing operators in the sum-product algorithm share several commonalities with the operators from the max-product algorithm. First, the operator T which computes message from variables to factors is identical in both algorithms. Then both algorithms utilize a normalization operator denoted by N. In fact, the only difference is the operator which updates messages sent from factors to variables uses a “sum” as opposed to a “max”.

3.4 Literature Review

Since Pearl popularized the algorithm in the late 1980s, there has been significant progress in understanding the behavior of loopy belief propagation. Research in this area has mostly centered around three key topics: convergence, stability, and alternatives. Studying the convergence and stability of the algorithm has led to definitive criteria that characterizes when the algorithm converges. This insight has led to the development of alternative message passing algorithms that address aspects of the model that often cause belief propagation to fail.

3.4.1 Convergence

Weiss (2000) analyzes the accuracy of belief propagation when the graph contains a single loop [80]. For these networks, belief propagation with sufficient damping is guaranteed to converge to a unique fixed point that corresponds to the MAP solution [81]. Although the resulting beliefs may be inaccurate, nodes can use local information in the messages to correct their beliefs. Weiss and Freeman (2001) extend this result to graphs with arbitrary topology and nodes that describe jointly Gaussian random variables [82].

Jordan and Tatikonda (2002) address the convergence of belief propagation by drawing a connection between Gibbs measures and the algorithm's computation tree[2]. Their main result is that belief propagation is guaranteed to converge when the model satisfies Dobrushin's condition (see [28], [75], [76]). The main idea of this condition is that oscillations will not occur when the influence that nodes have on each other is sufficiently small. Ihler et al. (2005) derive a slightly stronger condition by bounding message error in terms of the *dynamic range*[3] of the potentials [43].

Heskes (2004) derives sufficient conditions for unique fixed points that depends on both the topology of the graph and the strength of the potentials. His approach utilizes the connection between fixed points of loopy belief propagation and extrema of the Bethe free energy [39]. The main result is a set of conditions that guarantee when the Bethe free energy is convex. Equivalently, these conditions also imply that belief propagation has a unique fixed point [40]. One interesting corollary of this work is that the Bethe free energy is always convex when the graph is tree-structured.

3.4.2 Stability of Fixed Points

In stability analysis[4], the main objective is to understand how fixed point iteration behaves with respect to small perturbations of a fixed point. The stability of a fixed point is closely related to convergence since fixed point iteration tends to converge when the fixed point is attractive. There is a stream of belief propagation research which aims to derive stability conditions that describe when the fixed points are locally attractive.

Heske (2002) approached the stability problem by leveraging the relationship between belief propagation and the Bethe free energy. He proved that stable fixed points of loopy belief propagation are minima of the Bethe free energy [39]. Watanabe and Fukumizu (2009) established a formula that connects the Hessian of the Bethe free energy with the edge zeta function [79]. The formula clarifies the relation initially described by Heske, while also providing new insights on the connection between belief propagation and the edge zeta function. Martin and Lasgouttes (2012) derive a sufficient condition for local stability in terms of the graph structure and the beliefs values

[2]The computation tree represents an unwrapping of the original graph with respect to belief propagation [76].

[3]The dynamic range $d(f)$ of a positive function is given by $d(f) = \sup_{x,y} \sqrt{f(x)/f(y)}$

[4]We assume a basic understanding of a fixed point being (globally or locally) attractive versus repelling. Otherwise, we direct the reader to Devaney's book [18] as a resource.

at the fixed point [60]. Their work provides insight on why belief propagation is more likely to converge on sparse graphs.

3.4.3 Alternative Belief Propagation Algorithms

In general, the belief propagation literature focuses on two distinct topics. The first is to understand the behavior of the algorithm, while the second is to present convergent alternatives to the traditional belief propagation algorithms. The main objective of this section is to introduce some important alternative algorithms that perform approximate inference in the problem instances where the traditional algorithms fail.

In the last two decades, there has been a push to derive message passing algorithms from convex free energies. An important member of this class are the *generalized* belief propagation algorithms by Yedida et al. (2000). The main idea of this approach is reduce the number of cycles in the graph by formulating message passing between regions of nodes [85]. They prove that the fixed points are identical to the stationary points of a region-based free energy. Although the algorithm outperforms traditional belief propagation, the performance is highly dependent upon how the graph is partitioned into regions. There have been a number of follow up works that have addressed this issue (see, e.g., [12],[83],[86]).

Another important class of algorithms are *tree-reweighted* belief propagation (TRW) methods introduced by Wainwright et al. (2003) [77]. Their approach was inspired by maximizing a lower bound on the log partition function of a graphical model by approximating the distribution with a convex combination of tree-structured distributions [78]. One outcome of this work is TRW in which messages are adjusted by edge weights determined by the structure of the graph. In a follow up work, Wainwright et al. (2008) derived a condition that guarantee when their algorithm converges on arbitrary graphs [72]. We return to this topic in the next chapter since it is closely related to convex combination belief propagation.

In addition, there are a number of different algorithms that are based upon ideas from convex optimization. Yuille (2004) developed a double-loop algorithm to minimize the Bethe free energy. The method is derived by using a concave-convex procedure where the free energy is decompose into a convex part and a concave part (see [89], [90]). The main advantage of the algorithm is that it's guaranteed to converge. However, it hasn't gained much attention since the runtime exceeds the traditional algorithm by an order of magnitude.

CHAPTER 4

Convex Combination Belief Propagation on Pairwise Models

In this chapter, we present convex combination belief propagation which is a convergent alternative to traditional belief propagation. This chapter focuses on describing the main ideas behind this class of algorithms, while also discussing some theoretical results. In Section 4.1, we describe our motivations for developing this algorithm along with a concise overview of related methods. We introduce the min-sum version of this algorithm and prove that it converges in Section 4.2. Section 4.3 is a more theoretical and presents a characterization of the beliefs obtained by this algorithm. Then Section 4.4 focuses on applying convex combination belief propagation to perform image restoration. Lastly, we conclude by discussing the sum-product algorithm in Section 4.5 and discuss some limitations of the algorithm in Section 4.6.

4.1 Introduction

Belief propagation is a well-established message passing algorithm that has diverse applications ranging from neuroscience to economics to statistical physics. In fact, the algorithm has been independently rediscovered in a number of different disciplines over the last 60 years. The modern formulation of belief propagation was introduced by Judea Pearl in the 1980s as a method that performs exact inference on a tree-structured graph in polynomial time [64]. There is a popular alternative form of this algorithm referred to as *loopy* belief propagation that uses fixed point iteration to perform approximate inference in graphs that contains cycles.

The performance of this algorithm is highly dependent upon the topology of the graph. Broadly speaking, the algorithm performs exact inference when the graph tree-structured and approximate inference when the graph contains cycles. However, the topology has a more intricate effect upon the accuracy and performance of the algorithm. For example, the algorithm is essentially guaranteed to perform exact inference when the graph only contains a single cycle, except in a few rare cases [82]. The algorithm is also known to perform extremely well when the graph is sparse. Loopy belief propagation has gained popularity within the artificial intelligence and information theory communities due to its success on these types of graphs.

As the topology of the graph becomes more complex, the performance of the algorithm deteriorates. Given that loopy belief propagation is a fixed point iteration scheme, cycles in the graph cause the algorithm to become unstable. One issue is that graphs with many cycles are known to result in multiple fixed points, which makes the algorithm sensitive to the initialization. The local dynamics of fixed points are even more problematic because some may be repelling. This causes the messages to oscillate which often results in the algorithm failing to converge. Despite the potential for an inconclusive result, loopy belief propagation is often used anyways since approximate inference is central to many applications involving graphical models.

Over the last 20 years, approximate inference has been an active area of research since exact inference is computationally intractable in high dimensional models. Some general approaches to this problem include stochastic simulation, variational methods, and alternative message passing algorithms. Since the main objective of this chapter is to present a new algorithm that falls into the latter category, we provide a concise literature review that describes some important algorithms of this type. Wainwright et al. (2003) developed a class of local message passing algorithms referred to as *tree-reweighted* belief propagation [77]. The approach was inspired by maximizing a lower bound on the log partition function by approximating the distribution with convex combinations of tree-structured distributions [77]. Their algorithm incorporates edge-based weights that are determined by the structure of the graph [72]. The algorithm converges to a unique fixed point for arbitrary graphs when the weights satisfy certain conditions. One drawback of TRW it can difficult to determine weights that both satisfy these conditions and result in a good approximation.

Kolmorogov (2006) builds upon their work by developing *sequential* tree-reweighted belief propagation [52]. This algorithm utilizes sequential message updates as opposed to parallel updates. Once an update schedule is determined, the algorithm is guaranteed to converge. However, one drawback is that the solution depends on the update schedule. Hazan and Shashua (2010) developed another closely related algorithm called *norm-product* belief propagation [35]. They take a general perspective on BP and TRW by using convex duality to derive a local message passing algorithm from a fractional free energy. This algorithm similarly utilizes edge weights to guarantee convergence. One challenge is that the weights are determined by solving an optimization problem which may be quite difficult. Felzenszwalb and Svaiter (2019) used a type of non-linear diffusion to obtain globally convergent methods for approximate inference in graphical models [20]. The structure of this algorithms is similar to belief propagation in the sense that information is propagated across a graph using fixed point iteration. However, their algorithm only keeps track of information associated with the vertices of a graph instead of utilizing messages sent along edges.

In this chapter, we present a new class of message passing algorithms called *convex combination* belief propagation. Convex combination belief propagation is an inference engine that is designed for the most difficult inference tasks. Our motivation for developing convex combination belief propagation is that loopy belief propagation often fails on graphs with complex topology. In particular, graphs with short cycles and dense regions are especially problematic. These regions may become message passing feedback loops where statistical information is over-counted which causes the algorithm to fail. Convex combination belief propagation addresses this problem by utilizing edge-weights that mitigates the effect of feedback loops. Most importantly, the algorithm converges to a unique solution regardless of the topology of the graph and initialization of the messages. One advantage of this algorithm over similar methods (e.g. TRW, sequential-TRW, norm-product BP) is that the edge-weight condition is much easier to verify and it can readily incorporated into existing implementations of belief propagation.

4.2 Min-Sum Algorithm

In this section, we present *convex combination* belief propagation which is a convergent alternative to loopy belief propagation. The main objectives of this section are to introduce the message passing operator used in the min-sum version of the algorithm. Then prove that the algorithm is guaranteed to converge to a unique fixed point.

4.2.1 Theoretical Settings

Let $G = (V, E)$ be an undirected graph with $V = \{1, \ldots, n\}$ and let $\Omega = \{1, \ldots, N\}$ be a set of labels. The cost $E : \Omega^n \to \mathbb{R}_+$ of a labeling x is given by the Gibbs energy

$$E(x) = \sum_{i \in V} g_i(x_i) + \sum_{\{i,j\} \in E} h_{ij}(x_i, x_j).$$

The functions g_i capture local information about a node. The value of $g_i(x_i)$ is a cost for assigning label x_i to node i. The function h_{ij} captures pairwise relationships to enforce that neighboring nodes have compatible labels. Note that this energy is a pairwise model since it utilizes unary and binary cost functions.

4.2.2 Message Passing Operator

We present a local message passing algorithm referred to as *convex combination* belief propagation. The message passing operator in our algorithm is a slight modification of the operator from loopy belief propagation. The key difference is that our operator utilizes edge-weights. Messages sent between neighboring nodes incorporate a convex combination of messages from the other neighbors. The weights dampen messages so that each node prioritizes information from nearby nodes.

Definition 4.2.1. *The message passing operator $\hat{Q} : \mathcal{M} \to \mathcal{M}$ in min-sum convex combination belief propagation is given by*

$$\left(\hat{Q}m\right)_{ij}(x_j) = \min_{x_i} \Big\{ g_i(x_i) + h_{ij}(x_i, x_j) + \gamma \sum_{k \in \mathcal{N}(i) \setminus j} w_{ki} m_{ki}(x_i) \Big\},$$

where the weights must satisfy $\sum_{k \in \mathcal{N}(i) \setminus j} w_{ki} \leq 1$ *with* $w_{ki} > 0$ *and* $\gamma \in (0, 1)$.

The only conditions imposed upon the weights is that they must be non-negative and sum to one. The simplest way to define each weight is to set $w_{ki} = 1/(d(i) - 1)$, where $d(i)$ is the degree of node i. Alternatively one can give more weight to some edges based on some additional information from a particular application (see [20]). Intuitively, the weights control how much influence neighboring nodes have upon each other. When the message sent from node i to j incorporates uniform weights, the other neighbors have equal influence upon node j. Non-uniform weights may be used to give some neighbors more influence.

This operator also incorporates a damping factor $\gamma \in (0, 1)$. This ensures the algorithm always converges. In addition, the rate of convergence is dependent upon its magnitude. We recommend setting this factor very close to one since the algorithm generally converges very quickly. Another important aspect of this algorithm is that we do not center updated messages, which differs from loopy belief propagation where the updates are centered to have mean zero. The inclusion of the weights w_{ki} and factor γ are sufficient to ensure that fixed point iteration with $\hat{Q}$ converges to a unique fixed point.

4.2.3 Convergence

Next we prove that convex combination belief propagation converges to a unique fixed point independent of the topology of the graph and initialization of the messages. Our general approach is to consider belief propagation as a discrete-time map, then use results from the theory of dynamical systems. In particular, we use Banach's fixed point theorem which guarantees existence and uniqueness of fixed points of certain discrete-time maps referred to as *contractions*. Given a metric space (X, d), an operator $F : X \rightarrow X$ is called a contraction if there is a constant $\gamma \in (0, 1)$ such that

$$d(F(x), F(y)) \leq \gamma d(x, y)$$

for any $x, y \in X$ where d is a metric. Banach's fixed point theorem states that a contractive operater (i.e. contraction) defined over a complete metric space has a unique and globally attractive fixed point $x^{\star} \in X$ [31].

In order to use this result, we must set up the problem under the framework of a metric space. Recall that each message sent between two nodes is a real-valued vector $m_{ij} \in \mathbb{R}^N$ with $N = |\Omega|$ and the pair $(\mathbb{R}^N, d_\infty)$ is a complete metric space, where $d_\infty : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ is the metric induced by the ℓ^∞-norm.

Observation 4.2.2. *Let $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ be the metric given by*

$$d(m,n) = \max_{i \in V} \max_{j \in \mathcal{N}(i)} \|m_{ij} - n_{ij}\|_\infty,$$

then the pair $\big(\mathcal{M}, d\big)$ is a complete metric space.

Lemma 4.2.3. *The operator $\hat{Q}$ is contractive with Lipschitz constant γ.*

Proof. Choose any $m, n \in \mathcal{M}$, then

$$\begin{aligned}
\big(\hat{Q}m\big)_{ij}(x_j) &= \min_{x_i} \Big\{ g_i(x_i) + h_{ij}(x_i,x_j) + \gamma \sum_{k \in \mathcal{N}(i)\backslash j} w_{ki} m_{ki}(x_i) \Big\} \\
&= \min_{x_i} \Big\{ g_i(x_i) + h_{ij}(x_i,x_j) + \gamma \sum_{k \in \mathcal{N}(i)\backslash j} w_{ki} \Big(n_{ki}(x_i) - n_{ki}(x_i) + m_{ki}(x_i) \Big) \Big\} \\
&\le \min_{x_i} \Big\{ g_i(x_i) + h_{ij}(x_i,x_j) + \gamma \sum_{k \in \mathcal{N}(i)\backslash j} w_{ki} n_{ki}(x_i) + \gamma \sum_{k \in \mathcal{N}(i)\backslash j} w_{ki} \|m_{ki} - n_{ki}\|_\infty \Big\} \\
&\le \min_{x_i} \Big\{ g_i(x_i) + h_{ij}(x_i,x_j) + \gamma \sum_{k \in \mathcal{N}(i)\backslash j} w_{ki} n_{ki}(x_i) \Big\} + \gamma \sum_{k \in \mathcal{N}(i)\backslash j} w_{ki} \|m_{ki} - n_{ki}\|_\infty \\
&= \big(\hat{Q}n\big)_{ij}(x_j) + \gamma \sum_{k \in \mathcal{N}(i)\backslash j} w_{ki} \|m_{ki} - n_{ki}\|_\infty
\end{aligned}$$

$$\implies \big(\hat{Q}m\big)_{ij}(x_j) - \big(\hat{Q}n\big)_{ij}(x_j) \le \gamma \sum_{k \in \mathcal{N}(i)\backslash j} w_{ki} \|m_{ki} - n_{ki}\|_\infty.$$

Since this inequality holds when m and n are interchanged, we can take the absolute value of the left hand side. In addition, this inequality holds for any $x_j \in \Omega$, so it also holds for the maximum of the left hand side.

$$\begin{aligned}
\left\|(\hat{Q}m)_{ij} - (\hat{Q}n)_{ij}\right\|_\infty &\le \gamma \sum_{k\in\mathcal{N}(i)\backslash j} w_{ki}\|m_{ki} - n_{ki}\|_\infty \\
&\le \gamma \max_{k\in\mathcal{N}(i)\backslash j} \|m_{ki} - n_{ki}\|_\infty \\
&\le \gamma \max_{i\in V}\max_{j\in\mathcal{N}(i)}\max_{k\in\mathcal{N}(i)\backslash j} \|m_{ki} - n_{ki}\|_\infty \\
&= \gamma \max_{i\in V}\max_{j\in\mathcal{N}(i)} \|m_{ij} - n_{ij}\|_\infty
\end{aligned}$$

The simplification on the second line holds by using that the weights define a convex combination. Since the inequality holds for any $\{i, j\} \in E$ in the left hand side,

$$\max_{i\in V}\max_{j\in\mathcal{N}(i)} \left\|(\hat{Q}m)_{ij} - (\hat{Q}n)_{ij}\right\|_\infty \le \gamma \max_{i\in V}\max_{j\in\mathcal{N}(i)} \|m_{ij} - n_{ij}\|_\infty$$

$$\implies d\big(\hat{Q}m, \hat{Q}n\big) \le \gamma\, d(m, n).$$

□

Theorem 4.2.1. *The message passing operator $\hat{Q}$ has a unique fixed point $m^\star \in \mathcal{M}$ and any sequence of messages defined by $m^{(\ell+1)} := \hat{Q}m^{(\ell)}$ converges to $m^\star$. Furthermore, after ℓ iterations*

$$d\big(\hat{Q}^{(\ell)}m^{(0)}, m^\star\big) \le \gamma^\ell\, d\big(m^{(0)}, m^\star\big).$$

Proof. Given that $\hat{Q}$ is a contraction according to Lemma 4.2.3, the result holds by applying Banach's fixed point theorem. □

This result establishes that the min-sum version of convex combination belief propagation is guaranteed to converge. Since the argument does not make any assumptions about the graph, the algorithm converges independent of the topology of the graph.

In the previous chapter, we introduced damped belief propagation which generally improves the performance of loopy belief propagation, but there are few theoretical guarantees on when damped belief propagation converges. In practice, damped belief propagation often fails to converge on graphs with complex topology and a large number of nodes. As an example, we compare damped and convex combination belief propagation on a graph of this type.

Example 4.2.4. *Let $G = (V, E)$ be a complete graph with 10 nodes and let $\Omega = \{-1, 1\}$ be a set of labels. Consider the energy function*

$$E(x) = \sum_{i \in V} y_i x_i + \sum_{\{i,j\} \in E} \lambda_{ij} x_i x_j$$

Each y_i was uniformly sampled from $\{-1, 1\}$ and λ_{ij} was sampled from a standard normal distribution. Let $\gamma = 0.9$ and $w_{ki} = 1/(|\mathcal{N}(i)| - 1)$ in convex combination belief propagation. In damped belief propagation, we varied the damping factor from $\alpha = 0.8$ to $\alpha = 0.99$ in an attempt to make the algorithm converge. The messages were initialized as $m_{ij}^{(0)} = (1, 1)$, then damped belief propagation (fixed point iteration with Q_α) and convex combination belief propagation (fixed point iteration with $\hat{Q}$) were applied to this example.

The resulting beliefs of the node 1 are shown in Figures 4.1 and 4.2, where $b_1^{(\ell)}$ and $\hat{b}_1^{(\ell)}$ denote the beliefs computed by damped and convex combination belief propagation on the ℓ-th iteration, respectively. We see that damped belief propagation does not converge for any damping factor, while convex combination belief propagation converges very quickly.

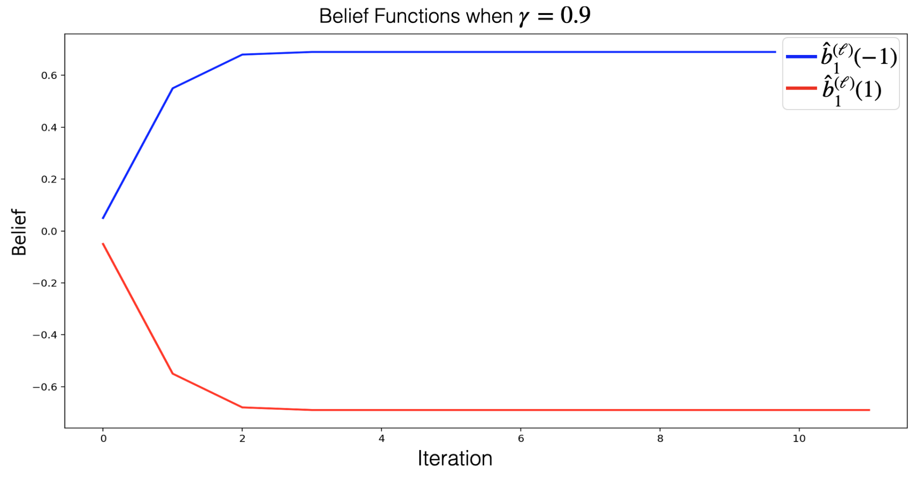


FIGURE 4.1: Beliefs obtained with convex combination belief propagation from Example 4.2.4. Here we see that the algorithm converges after a small number of iterations.

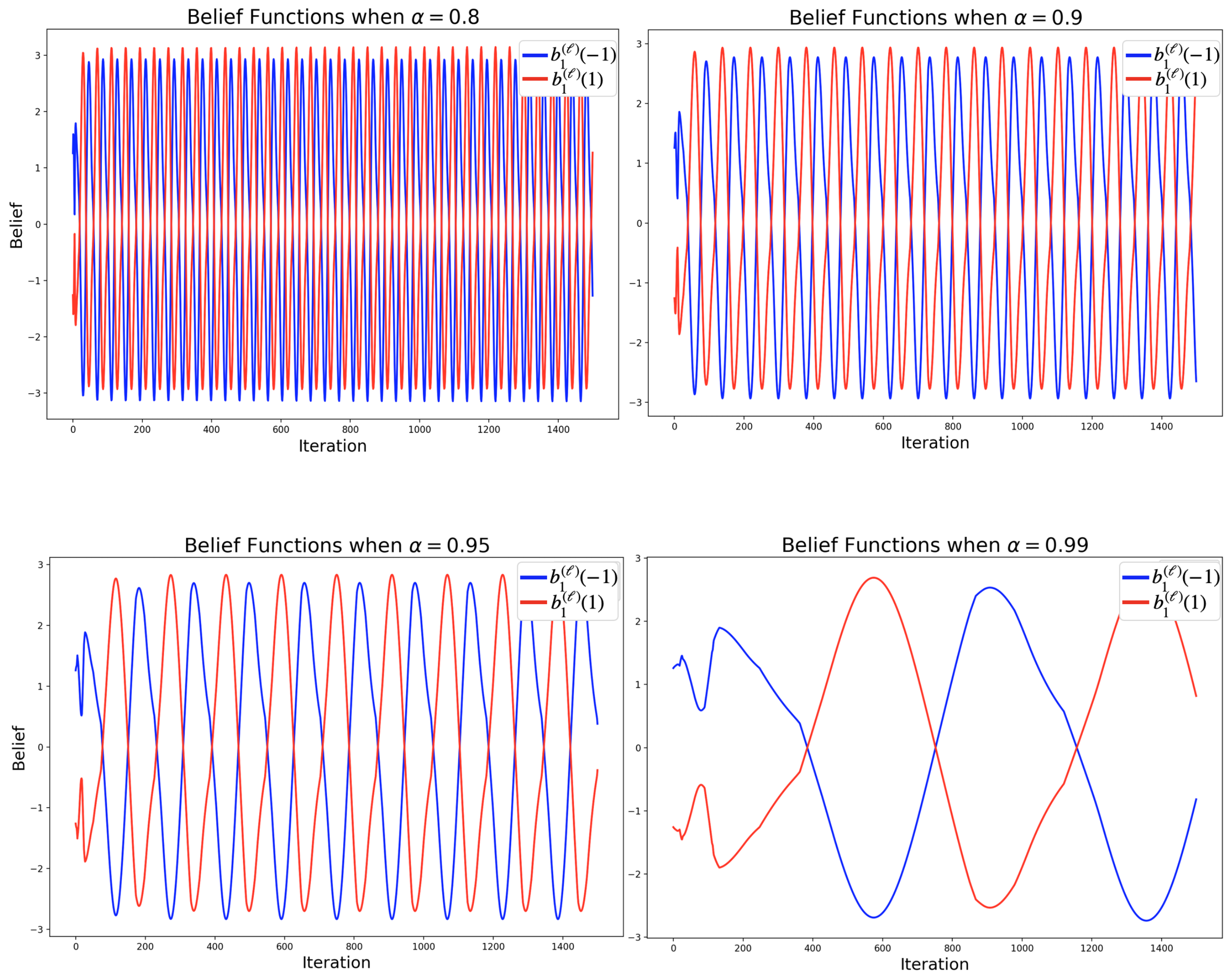


FIGURE 4.2: Beliefs obtained with damped belief propagation from Example 4.2.4. Here we see that the algorithm does not converge for any damping factor.

4.3 Characterization of Beliefs

Although convex combination belief propagation is guaranteed to converge, the algorithm is only useful if it converges to a meaningful result. When the graph is tree-structured, the traditional min-sum algorithm computes the exact min-marginals of the Gibbs energy. In this section, we provide an analogous characterization of the beliefs obtained with convex combination belief propagation. The main result in this section is beliefs are the exact min-marginals of a weighted energy which is closely related to the Gibbs energy.

4.3.1 Tree-Structured Graphs

Let m be the unique fixed point of $\hat{Q}$, then the belief function $\hat{b}_j : \Omega \to \mathbb{R}$ corresponding to node j is

$$\hat{b}_j(x_j) = g_j(x_j) + \sum_{i \in \mathcal{N}(j)} m_{ij}(x_j).$$

Next we prove that this belief is the exact min-marginal of a weighted energy $E_j(x)$. This energy is closely related to the Gibbs energy $E(x)$, but a key difference is that the cost functions are weighted according to the weights w_{ki} and factor γ used in the definition of $\hat{Q}$. Each belief $\hat{b}_j$ is the min-marginal of a different energy function $E_j(x)$ as the weights of each term depend on the choice of j.

Before defining the weighted energy for an arbitrary tree-structured graph, we provide a simple example of how to write down this energy for a small graph. In order to simplify the expressions for the energies, we define the cost function $H_{ij}(x_i, x_j) = g_i(x_i) + h_{ij}(x_i, x_j)$.

Example 4.3.1. *Let G be the graph shown in Figure 4.3. In this case, the Gibbs energy is*

$$E(x) = g_1(x_1) + H_{21}(x_2, x_1) + H_{32}(x_3, x_2) + H_{43}(x_4, x_3) + H_{53}(x_5, x_3).$$

Let $w_{ki} = 1/\big(|\mathcal{N}(i)| - 1\big)$ in the definition of $\hat{Q}$. When convex combination belief propagation is applied to this graph, the resulting belief function β_1 is the min-marginal of the energy:

$$E_1(x) = g_1(x_1) + H_{21}(x_2, x_1) + \gamma H_{32}(x_3, x_2) + \frac{\gamma^2}{2} H_{43}(x_4, x_3) + \frac{\gamma^2}{2} H_{53}(x_5, x_3),$$

This energy contains the same cost functions as the Gibbs energy, but each term $H_{ij}(x_i, x_j)$ is multiplied by a product of weights and a power of γ.

Now we will generalize the energy in the example above to the case of an arbitrary tree-structured graph.

Let $\mathcal{T}(j)$ be a tree-structured graph with a distinguished root node $j \in V$. Once a tree $\mathcal{T}$ is rooted at a node j, every node $i \neq j$ has a unique parent $\mathcal{P}(j, i)$, a set of children $\mathcal{C}(j, i)$, and a set of descendants $\mathcal{D}(j, i)$. Let $\mathcal{T}(j, i)$ denote the subtree of $\mathcal{T}(j)$ rooted at node i as illustrated in Figure 4.4. The subtree $\mathcal{T}(j, i)$ includes node i and its descendants.

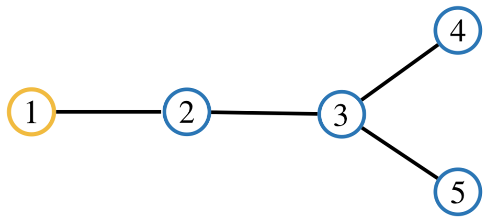


FIGURE 4.3: A simple tree-structured graph.

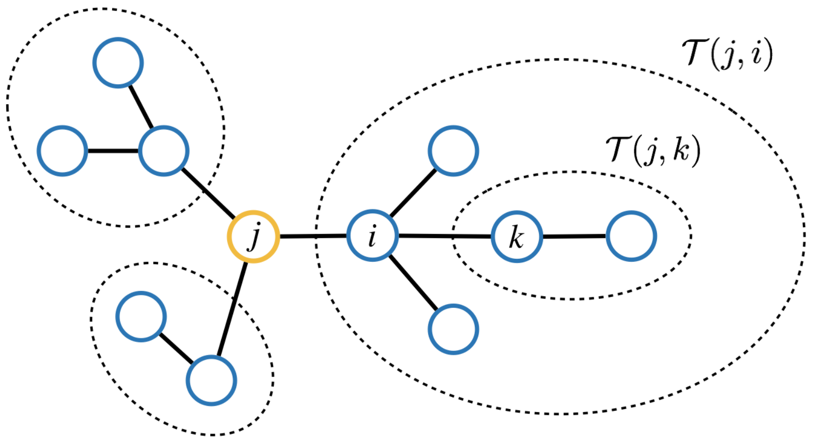


FIGURE 4.4: Subtrees of $\mathcal{T}(j)$

Let $R(\mathcal{T}(j))$ be the depth of a rooted tree. Let $\mathcal{N}_d(\mathcal{T}(j)) \subset V$ with $d \geq 0$ be the set of nodes at distance d from the root j.

Definition 4.3.2. *Let $w : V \times V \to \mathbb{R}_+$ be the weight function*

$$w(i,j) = \prod_{(k,l) \in E(i,j)} w_{kl},$$

where $E(i,j)$ is the collection of directed edges along the path from node i to j.

Definition 4.3.3. *Let $E_j : \Omega^n \to \mathbb{R}$ be the weighted energy function given by*

$$E_j(x) = g_j(x_j) + \sum_{i \in \mathcal{N}(j)} \sum_{d=0}^{R(\mathcal{T}(j,i))-1} \sum_{k \in N_d(\mathcal{T}(j,i))} \gamma^d w(k,i) H_{k\mathcal{P}(j,k)}(x_k, x_{\mathcal{P}(j,k)}),$$

This energy is closely related to the Gibbs energy but it puts more weight on terms "near" node j. For each edge $(k, \mathcal{P}(j,k))$ the cost associated with that edge is scaled by $\gamma^d w(k,i)$. When the damping factor and weights are removed from E_j, this energy is exactly the Gibbs energy.

Proposition 4.3.4. *If $w_{ki} = 1$ for all $\{k, i\} \in E$ and $\gamma = 1$, then $E_j(x) = E(x)$.*

Proof.

$$\begin{aligned}
E_j(x) &= g_j(x_j) + \sum_{i \in \mathcal{N}(j)} \sum_{d=0}^{R(\mathcal{T}(j,i))-1} \sum_{k \in \mathcal{N}_d(\mathcal{T}(j,i))} \gamma^d w(k,i) H_{k\mathcal{P}(j,k)}(x_k, x_{\mathcal{P}(j,k)}) \\
&= g_j(x_j) + \sum_{i \in N(j)} \sum_{d=0}^{R(\mathcal{T}(j,i))-1} \sum_{k \in \mathcal{N}_d(\mathcal{T}(j,i))} \Big(g_k(x_k) + h_{k\mathcal{P}(j,k)}\big(x_k, x_{\mathcal{P}(j,k)}\big)\Big) \\
&= g_j(x_j) + \sum_{k \in V \setminus j} \Big(g_k(x_k) + h_{k\mathcal{P}(j,k)}\big(x_k, x_{\mathcal{P}(j,k)}\big)\Big) \\
&= \sum_{k \in V} g_k(x_k) + \sum_{\{k,i\} \in E} h_{ki}(x_k, x_i) \\
&= E(x).
\end{aligned}$$

□

Next, we prove that convex combination belief propagation computes the exact min-marginals of the weighted energy in Definition 4.3.3. The main idea of the argument is to use that computing the min-marginals in a tree-structured graphical model can be broken down into subproblems that can be solved recursively. Each subproblem involves a function $F_{ji} : \Omega \to \mathbb{R}$ that corresponds to the min-marginals of an optimization problem defined by $\mathcal{T}(j, i)$.

Lemma 4.3.5. *Let $F_{ji} : \Omega \to \mathbb{R}$ be the function given by*

$$F_{ji}(x_i) = \min_{x_{\mathcal{D}(j,i)}} \Big\{ \sum_{d=1}^{R(\mathcal{T}(j,i))-1} \sum_{k \in \mathcal{N}_d(\mathcal{T}(j,i))} \gamma^d w(k,i)\, H_{k\mathcal{P}(j,k)}\big(x_k, x_{\mathcal{P}(j,k)}\big)\Big\},$$

then $F_{ji}(x_i) = \gamma \sum_{k \in \mathcal{N}(i) \setminus \mathcal{P}(j,i)} w_{ki} m_{ki}(x_i)$ where m is the unique fixed point of $\hat{Q}$.

Proof. This claim can be proven by inducting on the depth of the subtree $\mathcal{T}(j, i)$. In the base case when the depth is 1 we have $F_{ji}(x_i) = 0$ and the result follows trivially. Now for the induction step,

$$\begin{aligned}
F_{ji}(x_i) &= \min_{x_{\mathcal{D}(j,i)}} \Big\{ \sum_{d=1}^{R(\mathcal{T}(j,i))-1} \sum_{k\in\mathcal{N}_d(\mathcal{T}(j,i))} \gamma^d\, w(k,i) H_{k\mathcal{P}(j,k)}\big(x_k, x_{\mathcal{P}(j,k)}\big)\Big\} \\
&= \min_{x_{\mathcal{D}(j,i)}} \Big\{ \sum_{k\in\mathcal{N}(i)\setminus\mathcal{P}(j,i)} \gamma w_{ki} H_{ki}(x_k,x_i) + \sum_{d=2}^{R(\mathcal{T}(j,i))-1} \sum_{k'\in\mathcal{N}_d(\mathcal{T}(j,i))} \gamma^d w(k',i)\, H_{k'\mathcal{P}(j,k')}\big(x_{k'}, x_{\mathcal{P}(j,k')}\big)\Big\} \\
&= \min_{x_{\mathcal{D}(j,i)}} \Big\{ \sum_{k\in\mathcal{N}(i)\setminus\mathcal{P}(j,i)} \gamma w_{ki} \Big(H_{ki}(x_k,x_i) + \sum_{d=1}^{R(\mathcal{T}(j,k))-1} \sum_{k'\in\mathcal{N}_d(\mathcal{T}(j,k))} \gamma^d w(k',k)\, H_{k'\mathcal{P}(j,k')}\big(x_{k'}, x_{\mathcal{P}(j,k')}\big)\Big)\Big\} \\
&= \gamma \sum_{k\in\mathcal{N}(i)\setminus\mathcal{P}(j,i)} w_{ki} \min_{x_k} \Big\{ H_{ki}(x_k,x_i) + \min_{x_{\mathcal{D}(j,k)}} \Big\{ \sum_{d=1}^{R(\mathcal{T}(j,k))-1} \sum_{k'\in\mathcal{N}_d(\mathcal{T}(j,k))} \gamma^d w(k',k)\, H_{k'\mathcal{P}(j,k')}\big(x_{k'}, x_{\mathcal{P}(j,k')}\big)\Big\}\Big\} \\
&= \gamma \sum_{k\in\mathcal{N}(i)\setminus\mathcal{P}(j,i)} w_{ki} \min_{x_k} \Big\{ H_{ki}(x_k,x_i) + F_{jk}(x_k)\Big\} \\
&= \gamma \sum_{k\in\mathcal{N}(i)\setminus\mathcal{P}(j,i)} w_{ki} \min_{x_k} \Big\{ H_{ki}(x_k,x_i) + \gamma \sum_{k'\in\mathcal{N}(k)\setminus\mathcal{P}(j,k)} w_{k'k} m_{k'k}(x_k)\Big\} \\
&= \gamma \sum_{k\in\mathcal{N}(i)\setminus\mathcal{P}(j,i)} w_{ki} m_{ki}(x_i).
\end{aligned}$$

□

Theorem 4.3.1. *The belief function $\hat{b}_j$ is the min-marginal of E_j with respect to the j-th variable.*

Proof. Choose any $j \in V$, then the min-marginal p_j of E_j is

$$\begin{aligned}
p_j(x_j) &= \min_{x_{V\setminus j}} E_j(x) \\
&= g_j(x_j) + \min_{x_{V\setminus j}} \Big\{ \sum_{i\in\mathcal{N}(j)} \sum_{d=0}^{R(\mathcal{T}(j,i))-1} \sum_{k\in\mathcal{N}_d(\mathcal{T}(j,i))} \gamma^d w(k,i) H_{k\mathcal{P}(j,k)}(x_k, x_{\mathcal{P}(j,k)})\Big\} \\
&= g_j(x_j) + \sum_{i\in\mathcal{N}(j)} \min_{x_i} \Big\{ H_{ij}(x_i,x_j) + \min_{x_{\mathcal{D}(j,i)}} \Big\{ \sum_{d=1}^{R(\mathcal{T}(j,i))-1} \sum_{k\in\mathcal{N}_d(\mathcal{T}(j,i))} \gamma^d w(k,i)\, H_{k\mathcal{P}(j,k)}(x_k, x_{\mathcal{P}(j,k)})\Big\}\Big\}
\end{aligned}$$

Now using Lemma 4.3.5 implies that

$$\begin{aligned}
&= g_j(x_j) + \sum_{i \in \mathcal{N}(j)} \min_{x_i} \Big\{ H_{ij}(x_i, x_j) + F_{ji}(x_i) \Big\} \\
&= g_j(x_j) + \sum_{i \in \mathcal{N}(j)} \min_{x_i} \Big\{ H_{ij}(x_i, x_j) + \gamma \sum_{k \in \mathcal{N}(i) \setminus j} w_{ki} m_{ki}(x_i) \Big\} \\
&= g_j(x_j) + \sum_{i \in \mathcal{N}(j)} m_{ij}(x_j) \\
&= \hat{b}_j(x_j).
\end{aligned}$$

□

4.3.2 Simple Cycles

Next we generalize the previous characterization result to the case of a simple cycle. Our general approach is to first characterize the belief obtained after a finite number of iterations. Then make a limiting argument to get a characterization of the belief obtained from the fixed point of the algorithm (i.e. after an infinite number of iterations).

Finite Unwrapped Graphs

Let $C_n = (V_n, E_n)$ be an undirected cycle with $V_n = \{0, \dots, n-1\}$ and assume that the clockwise order of the vertices is $(0, \dots, n-1)$. The analysis in this section uses the notion of the *unwrapped* graph of a cycle. The unwrapped graph is initialized with a distinguished root node that represents some node on the cycle. We assume that the root corresponds to node $0 \in V_n$ in order to simplify notation. Then nodes are iteratively added to the unwrapped graph by simultaneously traversing the cycle clockwise and counterclockwise, and adding a copy of each node that is passed.

Definition 4.3.6. *Let $\hat{C}_{n,t}$ be the finite unwrapped graph of the undirected cycle C_n. This graph is generated recursively with $\hat{C}_{n,0} = \big(\{0\}, \emptyset\big)$ and after t iterations $\hat{C}_{n,t} = (\hat{V}_{n,t}, \hat{E}_{n,t})$ with*

$$\begin{aligned}
\hat{V}_{n,t} &= \hat{V}_{n,t-1} \cup \{-t,\ t\} \\
\hat{E}_{n,t} &= \hat{E}_{n,t-1} \cup \big\{ \{-t, -t+1\}, \{t-1,\ t\} \big\}.
\end{aligned}$$

Example 4.3.7. *The unwrapped graph $\hat{C}_{3,t}$ is shown in Figure 4.5.*

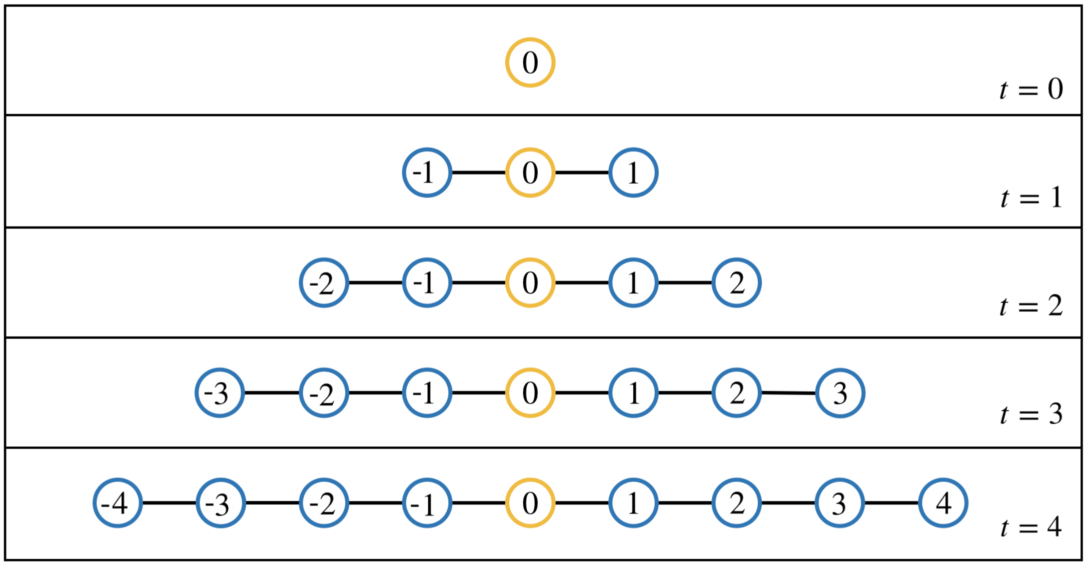


FIGURE 4.5: Unwrapping C_3 to obtain its unwrapped graph $\hat{C}_{3,t}$, where each t corresponds to an iteration of unwrapping.

Each node in the unwrapped graph is denoted by an integer, where negative nodes correspond to unwrapping the cycle in the counterclockwise direction. This notation is convenient because the correspondence between nodes in the unwrapped graph and cycle can be easily recovered via the modulo operation. For example, node $i \in \hat{V}_{n,t}$ corresponds to $i' \in V_n$ where $i' = i \mod n$.

The unwrapped graph is useful because there is an equivalence between performing message passing on a graph and its unwrapped graph. For any number of iterations of belief propagation, there exists a node on an unwrapped graph that receives the exact same message received by the corresponding node on the original graph [80]. To make this statement more precise, we define the convex combination message passing equations in the case of the unwrapped graph.

Definition 4.3.8. *Let $\hat{C}_{n,t}$ be an unwrapped graph and let $i' \in V_n$ be given by $i' = i \mod n$ for any $i \in \hat{V}_{n,t}$.*

Definition 4.3.9. *Let $\hat{H}_{ij} : \Omega \times \Omega \to \mathbb{R}$ be the cost function given by*

$$\hat{H}_{ij}(x_i, x_j) := g_{i'}(x_{i'}) + h_{i'j'}(x_{i'}, x_{j'})$$

with $\{i, j\} \in \hat{E}_{n,t}$.

Let $\hat{m}_{ij,t}$ denote the message sent from node i to j on the unwrapped graph $\hat{C}_{n,t}$. Message are computed with the min-sum equation

$$\hat{m}_{i+1\,i,t}(x_j) = \min_{x_i} \Big\{ \hat{H}_{ij}(x_i, x_j) + \gamma\, \hat{m}_{i+2\,i+1,t}(x_i) \Big\},$$

where $\gamma \in (0,1)$ and $w_{ki} = 1$ because each node has two neighbors.

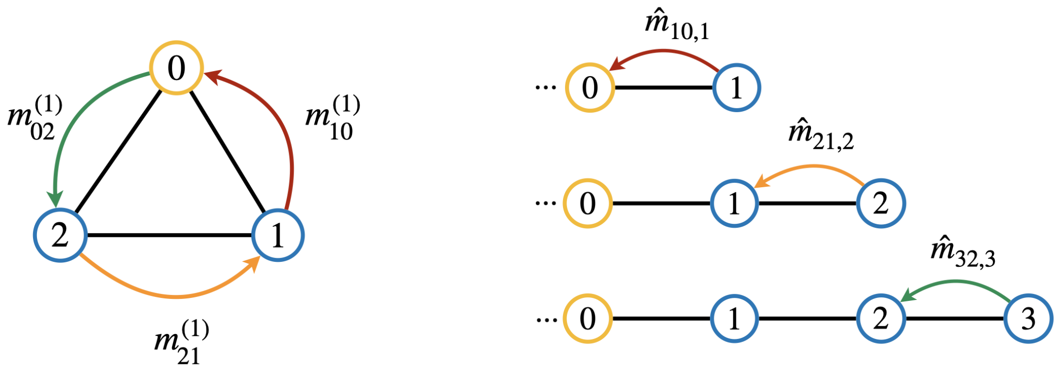


FIGURE 4.6: The relationship between message passing on the cycle C_3 and its unwrapped graphs is shown. Each message sent between nodes is represented by an arrow and identical messages have the same colored arrow.

Next we define a set of equations that precisely describes the correspondence between message passing on a cycle and its unwrapped graph. Since the notation used in this set of equations is a bit cumbersome, we provide a simple example in Figure 4.6 that captures the essence of the equations. In the case when $t = 1$ and $n \geq 3$, the set of equations is

$$\begin{aligned} m_{10}^{(1)}(x_0) &= \hat{m}_{10,1}(x_0) \\ m_{21}^{(1)}(x_1) &= \hat{m}_{21,2}(x_1) \\ &\vdots \\ m_{n-1\,n-2}^{(1)}(x_{n-2}) &= \hat{m}_{n-1\,n-2,n-1}(x_{n-2}) \\ m_{0n}^{(1)}(x_{n-1}) &= \hat{m}_{n\,n-1,n}(x_{n-1}) \end{aligned} \tag{4.1}$$

under the assumption that $m^{(0)} = \vec{0}$. In the first iteration of convex combination belief propagation, the value of $m_{i+1\,i}^{(1)}$ is completely dependent upon the minimizing the potential $H_{i+1\,i}$. The reason being that node $i+1$ receives a vector of zeros from the other neighbor due to $m^{(0)} := \vec{0}$. Similarly, the message $\hat{m}_{i+1\,i,i+1}$ is also entirely dependent upon minimizing the potential $\hat{H}_{i+1\,i} = H_{i+1\,i}$ because node i is the only neighbor of node $(i+1) \in \hat{C}_{n,i+1}$.

If we continue to write down an analogous set of equations in the cases when $t = 2$ and $t = 3$, we see a pattern emerge. In the most general case when $t \in \mathbb{N}$ and $n \geq 3$, the relationship between message passing on a cycle and its unwrapped graph is that

$$m^{(t)}_{i+1\,i}(x_i) = \hat{m}_{i+1\,i,\,t+i}(x_i) \qquad \forall i < n \tag{4.2}$$

$$m^{(t)}_{0\,n-1}(x_{n-1}) = \hat{m}_{n\,n-1,\,t+n-1}(x_{n-1}) \tag{4.3}$$

under the assumption that $m^{(0)} = \vec{0}$. This set of equations can be proven by fixing n and inducting on t. In the next example, we prove this result in the case when $n = 3$. Although this is a special case of the more general result to be presented later, we start with this simple case because it illustrates most of the underlying ideas without getting involved with the additional technicalities of the general case.

Example 4.3.10. *Given the cycle C_3, then the following equations hold for all $t > 0$*

$$\begin{aligned} m^{(t)}_{10}(x_0) &= \hat{m}_{10,t}(x_0) \\ m^{(t)}_{21}(x_1) &= \hat{m}_{21,t+1}(x_1) \\ m^{(t)}_{02}(x_2) &= \hat{m}_{32,t+2}(x_2) \end{aligned}$$

when $m^{(0)} := \vec{0}$. Under the assumption that the Equation 4.1 holds, then the first equation is true by

$$\begin{aligned} m^{t+1}_{10}(x_0) &= \min_{x_1} \Big\{ H_{10}(x_1, x_0) + \gamma\, m^{(t)}_{21}(x_1) \Big\} \\ &= \min_{x_1} \Big\{ \hat{H}_{10}(x_1, x_0) + \gamma\, \hat{m}_{21,t+1}(x_1) \Big\} \\ &= \hat{m}_{10,t+1}(x_0), \end{aligned}$$

which holds by the inductive hypothesis and definition of $\hat{m}_{10,t+1}$. The second equation can be proven by adapting the argument used above. The argument behind the third equation is a bit more *involved.*

$$\begin{aligned} m_{02}^{(t+1)}(x_2) &= \min_{x_0}\Big\{H_{02}(x_0,x_2) + \gamma\, m_{10}^{(t)}(x_0)\Big\} \\ &= \min_{x_0}\Big\{H_{02}(x_0,x_2) + \gamma\, \hat{m}_{10,t}(x_0)\Big\} \\ &= \min_{x_3}\Big\{\hat{H}_{32}(x_3,x_2) + \gamma\, \hat{m}_{10,t}(x_3)\Big\} \\ &= \min_{x_3}\Big\{\hat{H}_{32}(x_3,x_2) + \gamma\, \hat{m}_{43,t+3}(x_3)\Big\} \\ &= \hat{m}_{32,t+3}(x_2), \end{aligned}$$

where the second line holds by invoking the inductive hypothesis. The final line holds by using that nodes $4 \in \hat{V}_{3,t+3}$ and $1 \in \hat{V}_{3,t}$ correspond to node $1 \in V_3$. Moreover, these nodes have the exact same number of edges between themselves and the rightmost node on their respective graphs. The ordering of the edges is identical with respect to their corresponding edges on the cycle. Thus, these nodes receive an identical set of messages from their neighbor to the right and so $\hat{m}_{10,t}(x_3) = \hat{m}_{43,t+3}(x_3)$.

Theorem 4.3.2. *Given the cycle C_n, then the Equations (1.2) and (1.3) hold for all $n \geq 3$ and $t < \infty$ when $m^{(0)} := \vec{0}$.*

Proof. We prove this claim by fixing $n \in \mathbb{N}$ and inducting on t, the base case holds for every $i < n$ by

$$\begin{aligned} m_{i+1\,i}^{(1)}(x_i) &= \min_{x_{i+1}}\Big\{H_{i+1\,i}(x_{i+1},x_i) + \gamma\, m_{i+2\,i+1}^{(0)}(x_{i+1})\Big\} \\ &= \min_{x_{i+1}}\ H_{i+1\,i}(x_{i+1},x_i) \\ &= \min_{x_{i+1}}\ \hat{H}_{i+1\,i}(x_{i+1},x_i) \\ &= \hat{m}_{i+1\,i,1+i}(x_i), \end{aligned}$$

where the second line holds by using that $m^{(0)} = \vec{0}$. We conclude that the base case holds with

$$\begin{aligned} m^{(1)}_{0\,n-1}(x_{n-1}) &= \min_{x_0} \left\{ H_{0\,n-1}(x_0, x_{n-1}) + \gamma\, m^{(0)}_{10}(x_0) \right\} \\ &= \min_{x_0}\; H_{0\,n-1}(x_0, x_{n-1}) \\ &= \min_{x_0}\; \hat{H}_{n\,n-1}(x_n, x_{n-1}) \\ &= \hat{m}_{n\,n-1,n}(x_{n-1}). \end{aligned}$$

Now suppose the claim holds for some $t < \infty$, then the inductive step holds in the case $i < n$ holds by

$$\begin{aligned} m^{(t+1)}_{i+1\,i}(x_i) &= \min_{x_{i+1}} \left\{ H_{i+1\,i}(x_{i+1}, x_i) + \gamma\, m^{(t)}_{i+2\,i+1}(x_{i+1}) \right\} \\ &= \min_{x_{i+1}} \left\{ \hat{H}_{i+1\,i}(x_{i+1}, x_i) + \gamma\, \hat{m}_{i+2\,i+1,\,t+i+1}(x_{i+1}) \right\} \\ &= \hat{m}_{i+1\,i,\,t+1+i}(x_i), \end{aligned}$$

where the second line holds by the inductive hypothesis. We conclude that the claim holds by

$$\begin{aligned} m^{(t+1)}_{0\,n-1}(x_{n-1}) &= \min_{x_0} \left\{ H_{0\,n-1}(x_0, x_{n-1}) + \gamma\, m^{(t)}_{10}(x_0) \right\} \\ &= \min_{x_n} \left\{ \hat{H}_{n\,n-1}(x_n, x_{n-1}) + \gamma\, \hat{m}_{10,t}(x_n) \right\} \\ &= \min_{x_n} \left\{ \hat{H}_{n\,n-1}(x_n, x_{n-1}) + \gamma\, \hat{m}_{n+1\,n,t+n}(x_n) \right\} \\ &= \min_{x_n} \left\{ \hat{H}_{n\,n-1}(x_n, x_{n-1}) + \gamma\, \hat{m}_{n+1\,n,t+1+n-1}(x_n) \right\} \\ &= \hat{m}_{n\,n-1,\,t+1+n-1}(x_{n-1}). \end{aligned}$$

The third line holds by using that both node $1 \in \hat{V}_{n,t}$ and node $n+1 \in \hat{V}_{n,t+n}$ correspond to node $1 \in V_n$. Moreover, these nodes have the exact same number of edges between themselves and the right most node on their respective graphs. The order of the edges is identical with respect to their corresponding edges on the cycle. Thus, these nodes receive an identical set of messages from their neighbor to the right and so $\hat{m}_{10,t}(x_n) = \hat{m}_{n+1\,n,t+n}(x_n)$. $\square$

This result establishes an equivalence between message passing on the cycle and its unwrapped graph. Given any message $m_{ij}^{(t)}$ sent between nodes on the cycle, these equations specify which node on the unwrapped graph receives an identical set of messages. Note that these equations describe the relationship message passing counterclockwise around a cycle and message passing on the unwrapped graph from the rightward nodes to the root (see Figure 4.6). There is an analogous relationship between message passing clockwise around a cycle and from the leftward nodes on the unwrapped graph towards the roots. This relationship is established in Corollary 4.3.11 and illustrated in Figure 4.7.

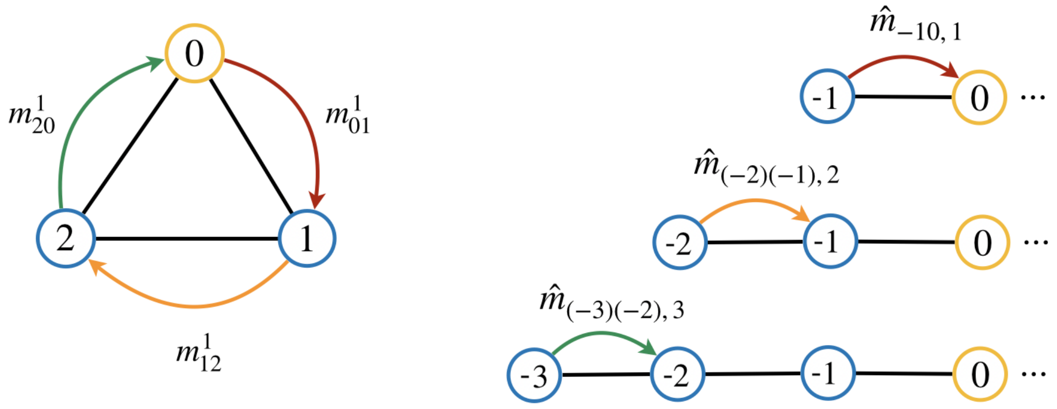


FIGURE 4.7: The relationship between passing messages around the cycle C_3 in the clockwise direction and from the leftward nodes to the root on the unwrapped graphs is shown. Each message sent between nodes is represented by an arrow and corresponding messages have the same colored arrow.

Corollary 4.3.11. *Given the cycle C_n for any $n \in \mathbb{N}$, then the following equations holds for every $i \in \{-(n-1), \ldots, 0\}$ and $t < \infty$*

$$m^t_{i-1\,i}(x_i) = \hat{m}_{i-1\,i,\,t+i}(x_i)$$

$$m^t_{(n-1)\,0}(x_0) = \hat{m}_{(n-1)\,n,\,t+(n-1)}(x_0)$$

under the assumption that $m^0 = \vec{0}$.

Proof. This claim holds by using that the unwrapped graph is symmetric about its root, so the argument in Theorem 4.3.2 can be adapted to prove this claim. $\square$

Next we use these results to establish an analogous equivalence between the corresponding belief functions. This result follows naturally because beliefs are computed from messages. Let $\beta_0^{(t)}$ be the belief corresponding to the node $0 \in V_n$ obtained after t iterations of convex combination belief propagation.

Definition 4.3.12. *Let $\hat{\beta}_{0,t} : \Omega \to \mathbb{R}$ be the belief function of node $0 \in \hat{V}_{n,t}$ after t iterations of convex combination belief propagation be given by*

$$\hat{\beta}_{0,t}(x_0) = g_0(x_0) + \sum_{i \in N(0)} \hat{m}_{i0,t}(x_0).$$

Corollary 4.3.13. *Let C_n be a cycle and $\hat{C}_{n,t}$ be its unwrapped graph, then $\beta_0^{(t)}(\tau) = \hat{\beta}_{i,t}(\tau)$ for all $\tau \in \Omega$ and $t < \infty$ when $m^{(0)} := \vec{0}$.*

Proof. Given any $\tau \in \Omega$, then

$$\beta_0^{(t)}(\tau) = g_0(\tau) + \sum_{i \in N(0)} m_{i0}^{(t)}(\tau) = g_j(\tau) + \sum_{i \in N(0)} \hat{m}_{i0,t}(\tau) = \hat{\beta}_{0,t}(\tau),$$

which holds by using that $m_{i0}^{(t)}(\tau) = \hat{m}_{i0,t}(\tau)$ by Theorem 4.3.2 and Corollary 4.3.11. □

The significance of Theorem 4.3.2 and Corollary 4.3.11 is that they reduce message passing on a cycle to message passing on its unwrapped graph which is tree-structured. Given that $\hat{\beta}_{0,t}$ corresponds to a node on a tree-structured graph, Theorem 4.3.1 characterizes this belief as the exact min-marginal of the weighted energy in Definition 4.3.3.

Definition 4.3.14. *Let $\hat{H}_{d^+} : \Omega \times \Omega \to \mathbb{R}$ and $\hat{H}_{d^-} : \Omega \times \Omega \to \mathbb{R}$ be the cost functions given by*

$$\hat{H}_{d^+}(x_{d^+}) := \hat{H}_{d,d+1}\big(x_d, x_{d+1}\big)$$
$$\hat{H}_{d^-}(x_{d^-}) := \hat{H}_{-d-1,-d}\big(x_{-d-1}, x_{-d}\big).$$

Definition 4.3.15. *Let $\hat{E}_{0,t} : \Omega^n \to \mathbb{R}$ be the energy given by*

$$\hat{E}_{0,t}(x) = g_0(x_0) + \sum_{d=0}^{t} \gamma^d \Big(\hat{H}_{d^-}(x_{d^-}) + \hat{H}_{d^+}(x_{d^+})\Big)$$

with $\gamma \in (0,1)$.

This energy is an adaptation of the energy in Definition 4.3.3 to the special case of a path graph rooted at its center node. Given that the graph $\hat{C}_{n,t}$ is tree-structured, then $\hat{\beta}_{0,t}$ is the exact min-marginal of $\hat{E}_{0,t}$ by Theorem 4.3.1. Thus, we can characterize $\beta_0^{(t)}$ by equivalently characterizing $\hat{\beta}_{0,t}$.

Theorem 4.3.3. *Given the cycle* C_n *and any* $t \in \mathbb{N}$*, then the belief* $\beta_0^{(t)}$ *is the exact min-marginal of node* $0 \in V_n$ *with respect to the energy* $\hat{E}_{0,t}$ *when* $m^{(0)} := \vec{0}$.

Proof. Given that the graph $\hat{C}_{n,t}$ is tree-structured, then $\hat{\beta}_{0,t}$ is the exact min-marginal of $\hat{E}_{0,t}$ by Theorem 4.3.1. Moreover, given $\beta_0^{(t)}(\tau) = \hat{\beta}_{0,t}(\tau)$ for all $\tau \in \Omega$ by Corollary 4.3.13, then $\beta_0^{(t)}(\tau)$ is the exact min-marginal of the $\hat{E}_{0,t}$. □

Infinite Unwrapped Graphs

Next we characterize the belief function β_0 obtained from the unique fixed point of $\hat{Q}$ when the graph is a cycle. We build upon the main result in the previous section by proving that β_0 is the min-marginal of an energy defined with respect to an *infinite* unwrapped graph.

The infinite unwrapped graph is a rooted path graph with infinitely many nodes that are labelled by the integers. Again, we assume that the root node of the unwrapped graph corresponds to node $0 \in V_n$ without loss of generality.

Definition 4.3.16. *Let* $\hat{C}_n = (\hat{V}_n, \hat{E}_n)$ *be the infinite unwrapped graph of the cycle* C_n *with*

$$\hat{V}_n := \mathbb{Z}$$
$$\hat{E}_n := \big\{\{i, i+1\} : i \in \mathbb{Z}\,\big\}.$$

Let $m \in \mathcal{M}$ be the unique fixed point of $\hat{Q}$. The fixed point can be equivalently written as the message obtained in the limit

$$m_{ij}(x_j) = \lim_{t\to\infty} m_{ij}^{(t)}(x_j) = \lim_{t\to\infty} \hat{m}_{ij,t}(x_j). \tag{4.4}$$

The first equality holds by using that $\hat{Q}$ being contractive implies that $m^{(t)} \to m$ and the second holds by Theorem 4.3.2. The variable t represents the iteration of belief propagation in the first limit. In the second limit, this variable represents the iteration of unwrapping the cycle to obtain its unwrapped graph. Thus, the limiting value m_{ij} has two equivalent interpretations. It is the message sent from node i to j after an infinite number of iterations of message passing on a cycle. Equivalently it is the message that node i sends to j when these are nodes on the infinite unwrapped graph.

Equation 4.4 captures the relationship between message passing on a cycle for an infinite number of iterations on a cycle and message passing on the infinite unwrapped graph. Now this relationship can be used to show an analogous relationship between the corresponding belief functions.

Proposition 4.3.17. *Given the cycle C_n and the unique fixed point $m \in \mathcal{M}$ of $\hat{Q}$, then*

$$\beta_0(\tau) = \lim_{t\to\infty} \beta_0^{(t)}(\tau) = \lim_{t\to\infty} \hat{\beta}_{0,t}(\tau).$$

Proof. Given any $\tau \in \Omega$, the first equality holds by

$$\begin{aligned}
\lim_{t\to\infty} \hat{\beta}_{0,t}(\tau) &= \lim_{t\to\infty} \Big(\hat{g}_0(\tau) + \sum_{i\in N(0)} \hat{m}_{i0,t}(\tau)\Big) \\
&= \lim_{t\to\infty} \Big(g_0(\tau) + \sum_{i\in N(0)} m_{i0}^{(t)}(\tau)\Big) \\
&= \lim_{t\to\infty} \beta_0^{(t)}(\tau),
\end{aligned}$$

where the second line holds by Theorem 4.3.2. Then the second equality holds by

$$\begin{aligned}
\lim_{t\to\infty} \beta_0^{(t)}(\tau) &= \lim_{t\to\infty} \Big(g_0(\tau) + \sum_{i\in N(0)} m_{i0}^{(t)}(\tau)\Big) \\
&= g_0(\tau) + \sum_{i\in N(0)} \lim_{t\to\infty} m_{i0}^{(t)}(\tau) \\
&= g_0(\tau) + \sum_{i\in N(0)} m_{i0}(\tau) \\
&= \beta_0(\tau).
\end{aligned}$$

□

Next we prove that β_0 is the min-marginal of an energy defined with respect to an *infinite* unwrapped graph. This energy is an adaptation of the energy $\hat{E}_{0,t}$ to the case of an infinitely long path graph.

Definition 4.3.18. *Let $\hat{E}_0 : \Omega^n \to \mathbb{R}$ be the energy function given by*

$$\hat{E}_0(x) = g_0(x_0) + \sum_{d=0}^{\infty} \gamma^d \Big(\hat{H}_{d^-}(x_{d^-}) + \hat{H}_{d^+}(x_{d^+}) \Big)$$

with $\gamma \in (0, 1)$.

Next we prove that β_0 is the exact min-marginal of the energy $\hat{E}_0$. The main idea of the argument is to split up this energy into three parts. One part corresponds to a finite path graph centered about the root. This implies $\hat{\beta}_0^{(t)}$ is the min-marginal of the truncated energy defined on a finite unwrapped graph. The two other parts are the tails of the energy, which converge to zero in the limit.

Lemma 4.3.19. *The tail of the energy $\hat{E}$ is uniformly bounded by*

$$\max_{x \in \Omega^n} \Big| \sum_{d=t}^{\infty} \gamma^d \Big(\hat{H}_{d^-}(x_{d^-}) + \hat{H}_{d^+}(x_{d^+}) \Big) \Big| \leq 2K \frac{\gamma^t}{1 - \gamma}$$

when $\gamma \in (0, 1)$.

Proof. Under the assumption that g_i and h_{ij} are both bounded, there exists some $K < \infty$ such that $|g_i(x_i) + h_{ij}(x_i, x_j)| \leq K$ for all $x_i, x_j \in \Omega$. Then the tail of the energy is bounded by

$$\begin{aligned}
\max_{x \in \Omega^n} \Big| \sum_{d=t}^{\infty} \gamma^d \Big(\hat{H}_{d^-}(x_{d^-}) + \hat{H}_{d^+}(x_{d^+}) \Big) \Big(&\leq \max_x \sum_{d=t}^{\infty} \gamma^d \Big) \hat{H}_{d^-}(x_{d^-}) + \hat{H}_{d^+}(x_{d^+}) \Big| \\
&\leq 2K \max_x \sum_{d=t}^{\infty} \gamma^d \\
&= 2K \sum_{d=t}^{\infty} \gamma^d \\
&= 2K \frac{\gamma^t}{1 - \gamma}.
\end{aligned}$$

□

Theorem 4.3.4. *Let C_n be an undirected cycle, then β_0 is the exact min-marginal of the energy $\hat{E}(x)$.*

Proof. Suppose that $t \in \mathbb{N}$ is fixed, then

$$\begin{aligned}
\max_{\tau\in\Omega}\Big|\beta_0(\tau) - \min_{\substack{x\in\Omega^n\\ x_0=\tau}} \hat{E}_0(x)\Big| &= \max_{\tau}\Big|\beta_0(\tau) - \Big(g_0(\tau) + \min_{\substack{x\\ x_0=\tau}} \sum_{d=1}^{\infty}\gamma^d\Big(\hat{H}_{d^-}(x_{d^-}) + \hat{H}_{d^+}(x_{d^+})\Big)\Big| \\
&\le \max_{\tau}\Big|\beta_0(\tau) - \Big(g_0(\tau) + \min_{\substack{x\\ x_0=\tau}} \sum_{d=1}^{t}\gamma^d\Big(\hat{H}_{d^-}(x_{d^-}) + \hat{H}_{d^+}(x_{d^+})\Big) \\
&\quad + \max_{\substack{x\\ x_0=\tau}} \sum_{d=t+1}^{\infty}\gamma^d\Big(\hat{H}_{d^-}(x_{d^-}) + \hat{H}_{d^+}(x_{d^+})\Big)\Big| \\
&= \max_{\tau}\big|\beta_0(\tau) - \min_{\substack{x\\ x_0=\tau}} \hat{E}_{0,t} + \max_{\substack{x\\ x_0=\tau}} \sum_{d=t+1}^{\infty}\gamma^d\Big(\hat{H}_{d^-}(x_{d^-}) + \hat{H}_{d^+}(x_{d^+})\Big)\Big| \\
&\le \max_{\tau}\Big|\beta_0(\tau) - \min_{\substack{x\\ x_0=\tau}} \hat{E}_{0,t} + 2K\frac{\gamma^{t+1}}{1-\gamma}\Big|,
\end{aligned}$$

where the last line holds by Lemma 4.3.19. Now we can use Proposition 4.3.17 to obtain the final result by

$$\begin{aligned}
&= \max_{\tau}\Big|\lim_{t\to\infty}\hat{\beta}_{0,t}(\tau) - \min_{\substack{x\\ x_0=\tau}} \hat{E}_{0,t}(x) + 2K\frac{\gamma^{t+1}}{1-\gamma}\Big| \\
&= \max_{\tau}\Big|\lim_{t\to\infty}\Big(\hat{\beta}_{0,t}(\tau) - \min_{\substack{x\\ x_0=\tau}} \hat{E}_{0,t}(x) + 2K\frac{\gamma^{t+1}}{1-\gamma}\Big)\Big| \\
&= \lim_{t\to\infty}\Big(\max_{\tau}\big|\hat{\beta}_{0,t}(\tau) - \min_{\substack{x\\ x_0=\tau}} \hat{E}_{0,t}(x)\big| + 2K\frac{\gamma^{t+1}}{1-\gamma}\Big) \\
&= \frac{2K}{1-\gamma}\lim_{t\to\infty}\gamma^{t+1} \\
&= 0,
\end{aligned}$$

where the last inequality holds by Theorem 4.3.13. □

4.4 Image Restoration

In this section, we demonstrate the practical application of convex combination belief propagation by using this algorithm to perform image restoration. The objective of image restoration is to estimate a clean (original) image given a corrupted version of it. A classical approach is to formulate this problem as finding the optimal labelling of a graph with respect to the Gibbs energy (see e.g. [6], [19], [27]). The cost functions used in the energy enforce that the restored image is piecewise smooth and consistent with the observed image. Once the image restoration model is formulated under this framework, convex combination belief propagation can be used to restore the corrupted image by obtaining an approximation to the minimum of the Gibbs energy.

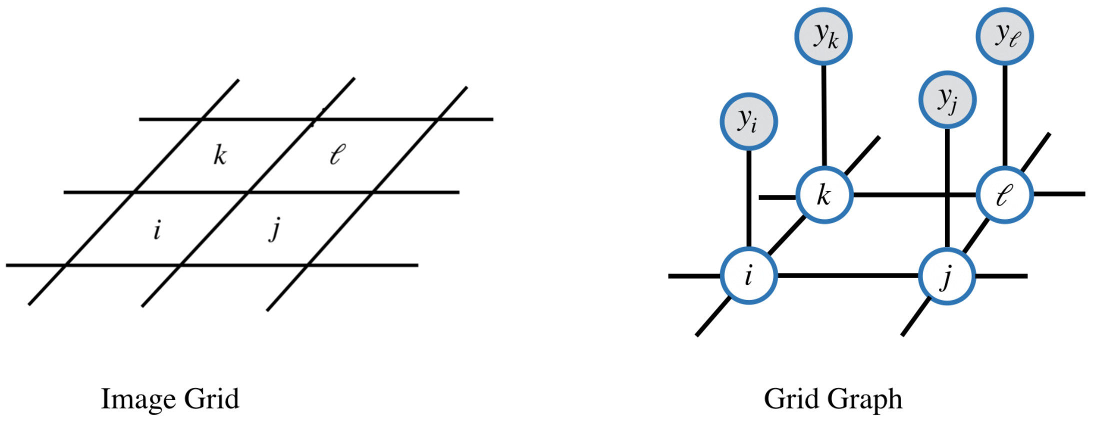


FIGURE 4.8: On the left, we see a small region of an image with pixels i, j, k, ℓ in the image grid. On the right, we see the corresponding nodes in the grid graph. In addition, each node has an observed label which is depicted by the gray colored node.

To formalize this approach, let $\mathcal{I}$ be an $n \times m$ image and let $G_{n,m} = (V, E)$ be an $n \times m$ grid graph that provides a graphical representation of the image. Each pixel in the image corresponds to a node in this graph and the edges connect neighboring pixels as shown in Figure 4.8. Under the assumption that $\mathcal{I}$ is an 8-bit image, the label space $\Omega = \{0, \ldots, 255\}$ is the set of possible pixel intensities. Let $x = (x_1, \ldots, x_{nm}) \in \Omega^{nm}$ denote a labelling of the graph and note that a labelling defines an image. Let $y = (y_1 \ldots, y_{mn}) \in \Omega^{nm}$ denote the observed (corrupted) image. The cost functions used in this application are

$$g_i(x_i) = |x_i - y_i|^2 \quad \text{and} \quad h_{ij}(x_i, x_j) = \lambda \min\big(|x_i - x_j|^2, \tau\big).$$

The function g_i enforces that each pixel in the restored image is *consistent* with the observed image. The pairwise function h_{ij} enforces that the restored image is piece-wise *smooth* with spatially coherent regions. The quadratic difference in h_{ij} is bounded by $\tau \in \Omega$ to allow for large differences between neighboring pixels, which occurs when two neighbors belong to different objects in the image. The parameter $\lambda > 0$ controls how much weight is placed on the consistency versus smoothing terms in the energy, where large values of λ result in smoother images.

Under this framework, an approximation to the clean image can be obtained by minimizing the Gibbs energy given by

$$E(x) = \sum_{i \in V} |x_i - y_i|^2 + \lambda \sum_{\{i,j\} \in E} \min\big(|x_i - x_j|^2, \tau\big).$$

For our experiments we used the RGB image shown on the left hand side of Figure 4.9. This image is composed of three color channels, where each channel is an 8-bit image with dimensions 400×466. We restore the RGB image by separately restoring each of the three color channels.

We generated a corrupted version of the image by adding independent noise to each pixel, by sampling from a Gaussian distribution with mean $\mu = 0$ and standard deviation $\sigma = 50$. In Figure 4.9, we show the original image on the left and a corrupted version of this image on the right. The weights used in the operator $\hat{Q}$ were set uniformly and we set $\gamma = 0.99$. The parameters in our image model were set as $\tau = 100$ and we tried several values of λ. We applied convex combination belief propagation to restore each channel of the corrupted image and the algorithm converged after at most 8 iterations.

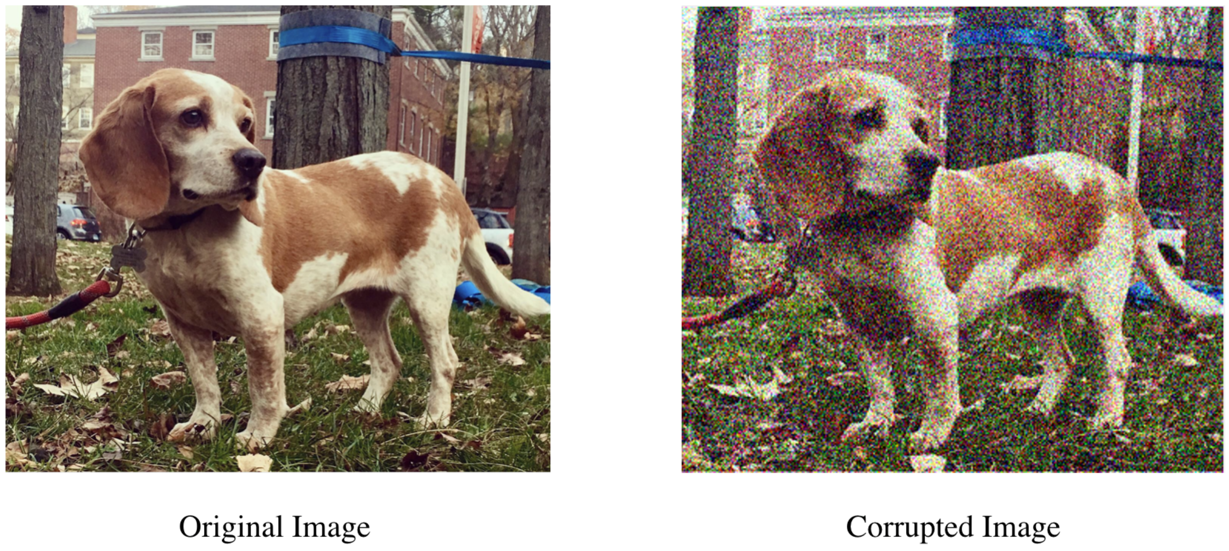


Figure 4.9: Data used in the image restoration experiment.

A crucial component of this model is choosing a good value for the smoothing parameter λ. We obtained the best results with $\lambda = 3$ as shown in Figure 4.11. The restored image is smooth almost everywhere while also preserving sharp discontinuities at the boundary of different objects in the image. The image restoration results in Figures 4.10 and 4.12 show the effect of using a value of λ that is too small and too big, respectively. When this parameter is too small, the resulting image still contains a significant amount of noise as seen in the zoomed section in Figure 4.10. When the magnitude of the λ is too large, the restored image appears blurry and the boundaries between objects are fuzzy as seen in the zoomed in section.

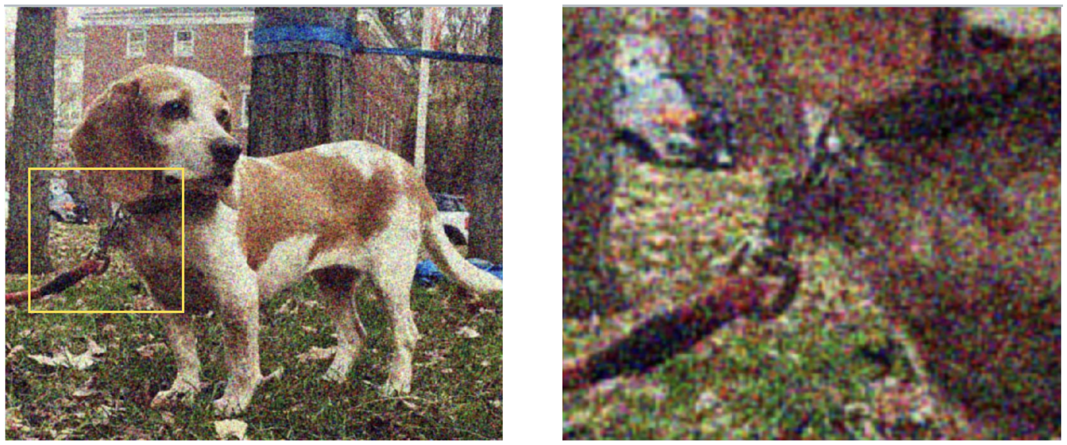


FIGURE 4.10: Image restoration using convex combination belief propagation with $\lambda = 1$.

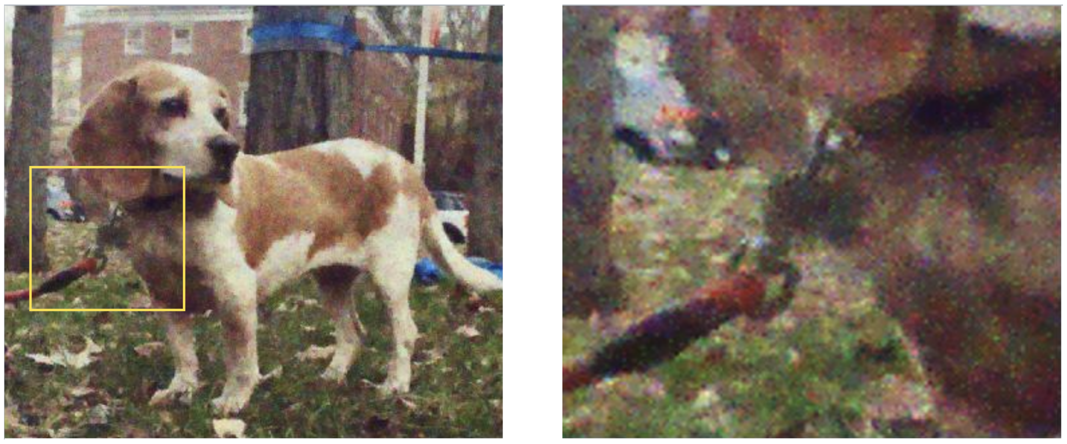


FIGURE 4.11: Image restoration using convex combination belief propagation with $\lambda = 4$.

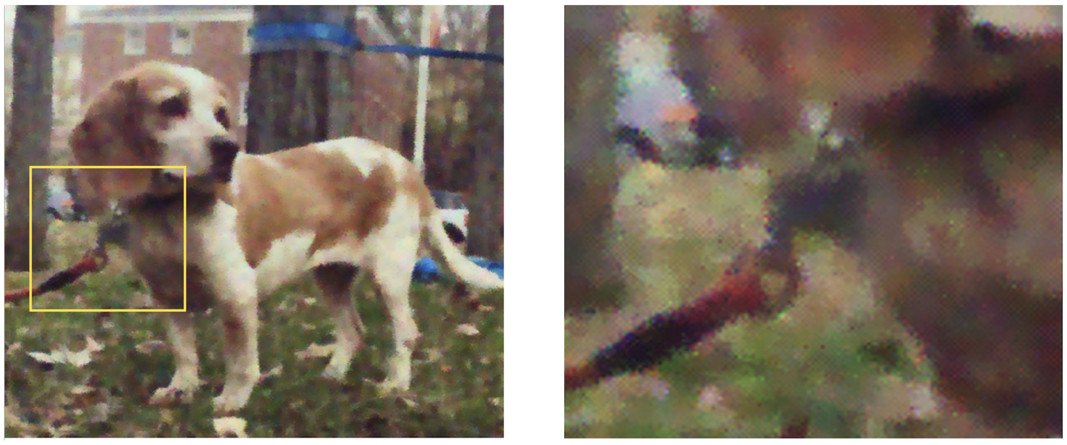


FIGURE 4.12: Image restoration using convex combination belief propagation with $\lambda = 10$.

4.5 Sum-Product Algorithm

The main objective of this section is present the sum-product version of convex combination belief propagation. We introduce the message passing operator in Section 4.5.2, then prove that the algorithm converges in Section 4.5.3.

4.5.1 Theoretical Settings

Let $G = (V, E)$ be an undirected graph with $V = \{1, \ldots, n\}$. Let $X = (X_1, \ldots, X_n)$ be a random vector and $\Omega = \{1, \ldots, N\}$ be the set of possible outcomes (labels) for each random variable. Let the joint distribution be

$$\mathbb{P}(X = x) = \frac{1}{Z} \prod_{i \in V} \phi_i(x_i) \prod_{\{i,j\} \in E} \psi_{ij}(x_i, x_j),$$

where ϕ_i and ψ_{ij} are assumed to be positive functions and Z is a normalization constant. Note that this graphical model is a pairwise Markov random field since the joint distribution consists of unary and binary potentials

4.5.2 Message Passing Operator

The operator used in the sum-product version of convex combination belief propagation is analogous to the the min-sum case. The incoming messages are also "weighted" so that each node receives a convex combination of messages from its neighbors. One difference is that the sum-product algorithm the combination of messages involves a weighted geometric average.

Definition 4.5.1. *The sum-product operator $\hat{S} : \mathcal{M} \to \mathcal{M}$ in convex combination belief propagation is*

$$\left(\hat{S}m\right)_{ij}(x_j) = \sum_{x_i} \Big(\phi_i(x_i) \psi_{ij}(x_i, x_j) \Big(\prod_{k \in \mathcal{N}(i) \backslash j} m_{ki}(x_i)^{w_{ki}} \Big)^{\gamma} \Big),$$

where the weights in the sum must satisfy $\sum_k w_{ki} \leq 1$ with $w_{ki} > 0$ and $\gamma \in (0, 1)$.

Note that we do not incorporate a normalization in the definition of the operator. The main purpose of the normalization factor in loopy belief propagation is to prevent numerical underflow, but our algorithm does not suffer from this problem.

4.5.3 Convergence of the Algorithm

Our approach to proving that belief propagation with the message passing operator $\hat{S}$ converges is analogous to the argument given in Section 4.2.3. The main idea is to show that $\hat{S}$ is contractive, then use this fact to conclude that there exists a unique and strongly attracting fixed point by using Banach's Fixed Point Theorem.

In the next lemma, we define a distance function over the set of positive reals and prove that it is a metric. Then we extend this distance function into a metric that is defined over the product space $\mathcal{M}$.

Lemma 4.5.2. *Let $f : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}$ be the distance function given by*

$$f(x_i, x_j) = |\log x_i - \log x_j|$$

for any $x_i, x_j \in \mathbb{R}_+$, then the pair $(\mathbb{R}_+, f)$ is a complete metric space.

Proof. It is clear that f is non-negative, symmetric, and that $d(x_i, x_j) = 0$ if and only if $x_i = x_j$. The triangle inequality holds for any $x_i, x_j, x_k \in \mathbb{R}_+$ by

$$\begin{aligned} f(x_i, x_j) &= |\log x_i - \log x_j| \\ &= |\log x_i - \log x_k + \log x_k - \log x_j| \\ &\leq |\log x_i - \log x_k| + |\log x_k - \log x_j| \\ &= f(x_i, x_k) + f(x_k, x_j). \end{aligned}$$

To show that this space is complete, choose any Cauchy sequence $\{x_n\} \subset \mathbb{R}_+$ and note that this sequence can be written as $\{x_n\} = \{e^{y_n}\}$ with $y_n = \log(x_n)$. The sequence $\{y_n\} \subset \mathbb{R}$ must be Cauchy with respect to the Euclidean metric because for any $\epsilon > 0$ there exists an $N > 0$ such that $f(x_n, x_m) < \epsilon$ for all $n, m > N$, which implies that

$$|y_n - y_m| = |\log x_n - \log x_m| = f(x_n, x_m) < \epsilon.$$

Since $\{y_n\}$ is a Cauchy sequence in a complete space, there exists some $y \in \mathbb{R}$ such that $y_n \to y$ which implies that $x_n = e^{y_n} \to e^y$. □

Proposition 4.5.3. *Let $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ be the distance function given by*

$$d(m,n) = \max_{i \in V} \max_{j \in \mathcal{N}(i)} \max_{x_j} \big| \log m_{ij}(x_j) - \log n_{ij}(x_j) \big|,$$

then the pair $\big(\mathcal{M}, d\big)$ is a complete metric space.

Proof. It is clear that d is non-negative, symmetric, and that $d(m,n) = 0$ if and only if $m = n$. To show the triangle inequality, choose any $m, n, p \in \mathcal{M}$ and observe that

$$\begin{aligned} d(m,n) &= \max_{i \in V} \max_{j \in \mathcal{N}(i)} \max_{x_j} \big| \log m_{ij}(x_j) - \log n_{ij}(x_j) \big| \\ &\le \max_{i \in V} \max_{j \in \mathcal{N}(i)} \max_{x_j} \Big(\big| \log m_{ij}(x_j) - \log p_{ij}(x_j) \big| + \big| \log p_{ij}(x_j) - \log n_{ij}(x_j) \big| \Big) \\ &\le \max_{i \in V} \max_{j \in \mathcal{N}(i)} \max_{x_j} \big| \log m_{ij}(x_j) - \log p_{ij}(x_j) \big| + \max_{i \in V} \max_{j \in \mathcal{N}(i)} \max_{x_j} \big| \log p_{ij}(x_j) - \log n_{ij}(x_j) \big| \\ &= d(m,p) + d(p,n). \end{aligned}$$

Now choose any Cauchy sequence $\{m^{(t)}\} \subset \mathcal{M}$, then $\{m_{ij}^{(t)}(x_j)\} \subset \mathbb{R}_+$ is a Cauchy sequence in $(\mathbb{R}_+, f)$ because for any $\epsilon > 0$ there exists an $N > 0$ such that for all $s, t > N$

$$\begin{aligned} f\big(m_{ij}^{(t)}(x_j),\, m_{ij}^{(s)}(x_j)\big) &= \big| \log m_{ij}^{(t)}(x_j) - \log m_{ij}^{(s)}(x_j) \big| \\ &\le \max_{i \in V} \max_{j \in \mathcal{N}(i)} \max_{x_j} \big| \log m_{ij}^{(t)}(x_j) - \log m_{ij}^{(s)}(x_j) \big| \\ &= d\big(m^{(t)}, m^{(s)}\big) \\ &< \epsilon \end{aligned}$$

Given that the pair $(\mathbb{R}_+, f)$ is a complete metric space by Lemma 4.5.2, there exists some m such that $m_{ij}^{(t)}(x_j) \to m_{ij}(x_j)$. Thus, the space $(\mathcal{M}, d)$ is complete because $m^{(t)} \to m \in \mathcal{M}$ by

$$d(m^{(t)}, m) = \max_{i \in V} \max_{j \in \mathcal{N}(i)} \max_{x_j} f\big(m_{ij}^{(t)}(x_j), m_{ij}(x_j)\big) \to 0.$$

□

Now that this problem is set up under the framework of a complete metric space, we prove that $\hat{S}$ has a unique and globally attracting fixed point by showing that this operator is contractive.

Lemma 4.5.4. *The operator $\hat{S}$ is contractive with Lipschitz constant γ.*

Proof. Choose any $m, n \in \mathcal{M}$, then

$$\begin{aligned}
\big(\hat{S}m\big)_{ij}(x_j) &= \sum_{x_i} \phi_i(x_i)\psi_{ij}(x_i,x_j) \prod_{k\in\mathcal{N}(i)\backslash j} m_{ki}(x_i)^{\gamma w_{ki}} \\
&= \sum_{x_i} \phi_i(x_i)\psi_{ij}(x_i,x_j) \prod_{k\in\mathcal{N}(i)\backslash j} n_{ki}(x_i)^{\gamma w_{ki}} \prod_{k\in\mathcal{N}(i)\backslash j} \frac{m_{ki}(x_i)^{\gamma w_{ki}}}{n_{ki}(x_i)^{\gamma w_{ki}}} \\
&\le \left(\sum_{x_i} \phi_i(x_i)\psi_{ij}(x_i,x_j) \prod_{k\in\mathcal{N}(i)\backslash j} n_{ki}(x_i)^{\gamma w_{ki}}\right)\left(\max_{x_i} \prod_{k\in\mathcal{N}(i)\backslash j} \frac{m_{ki}(x_i)^{\gamma w_{ki}}}{n_{ki}(x_i)^{\gamma w_{ki}}}\right) \\
&= \big(\hat{S}n\big)_{ij}(x_j)\ \max_{x_i} \prod_{k\in\mathcal{N}(i)\backslash j} \frac{m_{ki}(x_i)^{\gamma w_{ki}}}{n_{ki}(x_i)^{\gamma w_{ki}}}.
\end{aligned}$$

Taking the logarithm of both sides yields that

$$\begin{aligned}
\log\big(\hat{S}m\big)_{ij}(x_j) &\le \log\left(\big(\hat{S}n\big)_{ij}(x_j)\ \max_{x_i} \prod_{k\in\mathcal{N}(i)\backslash j} \frac{m_{ki}(x_i)^{\gamma w_{ki}}}{n_{ki}(x_i)^{\gamma w_{ki}}}\right) \\
&= \log\big(\hat{S}n\big)_{ij}(x_j) + \gamma \max_{x_i} \sum_{k\in\mathcal{N}(i)\backslash j} w_{ki} \log \frac{m_{ki}(x_i)}{n_{ki}(x_i)} \\
&\le \log\big(\hat{S}n\big)_{ij}(x_j) + \gamma \max_{x_i} \sum_{k\in\mathcal{N}(i)\backslash j} w_{ki} \left|\log \frac{m_{ki}(x_i)}{n_{ki}(x_i)}\right|
\end{aligned}$$

$$\implies \log\big(\hat{S}m\big)_{ij}(x_j) - \log\big(\hat{S}n\big)_{ij}(x_j) \le \gamma \max_{x_i} \sum_{k\in\mathcal{N}(i)\backslash j} w_{ki} \left|\log \frac{m_{ki}(x_i)}{n_{ki}(x_i)}\right|.$$

Since this inequality holds when m and n are interchanged, we can take the absolute value of the left hand side. Moreover, given that the above inequality holds for any $x_j \in \Omega$, it must hold for the maximum over x_j the left hand side.

$$\max_{x_j} \Big| \log \frac{(\hat{S}m)_{ij}(x_j)}{(\hat{S}n)_{ij}(x_j)} \Big| \leq \gamma \max_{x_i} \sum_{k \in \mathcal{N}(i) \setminus j} w_{ki} \Big| \log \frac{m_{ki}(x_i)}{n_{ki}(x_i)} \Big|$$
$$\leq \gamma \max_{i \in V} \max_{j \in N(i)} \max_{x_i} \sum_{k \in \mathcal{N}(i) \setminus j} w_{ki} \Big| \log \frac{m_{ki}(x_i)}{n_{ki}(x_i)} \Big|$$
$$\leq \gamma \max_{i \in V} \max_{j \in N(i)} \max_{k \in \mathcal{N}(i) \setminus j} \max_{x_i} \Big| \log \frac{m_{ki}(x_i)}{n_{ki}(x_i)} \Big|$$
$$= \gamma \max_{i \in V} \max_{j \in \mathcal{N}(i)} \max_{x_i} \Big| \log \frac{m_{ij}(x_i)}{n_{ij}(x_i)} \Big|.$$

Since the edge $\{i, j\} \in E$ was chosen arbitrarily from the beginning, the inequality holds for any $\{i, j\} \in E$. The final result is obtained by taking the maximum of the left hand side over all the edges in the graph.

$$\max_{i \in V} \max_{j \in \mathcal{N}(i)} \max_{x_i} \Big| \log \frac{(\hat{S}m)_{ij}(x_j)}{(\hat{S}n)_{ij}(x_j)} \Big| \leq \gamma \max_{i \in V} \max_{j \in \mathcal{N}(i)} \max_{x_i} \Big| \log \frac{m_{ij}(x_i)}{n_{ij}(x_i)} \Big|$$

$$\implies d\big(\hat{S}m, \hat{S}n\big) \leq \gamma\, d(m, n)$$

□

Theorem 4.5.1. *The message passing operator $\hat{S}$ has a unique fixed point $m^\star \in \mathcal{M}$ and the sequence defined by $m^{(t+1)} := \hat{S}m^{(t)}$ converges to $m^\star$. Furthermore, after t iterations*

$$d(\hat{S}^{(t)} m^{(0)}, m^\star) \leq \gamma^t d(m^{(0)}, m^\star).$$

Proof. Given that the pair $(\mathcal{M}, d)$ is a complete metric space by Proposition 4.5.3 and $\hat{S}$ is a contraction by Lemma 4.5.4, then the result holds by applying Banach's fixed point theorem. □

4.6 Discussion

The purpose of this chapter is to introduce convex combination belief propagation which is a convergent alternatives to traditional belief propagation. The primary advantage of this class of algorithms is that they converge to unique fixed points on graphs with arbitrary topology, independent of the initialization of messages. We provided a characterization of the beliefs obtained from the min-sum version of the algorithm in the case when the graph is tree-structured. In addition, we demonstrated the practical application of this algorithm for image restoration.

Although convex combination belief propagation has good theoretical properties, one drawback is that the resulting beliefs may not provide a good approximation of the true min-marginals (or marginal distributions) in the case of the min-sum (or sum-product) algorithm. For example, traditional belief propagation is guaranteed to compute the exact min-marginals (or marginals) when the graph is tree-structured. However, we showed the beliefs obtained with convex combination belief propagation are the min-marginals of a weighted energy function when the graph is tree-structured. Although the beliefs are likely to differ from the true min-marginals (or marginals), the beliefs still provide useful information. To illustrate this point, we provide an example of using convex combination belief propagation to obtain a fixed point in the spin glass model from statistical physics.

Example 4.6.1. *Let $G = (V, E)$ be a complete graph with twelve nodes and let $\Omega = \{-1, 1\}$ be a set of labels. Let $E : \Omega^{12} \to \mathbb{R}$ be the energy*

$$E(x) = \sum_{i \in V} y_i x_i + \sum_{\{i,j\} \in E} \lambda_{ij} x_i x_j$$

each y_i was uniformly sampled from $\{-1, 1\}$ and λ_{ij} was sampled from a standard normal distribution.

Let $\gamma = 0.9$ and $w_{ki} = 1/(|\mathcal{N}(i)| - 1)$ in convex combination belief propagation. Let $\alpha = 0.9$ in damped belief propagation. We initialized the messages as $m_{ij}^{(0)} = (1, 1)$ and applied both belief propagation algorithms to this example. For each algorithm, we computed the belief function of every node along with the true min-marginals of the energy. In Figure 4.13, we show the value of these functions when $x_i = 1$. (Note: each function was shifted to have mean zero.)

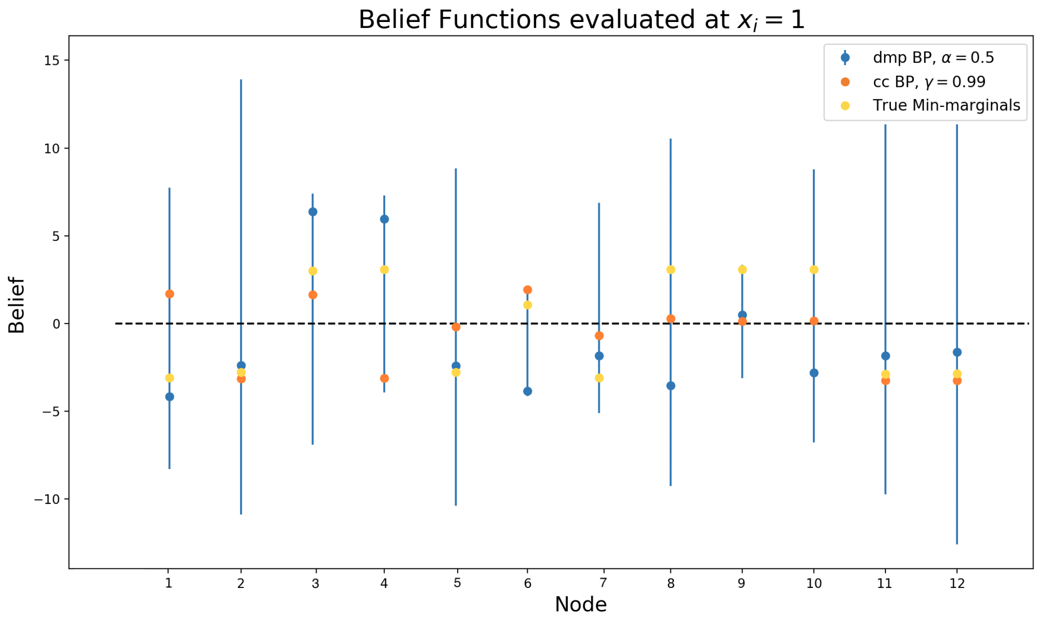


FIGURE 4.13: Beliefs obtained from min-sum belief propagation algorithms and exact min-marginals of the energy from Example 4.6.1. Damped belief propagation (dmp BP) failed to converge and the vertical lines show how the beliefs oscillated in the last 100 iterations before stopping the algorithm after 1000 iterations. Convex combination belief propagation (cc BP) converged as expected, but the resulting beliefs differ from the true min-marginals.

In Figure 4.13, we see that the beliefs obtained with damped belief propagation oscillate despite the damping factor being quite large. In contrast, convex combination belief propagation converged, but some of the beliefs are not a good approximation of the true marginals. We have seen convex combination belief propagation obtain good approximations to min-marginals across many problem instances, but there is no theoretical bound on the quality of this approximation. This is an issue that is shared by both convex combination and damped belief propagation (when damped BP converges).

In damped belief propagation, the message passing operator Q_α can have multiple fixed points. Some fixed points may be attractive while others may be repelling which makes the dynamics of fixed point iteration unstable. Convex combination belief propagation is always guaranteed to converge. Although the beliefs may differ from the true min-marginals, the beliefs provide useful information to find a good approximation of the MAP solution. In this setting, the accuracy of the beliefs is less important because they are only used to choose a good label for each node in the graph.

Consider the problem of obtaining an optimal labelling in Example 4.6.1. Each belief in Figure 4.13 was centered to have mean zero. Since the label space consists of exactly two elements each belief function is positive for one label and negative for the other (under the assumption there are no ties between labels). In this framework, the value "0" functions as a decision boundary where favorable labels correspond to negative beliefs. The beliefs obtained with damped belief propagation in Example 4.6.1 are problematic because they oscillate. In some special cases of oscillations, the beliefs can be useful as long as they choose the same label for the entire period of the oscillation. However, the beliefs in Example 4.6.1 do not fall into this category. Instead these beliefs are uninformative because they oscillate about the decision boundary. Given these circumstances it is difficult to confidently choose a labelling with the beliefs.

In contrast, although the beliefs obtained with convex combination belief propagation differ from the true min-marginals, they agree on the optimal label for all but one node (see Figure 4.13). This example highlights the strength of this algorithm, namely that it converges on difficult problem instances and the beliefs tend to align with the true min-marginals. Thus, one conclusion to draw from this example is that convex combination belief propagation provides useful information in decision based applications.

Next we include a similar example for the sum-product message passing operators.

Example 4.6.2. *Let $G = (V, E)$ be a complete graph with twelve nodes and let $\Omega = \{-1, 1\}$ be a set of labels. Let the joint distribution of this model be*

$$\mathbb{P}(X = x) = \frac{1}{Z} \exp\left(-\sum_{i \in V} y_i x_i - \sum_{\{i,j\} \in E} \lambda_{ij} x_i x_j \right)$$

y_i was uniformly sampled from $\{-1, 1\}$ and λ_{ij} was sampled from a standard normal distribution.

Let $\gamma = 0.9$ and $w_{ki} = 1/(|\mathcal{N}(i)| - 1)$ in convex combination belief propagation and let $\alpha = 0.9$ in damped belief propagation. We used the initial set of messages $m_{ij}^{(0)} = (1, 1)$, then applied damped and convex combination belief propagation. For each algorithm, we computed beliefs of every node along with the true marginals. In Figure 4.14, we show the value of these functions when $x_i = 1$.

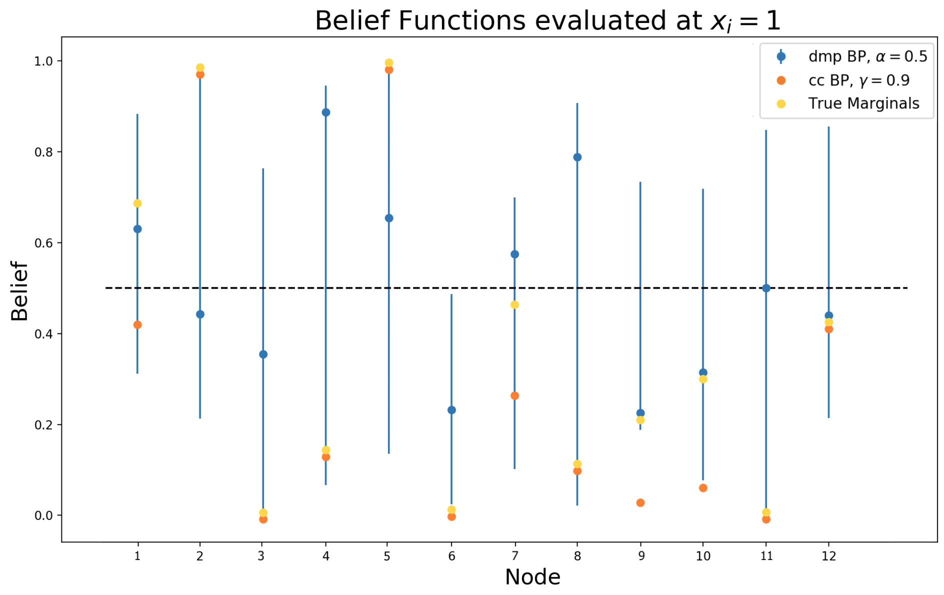


FIGURE 4.14: Beliefs obtains from sum-product belief propagation and exact marginals from Example 4.6.2. Damped belief propagation failed to converge and the vertical lines show how the belief functions oscillated in the last 100 iterations before stopping the algorithm after 1000 iterations. Convex combination belief propagation converged as expected, but the resulting belief functions differ from the true marginals.

In this framework, a common objective is to obtain the most probable label for each node. A probable label for node i is obtained from by maximizing its belief. Given there are two possible labels and the beliefs are normalized to add to one, the value “0.5” functions as a decision boundary. In Figure 4.14 we see that the beliefs obtained from damped belief propagation are uninformative because they oscillate about the decision boundary. In contrast, convex combination belief propagation provides a conclusive result and agrees with the true marginals on all but two labels.

4.7 Conclusion

We conclude by noting that convex combination belief propagation is not only guaranteed to converge on arbitrary graphs but also converges quickly. As discussed above the resulting beliefs provide useful information in the context of decision making, a central component of an intelligence system. Many applications require powerful algorithms that can handle complex and large scale data sets. Convex combination belief propagation is a natural fit for these applications because it is designed to converge on the most challenging problems.

Although this algorithm has potential useful applications, we improve the accuracy of this algorithm by utilizing homotopy continuation. Since the weights in the message passing operators affects the accuracy, we use homotopy continuation to gradually phase out the weights. Homotopy continuation is the main focus of Chapters 6 and 7, before introducing this algorithm we generalize convex combination belief propagation to factor graphs.

CHAPTER 5

Convex Combination Belief Propagation on Factor Graphs

This chapter presents a generalization of convex combination belief propagation to factor graphs. In Sections 5.2 and 5.3, we introduce the message passing operators used in the sum-product and max-product algorithm, respectively. Then we conclude the chapter with some discussion of the shortcomings of convex combination belief propagation in Chapter 5.4.

5.1 Theoretical Settings

Let $G = (V \cup F, E)$ be an undirected bipartite graph with the set of variable nodes $V = \{1, \dots, n\}$ and factor nodes $F = \{1, \dots, m\}$. In this chapter, variable and factor nodes are concisely referred to as *variables* and *factors*, respectively. Let $X = (X_1, \dots, X_n)$ be a random vector and let $\Omega = \{1, \dots, N\}$ be the set of possible outcomes of each random variable. Let the joint distribution of the factor graph be

$$\mathbb{P}(X = x) = \frac{1}{Z} \prod_{f \in F} \Psi_f(x_{\mathcal{N}(f)}),$$

where Ψ_f is assumed to be positive.

5.2 Sum-Product Algorithm

The main objective of this section is to present the factor graph version of the sum-product algorithm in convex combination belief propagation. In Section 5.2.1, we define the message passing operators and the message update scheme. The remainder of this section focuses on proving that the algorithm is guaranteed to converge. We define a metric on the space of messages in Section 5.2.2, then prove that the normalization operator is Lipschitz with respect to an equivalent metric in Section 5.2.3. In Section 5.2.4, we prove that the algorithm is guaranteed to converge to a unique fixed point.

5.2.1 Message Passing Operators

Next we generalize convex combination belief propagation to factor graphs, then prove that the algorithm is guaranteed to converge to a unique fixed point. Belief propagation on factor graphs is more complex since it involves message passing between two difference types of nodes. In order to ensure that the algorithm converges, we use edge-based weights in both message passing operators.

One of the operators also incorporates a normalization to ensures that the messages sum to one. Normalization is not necessary to guarantee that the algorithm converges. However, it is needed to prove certain properties about the homotopy continuation algorithm in the next chapter. We include the normalization operator in this chapter in order to lay the theoretical foundation for the homotopy continuation algorithm.

The addition of the normalization operator also implies that its output is an vector in a high dimensional simplex. Let $\mathcal{K}$ be the simplex given by

$$\mathcal{K} = \bigotimes_{i \in V} \bigotimes_{f \in \mathcal{N}(i)} \Delta^N,$$

where Δ^N is the N-simplex.

Definition 5.2.1. *The message passing operator $\hat{T} : \mathcal{K} \to \mathcal{M}$ in sum-product convex combination belief propagation is given by*

$$\left(\hat{T}\mu\right)_{i\to f}(x_i) = \prod_{g\in\mathcal{N}(i)\backslash f} \mu_{g\to i}(x_i)^{\gamma w_{g\to i}},$$

where the weights in the sum must satisfy $\sum\limits_g w_{g\to i} \leq 1$ and $\gamma \in (0, 1/2)$.

Definition 5.2.2. *The message passing operator $\hat{S} : \mathcal{M} \to \mathcal{K}$ in sum-product belief propagation is $\hat{S} = N\hat{R}$. $\hat{R}$ computes a new message using a weighted version of the sum-product equation and N normalizes each message.*

$$\left(\hat{R}\nu\right)_{f\to i}(x_i) = \sum_{x_{\mathcal{N}(f)}} \Psi_f(x_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)\backslash i} \nu_{j\to f}(x_j)^{w_{j\to f}}$$

$$\left(N\nu\right)_{f\to i}(x_i) = \frac{\nu_{i\to f}(x_i)}{\sum\limits_{\tau\in\Omega} \nu_{i\to f}(\tau)},$$

where the weights in the sum must satisfy $\sum\limits_j w_{j\to f} \leq 1$.

The simplest way to define each weight is to set $w_{g\to i} = 1/\big(|\mathcal{N}(i)| - 1\big)$ and $w_{j\to f} = 1/\big(|\mathcal{N}(f)| - 1\big)$. Alternatively one can give more weight to certain edges based on some additional information from a particular application (see [20]). The operator $\hat{T}$ also includes a damping factor γ. Later in this section, we see that this parameter is necessary to guarantee convergence. In addition, the magnitude of this value controls the rate of convergence.

In traditional belief propagation, the messages are initialized and then repeatedly updated with T and S until both sets of messages converge. Our algorithm could utilize the same update scheme with $\hat{T}$ and $\hat{S}$. However, this scheme is inefficient in terms of memory usage. Instead we argue that it's better to initialize one set of messages $\mu^{(0)} \in \mathcal{K}$, then repeatedly update the messages with the composition operator $\hat{\Theta} = N\hat{S}\hat{T}$.

Definition 5.2.3. *The message passing operator $\hat{\Theta} : \mathcal{K} \to \mathcal{K}$ in sum-product convex combination belief propagation is $\hat{\Theta} = N\hat{S}\hat{T}$ given by*

$$\left(\hat{S}\hat{T}\mu\right)_{f\to i}(x_i) = \sum_{x_{\mathcal{N}(f)}} \Psi_f(x_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)\backslash i} \Big(\prod_{g\in N(j)\backslash f} \mu_{g\to j}(x_j)^{\gamma w_{g\to j}} \Big)^{w_{j\to f}}.$$

The composition operator can be defined with either $\hat{T}\hat{S}$ or $\hat{S}\hat{T}$. We use the latter because the beliefs are ultimately computed with the output of this operator. The message passing scheme defined by $\hat{\Theta}$ is equivalent to the scheme used in traditional belief propagation. But the advantage of is that it requires half as much memory, since only one set of messages is stored on each iteration. The composition operator exploits that each message update $\mu^{(n+1)}$ can be directly computed from $\mu^{(n)}$ without accessing $\nu^{(n)}$ from memory. Alternatively, the message $\nu^{(n)}_{j\to f}$ is computed as an intermediate step, then immediately used to obtain the update $\mu^{(n+1)}_{f\to i}$. Moreover, it is unnecessary to compute and store both sets of messages because their convergence is interdependent.

5.2.2 Message Passing in a Metric Space

The main objective of this section is to prove that the sum-product version of convex combination belief propagation is guaranteed to converge on factor graphs. We use Banach's fixed point theorem to prove this result, which requires that the problem is formulated under the framework of a metric space. In the next lemma, we define a distance function over the set of positive reals and prove that this pair is a complete metric space. Then we extend this distance function into a metric that is defined over the product space $\mathcal{K}$.

Lemma 5.2.4. *Let $f : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}$ be the distance function given by*

$$f(x_i, x_j) = |\log x_i - \log x_j|$$

for any $x_i, x_j \in \mathbb{R}_+$, then the pair $(\mathbb{R}_+, f)$ is a complete metric space.

Proof. It is clear that f is non-negative, symmetric, and that $d(x_i, x_j) = 0$ if and only if $x_i = x_j$. The triangle inequality holds for any $x_i, x_j, x_k \in \mathbb{R}_+$ by

$$\begin{aligned} f(x_i, x_j) &= |\log x_i - \log x_j| \\ &= |\log x_i - \log x_k + \log x_k - \log x_j| \\ &\le |\log x_i - \log x_k| + |\log x_k - \log x_j| \\ &= f(x_i, x_k) + f(x_k, x_j). \end{aligned}$$

To show that this space is complete, choose any Cauchy sequence $\{x_n\} \subset \mathbb{R}_+$ and note that this sequence can be written as $\{x_n\} = \{e^{y_n}\}$ with $y_n = \log(x_n)$. The sequence $\{y_n\} \subset \mathbb{R}$ is Cauchy with respect to the Euclidean metric because for any $\epsilon > 0$ there exists an $N > 0$ such that $f(x_n, x_m) < \epsilon$ for all $n, m > N$, which implies that

$$|y_n - y_m| = |\log x_n - \log x_m| = f(x_n, x_m) < \epsilon.$$

Since $\{y_n\}$ is a Cauchy sequence in a complete space, there exists some $y \in \mathbb{R}$ such that $y_n \to y$ which implies that $x_n = e^{y_n} \to e^y$. $\square$

Proposition 5.2.5. *Let $d : \mathcal{K} \times \mathcal{K} \to \mathbb{R}$ be the distance function given by*

$$d(\mu, \lambda) = \max_{i \in V} \max_{f \in \mathcal{N}(i)} \max_{x_i} \big| \log \mu_{f \to i}(x_i) - \log \lambda_{f \to i}(x_i) \big|$$

for any $\mu, \lambda \in \mathcal{K}$, then the pair $\big(\mathcal{K}, d\big)$ is a complete metric space.

Proof. It is clear that d is non-negative, symmetric, and that $d(\mu, \lambda) = 0$ if and only if $\mu = \lambda$. The triangle inequality holds for any $\mu, \lambda, \psi \in \mathcal{K}$ by

$$\begin{aligned}
d(\mu, \lambda) &= \max_{i \in V} \max_{f \in \mathcal{N}(i)} \max_{x_i} \big| \log \mu_{f \to i}(x_i) - \log \lambda_{f \to i}(x_i) \big| \\
&\leq \max_{i \in V} \max_{f \in \mathcal{N}(i)} \max_{x_i} \Big(\big| \log \mu_{f \to i}(x_i) - \log \psi_{f \to i}(x_i) | + | \log \psi_{f \to i}(x_i) - \log \lambda_{f \to i}(x_i \big| \Big) \\
&\leq \max_{i \in V} \max_{f \in \mathcal{N}(i)} \max_{x_i} \big| \log \mu_{f \to i}(x_i) - \log \psi_{f \to i}(x_i) | + \max_{i \in V} \max_{f \in \mathcal{N}(i)} \max_{x_i} | \log \psi_{f \to i}(x_i) - \log \lambda_{f \to i}(x_i) \big| \\
&= d(\mu, \psi) + d(\psi, \lambda).
\end{aligned}$$

Given any Cauchy sequence $\{\mu^{(n)}\} \subset \mathcal{K}$, then $\{\mu^{(n)}_{f\to i}(x_i)\} \subset \mathbb{R}_+$ is a Cauchy sequence in $(\mathbb{R}_+, f)$ because for any $\epsilon > 0$ there exists an $N > 0$ such that for all $n, m > N$

$$\begin{aligned} f\big(\mu^{(n)}_{f\to i}(x_i),\, \mu^{(m)}_{f\to i}(x_i)\big) &= \big|\log \mu^{(n)}_{f\to i}(x_i) - \log \mu^{(m)}_{f\to i}(x_i)\big| \\ &\le \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{x_i} \big|\log \mu^{(n)}_{f\to i}(x_i) - \log \mu^{(m)}_{f\to i}(x_i)\big| \\ &= d\big(\mu^{(n)}, \mu^{(m)}\big) \\ &< \epsilon. \end{aligned}$$

Given that the pair $(\mathbb{R}_+, f)$ is a complete metric space by Lemma 5.2.4, there exists some $\mu \in \mathcal{K}$ such that $\mu^{(n)}_{f\to i}(x_i) \to \mu_{f\to i}(x_i)$. Thus, the space $(\mathcal{K}, d)$ is complete because $\mu^{(n)} \to \mu \in \mathcal{K}$ by

$$d(\mu^{(n)}, \mu) = \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{x_i} f\big(\mu^{(n)}_{f\to i}(x_j), \mu_{f\to i}(x_i)\big) \to 0.$$

□

5.2.3 Lipschitz Continuity of the Normalization Operator

In this section, we prove that the normalization operator N is 2-Lipschitz with respect to the metric d. It is difficult to directly prove this fact using the definition of the metric. Instead we circumvent this issue by defining an equivalent metric d', then prove that N is Lipschitz in this metric. A metric d' is said to be equivalent to another metric d if there exists constants $\alpha, \beta > 0$ such that

$$\alpha\, d(\mu, \lambda) \le d'(\mu, \lambda) \le \beta\, d(\mu, \lambda).$$

for any $\mu, \lambda \in \mathcal{K}$. This notion is useful because N being Lipschitz in d' implies that it must also be Lipschitz with respect to any equivalent metric. However, one important note is that the Lipschitz constant often differs between equivalent metrics.

Proposition 5.2.6. *Let $d' : \mathcal{K} \times \mathcal{K} \to \mathbb{R}$ be the distance function given by*

$$d'(\mu, \lambda) = \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{\tau,\upsilon\in\Omega} \left| \log \frac{\mu_{f\to i}(\tau)}{\mu_{f\to i}(\upsilon)} - \log \frac{\lambda_{f\to i}(\tau)}{\lambda_{f\to i}(\upsilon)} \right|,$$

for any $\mu, \lambda \in \mathcal{K}$, then the pair $(\mathcal{K}, d')$ is a metric space.

Proof. It is clear that d' is non-negative, symmetric, and that $d'(\mu,\lambda) = 0$ if and only if $\mu = \lambda$. The triangle inequality holds for any $\mu, \lambda, \psi \in \mathcal{K}$ by

$$\begin{aligned}
d'(\mu,\lambda) &= \max_{i\in V}\max_{f\in\mathcal{N}(i)}\max_{\tau,\upsilon}\Big|\log\frac{\mu_{f\to i}(\tau)}{\mu_{f\to i}(\upsilon)} - \log\frac{n_{f\to i}(\tau)}{n_{f\to i}(\upsilon)}\Big| \\
&= \max_{i\in V}\max_{f\in\mathcal{N}(i)}\max_{\tau,\upsilon}\Big|\log\frac{\mu_{f\to i}(\tau)}{\mu_{f\to i}(\upsilon)} - \log\frac{\psi_{f\to i}(\tau)}{\psi_{f\to i}(\upsilon)} + \log\frac{\psi_{f\to i}(\tau)}{\psi_{f\to i}(\upsilon)} - \log\frac{\lambda_{f\to i}(\tau)}{\lambda_{f\to i}(\upsilon)}\Big| \\
&\leq \max_{i\in V}\max_{f\in\mathcal{N}(i)}\max_{\tau,\upsilon}\left(\Big|\log\frac{\mu_{f\to i}(\tau)}{\mu_{f\to i}(\upsilon)} - \log\frac{\psi_{f\to i}(\tau)}{\psi_{f\to i}(\upsilon)}\Big| + \Big|\log\frac{\psi_{f\to i}(\tau)}{\psi_{f\to i}(\upsilon)} - \log\frac{\lambda_{f\to i}(\tau)}{\lambda_{f\to i}(\upsilon)}\Big|\right) \\
&\leq \max_{i\in V}\max_{f\in\mathcal{N}(i)}\max_{\tau,\upsilon}\Big|\log\frac{\mu_{f\to i}(\tau)}{\mu_{f\to i}(\upsilon)} - \log\frac{\psi_{f\to i}(\tau)}{\psi_{f\to i}(\upsilon)}\Big| + \max_{i\in V}\max_{f\in\mathcal{N}(i)}\max_{\tau,\upsilon}\Big|\log\frac{\psi_{f\to i}(\tau)}{\psi_{f\to i}(\upsilon)} - \log\frac{\lambda_{f\to i}(\tau)}{\lambda_{f\to i}(\upsilon)}\Big| \\
&= d(\mu,\psi) + d(\psi,\lambda).
\end{aligned}$$

□

Lemma 5.2.7. *Given any $\mu, \lambda \in \mathcal{K}$, then there exists some $\upsilon \in \Omega$ such that*

$$\max_{\tau}\Big|\log\frac{\mu_{f\to i}(\tau)}{\lambda_{f\to i}(\tau)}\Big| \leq \max_{\tau}\Big|\log\frac{\mu_{f\to i}(\tau)}{\lambda_{f\to i}(\tau)} - \log\frac{\mu_{f\to i}(\upsilon)}{\lambda_{f\to i}(\upsilon)}\Big|$$

when $\mu_{f\to i}$ and $\lambda_{f\to i}$ are both normalized.

Proof. Assume, without loss of generality, that

$$\tau^\star := \arg\max_{\tau}\Big|\log\frac{\mu_{f\to i}(\tau)}{\lambda_{f\to i}(\tau)}\Big| \quad \text{and} \quad \log\frac{\mu_{f\to i}(\tau^\star)}{\lambda_{f\to i}(\tau^\star)} > 0.$$

The inequality on the right implies that $\lambda_{f\to i}(\tau^\star) < \mu_{f\to i}(\tau^\star)$, so the claim holds if $\mu_{f\to i}(\upsilon) \leq \lambda_{f\to i}(\upsilon)$ for some $\upsilon \in \Omega$. Conversely, if $\lambda_{f\to i}(\upsilon) < \mu_{f\to i}(\upsilon)$ for every $\upsilon \in \Omega$, then summing this inequality over every $\upsilon \in \Omega$ implies that

$$\sum_{\tau\in\Omega}\lambda_{f\to i}(\tau) < \sum_{\tau\in\Omega}\mu_{f\to i}(\tau) = 1.$$

But this contradicts the constraint that each message must sum to one. □

Proposition 5.2.8. *The metric d' is equivalent to d.*

Proof. Given any $\mu, \lambda \in \mathcal{K}$, then

$$\begin{aligned} d'(\mu,\lambda) &= \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{\tau,\upsilon} \Big| \log \frac{\mu_{f\to i}(\tau)}{\mu_{f\to i}(\upsilon)} - \log \frac{\lambda_{f\to i}(\tau)}{\lambda_{f\to i}(\upsilon)} \Big| \\ &\le \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{\tau} \Big| \log \frac{\mu_{f\to i}(\tau)}{\lambda_{f\to i}(\tau)} \Big| + \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{\upsilon} \Big| \log \frac{\mu_{f\to i}(\upsilon)}{\lambda_{f\to i}(\upsilon)} \Big| \\ &= 2 \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{\tau} \Big| \log \frac{\mu_{f\to i}(\tau)}{\lambda_{f\to i}(\tau)} \Big| \\ &= 2\, d(\mu,\lambda) \end{aligned}$$

The reverse inequality holds by applying Lemma 5.2.7 to

$$\begin{aligned} d(\mu,\lambda) &= \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{\tau} \Big| \log \frac{\mu_{f\to i}(\tau)}{\lambda_{f\to i}(\tau)} \Big| \\ &\le \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{\tau,\upsilon} \Big| \log \frac{\mu_{f\to i}(\tau)}{\mu_{f\to i}(\upsilon)} - \log \frac{\lambda_{f\to i}(\tau)}{\lambda_{f\to i}(\upsilon)} \Big| \\ &= d'(\mu,\lambda). \end{aligned}$$

We obtain the final result by combining the equalities to conclude that

$$d(\mu,\lambda) \le d'(m,n) \le 2\, d(\mu,\lambda).$$

□

Lemma 5.2.9. *The normalization operator N is 1-Lipschitz in the metric d'.*

Proof. Choose any $\mu, \lambda \in \mathcal{K}$, then

$$\begin{aligned} d'\big(N\mu, N\lambda\big) &= \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{\tau} \Bigg| \log \frac{\big(N\mu\big)_{f\to i}(\tau)}{\big(N\lambda\big)_{f\to i}(\tau)} - \log \frac{\big(N\mu\big)_{f\to i}(\upsilon)}{\big(N\lambda\big)_{f\to i}(\upsilon)} \Bigg| \\ &= \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{\tau} \Big| \log \frac{\mu_{f\to i}(\tau)}{\lambda_{f\to i}(\tau)} - \log \frac{\mu_{f\to i}(\upsilon)}{\lambda_{f\to i}(\upsilon)} \Big| \\ &= d(\mu,\lambda) \end{aligned}$$

where the simplification on the second line holds because the normalization constants cancel. □

Proposition 5.2.10. *The normalization operator N is 2-Lipschitz in the metric d.*

Proof. Choose any $\mu, \lambda \in \mathcal{K}$, then

$$\begin{aligned} d\big(N\mu, N\lambda\big) &\leq d'\big(N\mu, N\lambda\big) \\ &= d'(\mu, \lambda) \\ &\leq 2\, d(\mu, \lambda) \end{aligned}$$

which holds by applying Proposition 5.2.9 and Lemma 5.2.8. □

5.2.4 Convergence

Next we prove that $\hat{\Theta}$ is contractive by exploiting that it is a composition of operators. Since each individual operator is Lipschitz, $\hat{\Theta}$ must also be Lipschitz. Although the Lipschitz constant of each operator may be different, composing them results in a Lipschitz constant less than one.

Lemma 5.2.11. *The operator $\hat{T}$ is a contraction with Lipschitz constant γ with respect to the metric d.*

Proof. Choose any $\mu, \lambda \in \mathcal{K}$, then

$$\begin{aligned} \big(\hat{T}\mu\big)_{i\to f}(x_i) &= \prod_{g\in\mathcal{N}(i)\setminus f} \mu_{g\to i}(x_i)^{\gamma w_{g\to i}} \\ &= \left(\prod_{g\in\mathcal{N}(i)\setminus f} \lambda_{g\to i}(x_i)^{\gamma w_{g\to i}}\right)\left(\prod_{g\in\mathcal{N}(i)\setminus f} \frac{\mu_{g\to i}(x_i)^{\gamma w_{g\to i}}}{\lambda_{g\to i}(x_i)^{\gamma w_{g\to i}}}\right) \\ &= \big(\hat{T}\lambda\big)_{i\to f}(x_i) \left(\prod_{g\in\mathcal{N}(i)\setminus f} \frac{\mu_{g\to i}(x_i)^{\gamma w_{g\to i}}}{\lambda_{g\to i}(x_i)^{\gamma w_{g\to i}}}\right) \\ &\leq \big(\hat{T}\lambda\big)_{i\to f}(x_i) \left(\max_{y_i} \prod_{g\in\mathcal{N}(i)\setminus f} \frac{\mu_{g\to i}(y_i)^{\gamma w_{g\to i}}}{\lambda_{g\to i}(y_i)^{\gamma w_{g\to i}}}\right) \end{aligned}$$

Taking the logarithm of both sides yields that

$$\begin{aligned}
\log \big(\hat{T}\mu\big)_{i\to f}(x_i) &\le \log \big(\hat{T}\lambda\big)_{i\to f}(x_i) + \log\left(\max_{y_i} \prod_{g\in\mathcal{N}(i)\setminus f} \frac{\mu_{g\to i}(y_i)^{\gamma w_{g\to i}}}{\lambda_{g\to i}(y_i)^{\gamma w_{g\to i}}}\right) \\
&= \log \big(\hat{T}\lambda\big)_{i\to f}(x_i) + \gamma \sum_{g\in\mathcal{N}(i)\setminus f} w_{g\to i} \max_{y_i} \log \frac{\mu_{g\to i}(y_i)}{\lambda_{g\to j}(y_i)} \\
&\le \log \big(\hat{T}\lambda\big)_{i\to f}(x_i) + \gamma \sum_{g\in\mathcal{N}(i)\setminus f} w_{g\to i} \max_{y_i} \left| \log \frac{\mu_{g\to i}(y_i)}{\lambda_{g\to i}(y_i)}\right| \\
&\le \log \big(\hat{T}\lambda\big)_{i\to f}(x_i) + \gamma \max_{g\in\mathcal{N}(i)\setminus f} \max_{y_i} \left| \log \frac{\mu_{g\to i}(y_i)}{\lambda_{g\to i}(y_i)}\right|,
\end{aligned}$$

where the last inequality holds by using that the weights are a convex combination. Now the inequality can be written as

$$\log \frac{\big(\hat{T}\mu\big)_{i\to f}(x_i)}{\big(\hat{T}\lambda\big)_{i\to f}(x_i)} \le \gamma \max_{g\in\mathcal{N}(i)\setminus f} \max_{y_i} \left| \log \frac{\mu_{g\to i}(y_i)}{\lambda_{g\to j}(y_i)}\right| .$$

Since this inequality holds when μ and λ are interchanged, we can take the absolute value of the left hand side. Moreover, given that the above inequality holds for any $x_i \in \Omega$, it must hold for the maximum of the left hand side.

$$\begin{aligned}
\max_{x_i} \left| \log \frac{\big(\hat{T}\mu\big)_{i\to f}(x_i)}{\big(\hat{T}\lambda\big)_{i\to f}(x_i)}\right| &\le \gamma \max_{g\in\mathcal{N}(i)\setminus f} \max_{y_i} \left| \log \frac{\mu_{g\to i}(y_i)}{\lambda_{g\to i}(y_i)}\right| \\
&\le \gamma \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{y_i} \left| \log \frac{\mu_{f\to i}(y_i)}{\lambda_{f\to i}(y_i)}\right|
\end{aligned}$$

Since the edge $\{i, f\} \in E$ was chosen arbitrarily from the beginning, the inequality holds for any $\{i, f\} \in E$. The final result is obtained by taking the maximum of the left hand side over all the edges in the graph.

$$\max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{x_i} \left| \log \frac{\big(\hat{T}\mu\big)_{i\to f}(x_i)}{\big(\hat{T}\lambda\big)_{i\to f}(x_i)}\right| \le \gamma \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{y_i} \left| \log \frac{\mu_{f\to i}(y_i)}{\lambda_{f\to i}(y_i)}\right|$$

$$\implies d\big(\hat{T}\mu, \hat{T}\lambda\big) \le \gamma\, d(\mu, \lambda)$$

□

Lemma 5.2.12. *The operator $\hat{R}$ is non-expansive with respect to the metric d.*

Proof. Given any $\nu, \lambda \in \mathcal{K}$, then

$$\begin{aligned}
\big(\hat{R}\nu\big)_{f\to i}(x_i) &= \sum_{x_{\mathcal{N}(f)}} \Psi_f(x_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)\setminus i} \nu_{j\to f}(x_j)^{w_{j\to f}} \\
&= \sum_{x_{\mathcal{N}(f)}} \left(\Psi_f(x_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)\setminus i} \lambda_{j\to f}(x_j)^{w_{j\to f}} \prod_{j\in\mathcal{N}(f)\setminus i} \frac{\nu_{j\to f}(x_j)^{w_{j\to f}}}{\lambda_{j\to f}(x_j)^{w_{j\to f}}} \right) \\
&\leq \left(\sum_{x_{\mathcal{N}(f)}} \Psi_f(x_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)\setminus i} \lambda_{j\to f}(x_j)^{w_{j\to f}} \right) \left(\max_{y_j} \prod_{j\in\mathcal{N}(f)\setminus i} \frac{\nu_{j\to f}(y_j)^{w_{j\to f}}}{\lambda_{j\to f}(y_j)^{w_{j\to f}}} \right) \\
&= \big(\hat{R}\lambda\big)_{f\to i}(x_i) \left(\max_{y_j} \prod_{j\in\mathcal{N}(f)\setminus i} \frac{\nu_{j\to f}(y_j)^{w_{j\to f}}}{\lambda_{j\to f}(y_j)^{w_{j\to f}}} \right).
\end{aligned}$$

Next we take the logarithm of both sides to get that

$$\begin{aligned}
\log \big(\hat{R}\nu\big)_{f\to i}(x_i) &\leq \log \big(\hat{R}\lambda\big)_{f\to i}(x_i) + \log \left(\max_{y_j} \prod_{j\in\mathcal{N}(f)\setminus i} \frac{\nu_{j\to f}(y_j)^{w_{j\to f}}}{\lambda_{j\to f}(y_j)^{w_{j\to f}}} \right) \\
&= \log \big(\hat{R}\lambda\big)_{f\to i}(x_i) + \max_{y_j} \sum_{j\in\mathcal{N}(f)\setminus i} w_{j\to f} \log \frac{\nu_{j\to f}(y_j)}{\lambda_{j\to f}(y_j)} \\
&\leq \log \big(\hat{R}\lambda\big)_{f\to i}(x_i) + \sum_{j\in\mathcal{N}(f)\setminus i} w_{j\to f} \max_{y_j} \left| \log \frac{\nu_{j\to f}(y_j)}{\lambda_{j\to f}(y_j)} \right| \\
&\leq \log \big(\hat{R}\lambda\big)_{f\to i}(x_i) + \max_{j\in\mathcal{N}(f)\setminus i} \max_{y_j} \left| \log \frac{\nu_{j\to f}(y_j)}{\lambda_{j\to f}(y_j)} \right|
\end{aligned}$$

$$\implies \log \frac{\big(\hat{R}\nu\big)_{f\to i}(x_i)}{\big(\hat{R}\lambda\big)_{f\to i}(x_i)} \leq \max_{j\in\mathcal{N}(f)\setminus i} \max_{x_j} \left| \log \frac{\nu_{j\to f}(x_j)}{\lambda_{j\to f}(x_j)} \right|$$

Since this inequality holds when ν and λ are interchanged, we can take the absolute value of the left hand side. Moreover, given that the above inequality holds for any $x_i \in \Omega$, hence

$$\begin{aligned}
\max_{x_i} \left| \log \frac{\big(\hat{R}\nu\big)_{f\to i}(x_i)}{\big(\hat{R}\lambda\big)_{f\to i}(x_i)} \right| &\leq \max_{j\in\mathcal{N}(f)\setminus i} \max_{y_j} \left| \log \frac{\nu_{j\to f}(y_j)}{\lambda_{j\to f}(y_j)} \right| \\
&= \max_{i\in V} \max_{f\in\mathcal{N}(i)} \max_{y_i} \left| \log \frac{\nu_{i\to f}(y_i)}{\lambda_{i\to f}(y_i)} \right|.
\end{aligned}$$

Since the edge $\{i, f\} \in E$ was chosen arbitrarily from the beginning, the inequality holds for any $\{i, f\} \in E$. Thus, the final result is obtained by taking the maximum of the left hand side over all the edges in the graph.

$$\max_{i\in V}\max_{f\in\mathcal{N}(i)}\max_{x_i}\left|\log\frac{\big(\hat{R}\nu\big)_{f\to i}(x_i)}{\big(\hat{R}\lambda\big)_{f\to i}(x_i)}\right|\leq\max_{i\in V}\max_{f\in\mathcal{N}(i)}\max_{x_i}\left|\log\frac{\nu_{i\to f}(x_i)}{\lambda_{i\to f}(x_i)}\right|$$

$$\implies d\big(\hat{R}\nu,\hat{R}\lambda\big)\leq d(\nu,\lambda)$$

□

Proposition 5.2.13. *The operator $\hat{\Theta}$ is a contraction with respect to the metric d.*

Proof. Given any $\mu,\lambda\in\mathcal{K}$, then

$$\begin{aligned}d\big(\hat{\Theta}\mu,\hat{\Theta}\lambda\big)&=d\big(\hat{S}\hat{T}\mu,\hat{S}\hat{T}\lambda\big)\\&=d\big(N\hat{R}\hat{T}\mu,N\hat{R}\hat{T}\lambda\big)\\&\leq 2\,d\big(\hat{R}\hat{T}\mu,\hat{R}\hat{T}\lambda\big)\\&\leq 2\,d\big(\hat{T}\mu,\hat{T}\lambda\big)\\&\leq 2\gamma\,d(\mu,\lambda)\\&< d(\mu,\lambda)\end{aligned}$$

where lines 2 - 4 hold by using Lemmas 5.2.11 - 5.2.10. □

Theorem 5.2.1. *The message passing operator $\hat{\Theta}$ has a unique fixed point $\mu^\star\in\mathcal{K}$ and the sequence of messages defined by $\mu^{n+1}:=\hat{\Theta}\mu^n$ converges to $\mu^\star$ for any initialization $\mu^\star\in\mathcal{K}$. Furthermore, after n iterations*

$$d\big(\hat{\Theta}^{(n)}\mu^{(0)},\mu^\star\big)\leq(2\gamma)^n d(\mu^{(0)},\mu^\star).$$

Proof. Given that $\hat{\Theta}$ is a contraction by Proposition 5.2.13 and defined over a complete metric space by Proposition 5.2.5, then the result holds by applying Banach's Fixed Point Theorem. □

5.3 Max-Product Algorithm

The main objective of this section is to present the factor graph version of the max-product algorithm in convex combination belief propagation. In Section 5.3.1, we define the message passing operators and the message update scheme. Then we prove that the algorithm is guaranteed to converge to a unique fixed point in Section 5.3.2.

5.3.1 Message Passing Operators

Now we define a max-product version of convex combination belief propagation and show that the resulting operator has a unique and globally attractive fixed point. We prove this result by showing that this operator is contractive, then invoke Banach's fixed point theorem. This operator is analogous to the one derived from the sum-product equations in the sense that the incoming messages are "weighted" so that each node receives a convex combination of messages from its neighbors.

Definition 5.3.1. *The message passing operator $\hat{M} : \mathcal{M} \to \mathcal{K}$ in max-product belief propagation is $\hat{M} = N\hat{P}$. N normalizes each message and $\hat{P}$ computes a new message using a weighted version of the max-product equation with*

$$\left(\hat{P}\nu\right)_{f\to i}(x_i) = \max_{x_{\mathcal{N}(f)}} \left\{ \Psi_f(x_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)\setminus i} \nu_{j\to f}(x_j)^{w_{j\to f}} \right\}$$

where the weights in the sum must satisfy $\sum_j w_{j\to f} \leq 1$.

The operator $\hat{M}$ is part of a message passing scheme that involves sending messages from factors to nodes and nodes to factors. In Chapter 3, we described that the convention is to compute two sets of messages on each iteration, but this scheme is inefficient in terms of the space complexity. Instead we argued that it is better to use a composition operator that uses half as much memory. In the case of the max-product algorithm, we can utilize the same message passing scheme that consists of a composition of operators that involves $\hat{M}$.

Definition 5.3.2. *The message passing operator $\hat{Q} : \mathcal{K} \to \mathcal{K}$ in max-product convex combination belief propagation is $\hat{Q} = N\hat{M}\hat{T}$ with*

$$\left(\hat{M}\hat{T}\mu\right)_{f\to i}(x_i) = \max_{x_{\mathcal{N}(f)}} \left\{ \Psi_f(x_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)\setminus i} \Big(\prod_{g\in N(j)\setminus f} \mu_{g\to j}(x_j)^{\gamma w_{g\to j}} \Big)^{w_{j\to f}} \right\}.$$

5.3.2 Convergence

Next we prove that the algorithm is guaranteed to converge on graphs with arbitrary topology. The argument is analogous to the one provided in Theorem 5.2.1 and most of the supporting lemmas proven have already been proven in Section 5.2. In fact, all that remains is to prove that the message passing operator $\hat{P}$ is non-expansive.

Lemma 5.3.3. *The operator $\hat{P}$ is non-expansive with respect to the metric d.*

Proof. Choose any $\mu, \lambda \in \mathcal{K}$, then

$$\begin{aligned}
\left(\hat{P}\nu\right)_{f\to i}(x_i) &= \max_{x_{\mathcal{N}(f)}} \Big\{ \Psi_f(x_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)\setminus i} \nu_{w_{j\to f}}(x_j)^{w_{j\to f}} \Big\} \\
&= \max_{x_{\mathcal{N}(f)}} \Big\{ \Psi_f(x_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)\setminus i} \nu_{j\to f}(x_j)^{w_{j\to f}} \frac{\lambda_{j\to f}(x_j)^{w_{j\to f}}}{\lambda_{j\to f}(x_j)^{w_{j\to f}}} \Big\} \\
&\le \max_{x_{\mathcal{N}(f)}} \Big\{ \Psi_f(x_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)\setminus i} \lambda_{j\to f}(x_j)^{w_{j\to f}} \Big\} \max_{x_j} \frac{\nu_{j\to f}(x_j)^{w_{j\to f}}}{\lambda_{j\to f}(x_j)^{w_{j\to f}}} \\
&= \left(\hat{P}\lambda\right)_{f\to i}(x_i) \max_{x_j} \frac{\nu_{j\to f}(x_j)^{w_{j\to f}}}{\lambda_{j\to f}(x_j)^{w_{j\to f}}}.
\end{aligned}$$

Next we take the logarithm of both sides to get that

$$\begin{aligned}
\log\left(\hat{P}\nu\right)_{f\to i}(x_i) &\le \log\left(\hat{P}\lambda\right)_{f\to i}(x_i) + \log\left(\max_{x_j} \frac{\nu_{j\to f}(x_j)^{w_{j\to f}}}{\lambda_{j\to f}(x_j)^{w_{j\to f}}} \right) \\
&= \log\left(\hat{P}\lambda\right)_{f\to i}(x_i) + \max_{x_j} \log\left(\frac{\nu_{j\to f}(x_j)^{w_{j\to f}}}{\lambda_{j\to f}(x_j)^{w_{j\to f}}} \right) \\
&\le \log\left(\hat{P}\lambda\right)_{f\to i}(x_i) + \max_{x_j} \left| \log \frac{\nu_{j\to f}(x_j)^{w_{j\to f}}}{\lambda_{j\to f}(x_j)^{w_{j\to f}}} \right|
\end{aligned}$$

$$\implies \log\left(\hat{P}\nu\right)_{f\to i}(x_i) - \log\left(\hat{P}\lambda\right)_{f\to i}(x_i) \le \max_{x_j} \left| \log \frac{\nu_{j\to f}(x_j)^{w_{j\to f}}}{\lambda_{j\to f}(x_j)^{w_{j\to f}}} \right|.$$

Since this inequality holds when ν and λ are interchanged, we can take the absolute value of the left hand side. Moreover, given that the above inequality holds for any $x_i \in \Omega$, then

$$\begin{aligned}\max_{x_i}\left|\log\frac{\big(\hat{P}\nu\big)_{f\to i}(x_i)}{\big(\hat{P}\lambda\big)_{f\to i}(x_i)}\right| &\le \max_{x_j}\left|\log\frac{\nu_{j\to f}(x_j)^{w_{j\to f}}}{\lambda_{j\to f}(x_j)^{w_{j\to f}}}\right| \\ &\le \max_{i\in V}\max_{j\in\mathcal{N}(i)}\max_{x_j}\left|\log\frac{\nu_{j\to f}(x_j)^{w_{j\to f}}}{\lambda_{j\to f}(x_j)^{w_{j\to f}}}\right|.\end{aligned}$$

Since the edge $\{i, f\} \in E$ was chosen arbitrarily from the beginning, this inequality holds for any $\{i, f\} \in E$. The final result is obtained by taking the maximum of the left hand side over all the edges in the graph.

$$\max_{i\in V}\max_{j\in\mathcal{N}(i)}\max_{x_i}\left|\log\frac{\big(\hat{P}\nu\big)_{f\to i}(x_i)}{\big(\hat{P}\lambda\big)_{f\to i}(x_i)}\right| \le \max_{i\in V}\max_{j\in\mathcal{N}(i)}\max_{x_j}\left|\log\frac{\nu_{j\to f}(x_j)^{w_{j\to f}}}{\lambda_{j\to f}(x_j)^{w_{j\to f}}}\right|$$

$$\implies d(\hat{P}\nu, \hat{P}\lambda) \le d(\nu, \lambda).$$

□

Proposition 5.3.4. *The operator $\hat{Q}$ is a contraction with respect to the metric d.*

Proof. Given any $\mu, \lambda \in \mathcal{K}$, then

$$\begin{aligned}d\big(\hat{Q}\mu, \hat{Q}\lambda\big) &= d\big(N\hat{P}\hat{T}\mu, N\hat{P}\hat{T}\lambda\big) \\ &\le 2\, d\big(\hat{P}\hat{T}\mu, \hat{P}\hat{T}\lambda\big) \\ &\le 2\, d\big(\hat{T}\mu, \hat{T}\lambda\big) \\ &< \gamma\, d(\mu, \lambda).\end{aligned}$$

□

Theorem 5.3.1. *The operator $\hat{Q}$ has a unique fixed point $\mu^\star \in \mathcal{K}$ and the sequence of messages defined by $\mu^{(n+1)} := \hat{Q}\mu^{(n)}$ converges to $\mu^\star$ for any initialization $\mu^{(0)} \in \mathcal{K}$. Furthermore, after n iterations*

$$d\big(\hat{Q}^{(n)}\mu^{(0)}, \mu^\star\big) \le (2\gamma)^n d(\mu^{(0)}, \mu^\star).$$

Proof. Given that $\hat{Q}$ is a contraction by Proposition 5.3.4 and defined over a complete metric space by Proposition 5.2.5, then the result holds by applying Banach's Fixed Point Theorem. $\square$

5.4 Shortcomings of Convex Combination Belief Propagation

The main advantage of convex combination belief propagation is that it's guaranteed to converge to a unique fixed point on graphs with arbitrary topology. Although this algorithm has good theoretical properties, one drawback is that the resulting beliefs may not be a good approximation of the true marginals. This behavior should be anticipated because modifying the traditional message passing operators by adding weights and a damping factor affects its fixed point. To illustrate this issue with a concrete example, we use both belief propagation algorithms to obtain a fixed point in the spin glass model from statistical physics.

Example 5.4.1. *We applied convex combination and damped belief propagation to the graphical model in Example 4.2.4 and show the results in 5.1. Convex combination belief propagation converged after very few iterations, whereas damped belief propagation oscillated. The algorithm was stopped after 1000 iterations and we show how the beliefs oscillate in the last 20 iterations in Figure 5.1.*

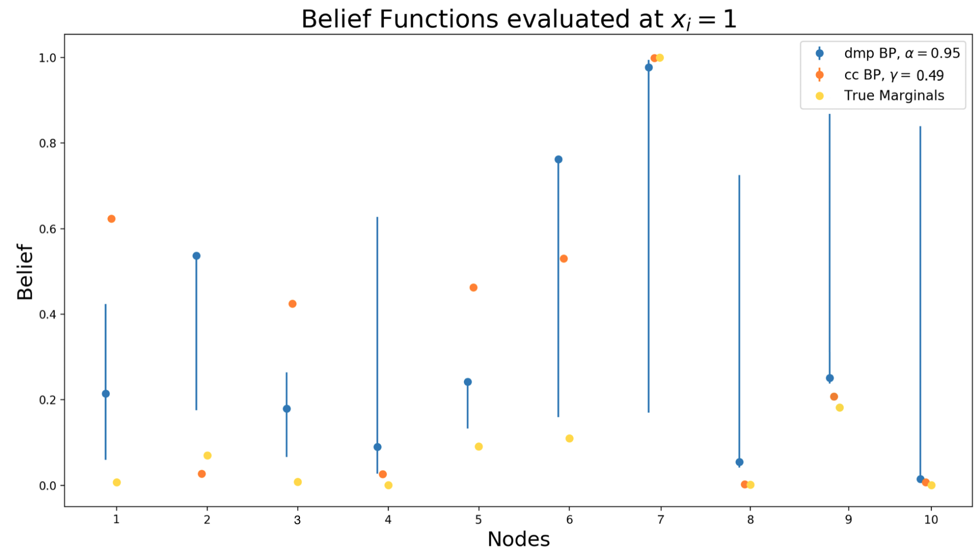


FIGURE 5.1: Beliefs and exact marginals from Example 5.4.1. Damped belief propagation failed to converge, the vertical lines show how the beliefs oscillate in the last 20 iterations.

In Example 5.4.1, damped belief propagation failed to converge and the oscillations are quite dramatic despite the damping factor being large. Although convex combination belief propagation converged in very few iterations, the beliefs are a poor approximation of the true marginals. There are many examples where convex combination belief propagation obtains good approximations, but there is no theoretical bound on the quality of this approximation.

Another drawback of convex combination belief propagation is that the quality of the approximation often depends upon the topology of the graph. As the size of the neighborhood of a node increases, the magnitude of each weight placed on incoming messages gets closer to zero. In this case, the beliefs are more likely to be a poor approximation of the true marginals. To illustrate this issue with another concrete example, we use both belief propagation algorithms to compute the exact marginals of joint distributions defined with respect to star factor graphs as shown in Figure 5.2.

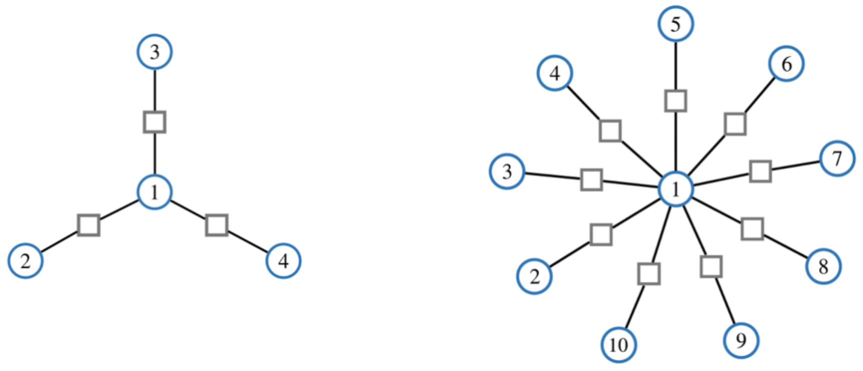


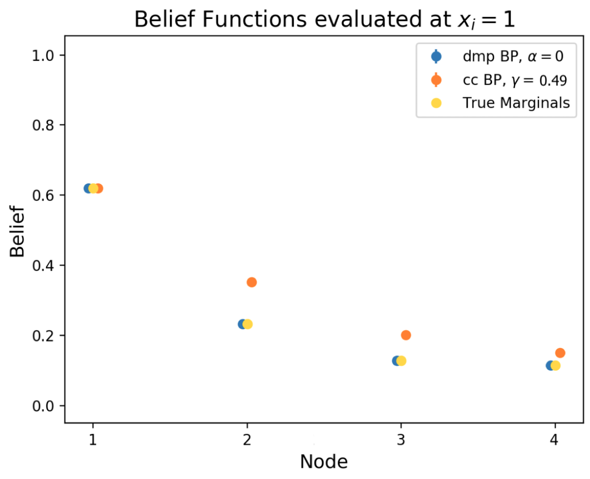
FIGURE 5.2: The star graphs S_4 and S_{10} are shown on the left and right, respectively. Blue circles represent variables and gray squares represent factors.

Example 5.4.2. *Let S_4 and S_{10} be the star graphs shown in Figure 5.2. Let $X = (X_1, \ldots, X_n)$ be a random vector with either $n = 4$ or $n = 10$ and let $\Omega = \{-1, 1\}$. The probability of each configuration of the random vector is given by the joint distribution*

$$\mathbb{P}(X = x) = \frac{1}{Z} \exp\Bigg(- \sum_i y_i x_i - \sum_{i,j} \lambda_{ij} x_i x_j \Bigg).$$

In this experiment, the local influence at each node is $y_i = 1$ and λ_{ij} was sampled from the normal distribution $\mathcal{N}(1, 0.5)$.

Let $w_{f\to i} = 1/\big(|\mathcal{N}(i)|-1\big)$ *and* $w_{i\to f} = 1/\big(|\mathcal{N}(f)|-1\big)$ *along with* $\gamma = 0.49$ *in convex combination belief propagation. Let* $\alpha = 0$ *in damped belief propagation since the graph is tree-structured and belief propagation is guaranteed to converge. We used the initial set of messages* $\mu^{(0)}_{f\to i} = (1,1)$*. For each belief propagation algorithm, we computed the beliefs along with the true marginals of the joint distributions. In Figure 5.3, we show the value of these functions when* $x_i = 1$*.*



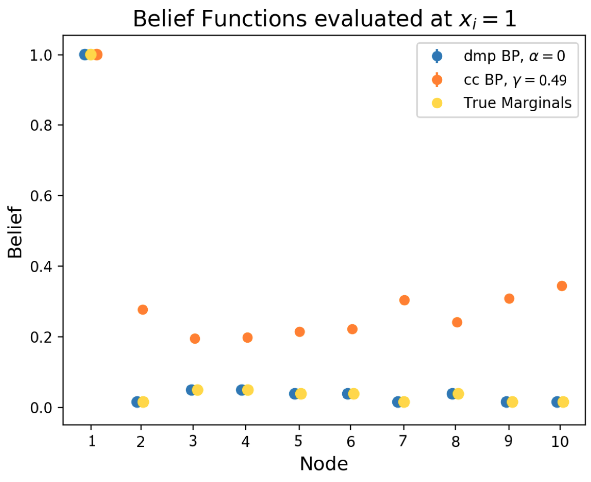


FIGURE 5.3: Beliefs obtains with belief propagation algorithms and exact marginals from Example 5.4.2. The beliefs shown on the left correspond to the star shaped graph S_4, while the beliefs on the right correspond to S_{10}.

In Figure 5.3, we see that damped belief propagation computes the exact marginal distributions since the underlying factor graphs are tree-structured. In contrast, the beliefs obtained with convex combination belief propagation differ from the true marginals in both cases. The beliefs are significantly less accurate in the case of the factor graph S_{10} due to the neighborhood of the center node being large. Although the weights in convex combination belief propagation are the key to the algorithm always converging, they are also the culprit behind the beliefs being inaccurate. This problem inspired the development of an algorithm that utilizes the message passing operator from convex combination belief propagation in a numerical homotopy continuation algorithm.

CHAPTER 6

Belief Propagation with Numerical Homotopy Continuation

This chapter presents a new belief propagation algorithm that utilizes both convex combination belief propagation and numerical homotopy continuation in a unified framework. In Section 6.1, we discuss our motivations and describe related work in the literature. We present the sum-product version of the numerical homotopy continuation algorithm in Section 6.2, then prove several theoretical results in Section 6.3. In Section 6.4, we present a series of numerical experiments, then discuss techniques that improve the performance in Section 6.5. Section 6.6 presents the max-product version of the continuation algorithm. Lastly, we provide some discussion of the algorithm and future work in Section 6.7

6.1 Introduction

Belief propagation is well-known for obtaining state of the art results in certain applications. However, the algorithm is also notorious for being unreliable when the underlying graph has complex topology. This is problematic because many real world networks fall into this category. Although there are alternative methods that are capable of performing inference in these settings, belief propagation is very efficient when it converges. This problem is the main motivation behind the developments in this thesis as well as many other works over the last few decades.

Convex combination belief propagation is our initial approach to this problem. The algorithm has good theoretical properties, but the accuracy of the approximation is unreliable. The weights in the message passing operator are the key to the algorithm converging, but also the culprit behind a poor approximation. We improve upon this algorithm by systematically phasing out the weights with homotopy continuation. This process is carried out with a homotopy operator that defines a continuous mapping between the message passing operators in convex combination and damped belief propagation. The result is a homotopy continuation algorithm which is significantly more accurate than convex combination belief propagation. In addition, this method converges at a much higher rate than damped belief propagation.

Homotopy continuation is a well-established numerical method that to solves high dimensional systems of nonlinear equations. Instead of directly solving a target system of equations, the method obtains a solution by first solving a simple system of equations. Then a homotopy operator gradually deforms the simple system into the target system. As the system deforms, the solutions of the simple system also gradually deform into solutions of the target system. The computational objective is to track a path of solutions that originates at a solution of the simple system and terminates at a solution of the target system.

There are a few related works which have applied similar techniques to this problem. Knoll et al. (2017) present the numerical polynomial-homotopy-continuation which is capable of obtaining all fixed points of belief propagation [50]. Their approach exploits that computing belief propagation fixed points can be realized as solving a polynomial system of equations. In this case, more advanced polynomial homotopy constructions can be applied. They use the polyhedral homotopy method by Huber and Sturmfels (1995), which is especially well-suited for finding all isolated nonzero complex solution [42]. This method obtains all belief propagation fixed points, but its very computationally expensive and only practical for simple examples.

Knoll et al. (2018) introduced a homotopy continuation based approach called *self-guided* belief propagation [51]. Their method interpolates between a pairwise model and a simplification of that model with only unary potentials. One key advantage of our algorithm over self-guided belief propagation is that it's applicable to the most general graphical models and incorporates edge information in the initial system.

6.2 Homotopy Continuation Algorithm

Next we present the sum-product version of a homotopy continuation algorithm that obtains a good approximation. In Section 6.2.1, we provide a concise overview of homotopy continuation methods. Section 6.2.2 presents the message passing scheme and a simple example where we compute the path of fixed points. Lastly, we describe the implementation and present pseudo code in Section 6.2.3.

6.2.1 Basics of Homotopy Continuation

Homotopy continuation is a powerful computational tool that can be used to solve high dimensional systems of nonlinear equations. Consider the system of equations given by

$$Fx = \vec{0}, \tag{6.1}$$

where $F : \mathbb{R}^n \to \mathbb{R}^m$ is a nonlinear operator. Although there are many computational methods for solving systems of nonlinear equations (e.g. Newton's method, conjugate-gradient methods), continuation is often used when the system is difficult to solve and conventional methods fail.

The main idea behind this approach is to obtain a solution to the system by defining a homotopy (or deformation) between the original system and another system $Gx = \vec{0}$ with $G : \mathbb{R}^n \to \mathbb{R}^m$ which can be easily solved. A standard homotopy $H : \mathbb{R}^n \times [0, 1] \to \mathbb{R}^m$ used in this method is

$$H(x, t) = (1 - t)Gx + tFx,$$

where $H(x, 0) = Gx$ and $H(x, 1) = Fx$. Instead of directly solving the target system of equations in (6.1), homotopy continuation indirectly solves this system by repeatedly solving

$$H(x, t) = \vec{0} \tag{6.2}$$

as t is gradually varied from 0 to 1. Most continuation algorithms rely on an iterative scheme (e.g. Newton's method) to solve this system on each time step. In this case, the solution from the previous time step is used as the initialization in the next time step.

As the time parameter t varies from 0 to 1, the simple system continuously deforms into the target system in Equation 6.1. In conjunction, solutions of the homotopy system also continuously deform in the process. This creates a path of solutions where each point belongs to the set

$$H^{-1}\big(\vec{0}\big) = \big\{(x,t) : H(x,t) = \vec{0}\big\}.$$

This path originates at a solution of the trivial system, then emanates from this point and often bifurcates into distinct branches. Each branch eventually terminates at a solution of the target system as $t \to 1$. Continuation is often referred as a path-tracing method because this algorithm traces one of these branches.

6.2.2 Message Passing with a Homotopy

Homotopy continuation is a natural fit for our problem because we have the following systems of equations

$$\hat{S}\hat{T}\mu = \mu \quad \text{and} \quad ST\mu = \mu. \tag{6.3}$$

The system on the left is trivial to solve with convex combination belief propagation. In contrast, the system on the right can be very difficult to solve, especially when the topology of the underlying graph is complex. Our solution is to define a homotopy between the two systems, then trace a path of fixed points that originates at the unique fixed point of convex combination belief propagation and terminates at a fixed point of loopy belief propagation.

Definition 6.2.1. *The message passing operator $\Theta : \mathcal{M} \to \mathcal{M}$ in the sum-product algorithm with damping is given by*

$$\Theta\mu = (1-\alpha)(NST)\mu + \alpha\mu$$

with $\alpha \in [0,1]$.

Definition 6.2.2. *The message passing operator $H : \mathcal{M} \times [0,1] \to \mathcal{M}$ in sum-product algorithm with homotopy continuation is*

$$H(\mu,t) = (1-t)\hat{\Theta}\mu + t\Theta\mu.$$

In order for numerical continuation to be feasible, the fixed points must form a continuous path as t is varied from 0 to 1. Later in this chapter, we prove that the homotopy operator in Definition 6.2.2 satisfies this property. In the meantime, we provide a concrete example of analytically computing a path of fixed points for a simple graphical model.

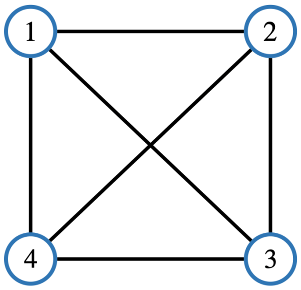


FIGURE 6.1: Complete graph with 4 nodes.

Example 6.2.3. *Let $G = (V, E)$ be a complete graph with four nodes as shown in Figure 6.1. Let $X = (X_1, \ldots, X_4)$ be a random vector and let $\Omega = \{-1, 1\}$ be a set of possible outcomes of each random variable. The joint distribution of this system is*

$$\mathbb{P}(X = x) = \frac{1}{Z} \exp\bigg(- \sum_{\{i,j\} \in E} x_i x_j \bigg).$$

This system is simple enough to analytically compute the fixed point set of the homotopy operator. This involves using a symbolic algebra system to solve the fixed point equation

$$\mu = (1-t)\hat{\Theta}\mu + t\Theta\mu.$$

Although this is a high dimensional system of nonlinear equations, we can compute a subset of fixed points by exploiting underlying symmetry in the system. In this approach, solving the fixed point equation corresponding to a single pair of neighboring nodes is equivalent to solving the entire system! In Figure 6.2, we show all of the solutions to the fixed point equation corresponding to an arbitrary node and factor for all $t \in [0, 1]$.

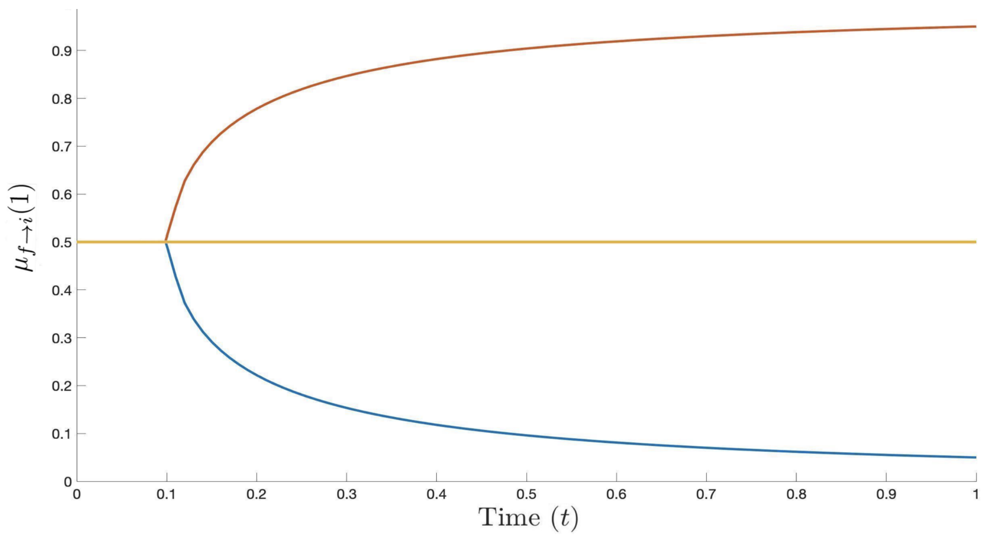


FIGURE 6.2: Solutions of the fixed point equation for every $t \in [0, 1]$. There is a unique fixed point of the homotopy operator when $t \in [0, 0.1)$. Then the path bifurcates into three distinct branches when $t = 0.1$.

In practice, our objective is solve the fixed point equation

$$\mu = H(\mu, t).$$

for some ascending sequence of time steps $(t_0, t_1, \ldots, t_N) \subset [0, 1]$. The initial time step is $t_0 = 0$ since this system has a unique solution $\mu_0^\star \in \mathcal{M}$ that can be easily obtained with convex combination belief propagation. The main computational objective of a continuation algorithm is to approximate a solution at each time step, where $\mu_i^\star \in \mathcal{M}$ denotes a solution of the fixed point equation at time t_i.

There are many numerical schemes that can be used to solve this system of equations. In the context of homotopy continuation, Newton's method is most commonly used. Although this method is well-understood and relatively simple to implement, one disadvantage is that it involves inverting a matrix. This poses a major computational challenge because we are interested in tackling the most difficult inference problems. In particular, our algorithm must be robust to high dimensional systems and graphs with dense neighborhoods.

Instead we propose using fixed point iteration with $H(\cdot, t)$ due to its relatively low computational overhead. This involves simultaneously performing damped belief propagation (fixed point iteration with Θ) and convex combination belief propagation (fixed point iteration with $\hat{\Theta}$) at each time step, then updating the current set of message via

$$\mu_t^{(n)} := H(\mu_t^{(n-1)}, t) = (1-t)\hat{\Theta}\mu_t^{(n-1)} + t\Theta\mu_t^{(n-1)}$$

with $t \in [0,1]$. Once the message $\mu_t^{(n)}$ converges to a fixed point, the time parameter t is updated by $t \leftarrow t + dt$ with $dt > 0$ being the time step. (Note that the set of messages includes the subscript t in order to clearly distinguish messages from distinct time steps.)

An underlying assumption of this algorithm (which is proven to be true) is that the fixed points of $H(\mu, t)$ form a continuous path. One implication of this assumption is that the approximate fixed point $\mu_t^{(n)}$ is near a fixed point of $H(\mu, t + dt)$ for sufficiently small $dt > 0$. Thus, we can efficiently obtain this fixed point by defining the initial set of messages in the next time step as $\mu_{t+dt}^{(0)} \leftarrow \mu_t^{(n)}$.

Example 6.2.4. *Here we use the same graphical model from Example 4.2.4. Then we applied belief propagation with homotopy continuation with* $dt = 0.1$. *Once fixed point iteration converged at each time step, we compute the belief function*

$$b_{t,1}(1) = \prod_{f \in \mathcal{N}(1)} \mu_{f\to 1}(1)$$

corresponding to node 1 at time t. The resulting belief function at each time is shown in Figure 6.3 along with the true marginal.

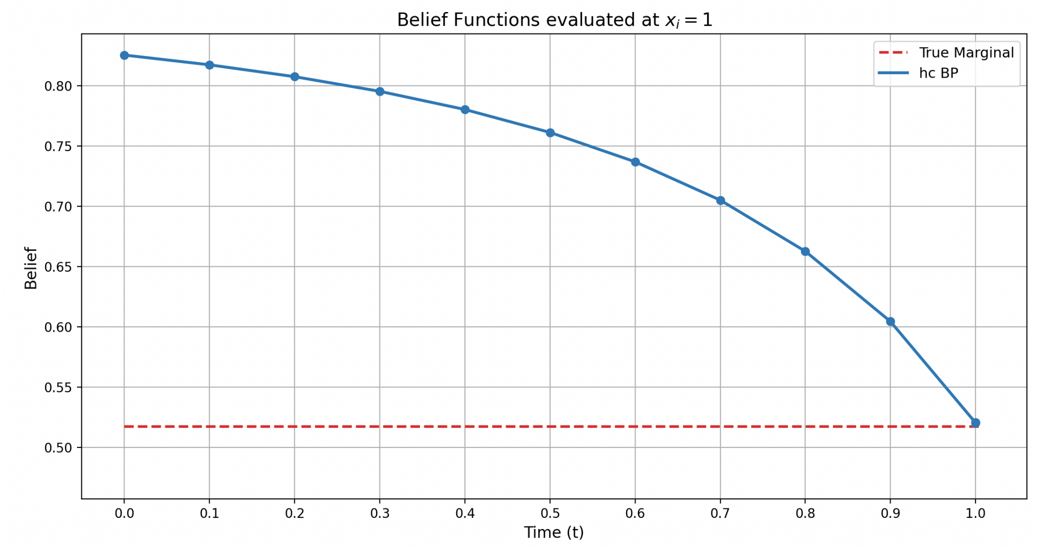


FIGURE 6.3: Single belief function evolving in time as t is varied from 0 to 1 with time steps of $dt = 0.1$.

6.2.3 Implementation

Next we present pseudo code in Algorithm 1. There are several user-defined parameters that influence the performance of the algorithm. The purpose of this section is to discuss the implementation and provide insight into choosing good parameter values.

An important aspect of any continuation algorithm is a time parameter that controls the deformation between the systems of nonlinear equations shown in Equation 6.3. In addition, there is also a step size parameter dt that controls the rate at which this deformation occurs. In general, continuation algorithms are more likely to converge at each time step when dt is small. However, this also means that more time steps are necessary to reach the final time step, which may result in longer running times.

We recommend setting $dt = 0.1$ and provide experimental results with this time step in Section 6.4. However, some very difficult problem instances may require a smaller time step in order to obtain a good approximation. This potential issue can be addressed by using a subroutine that adaptively updates the time step. If the algorithm fails to converge, then the adaptive step size routine reverts to the last convergent time step, decreases the step size, and tries again. In order to guarantee that the running time is finite, the step size must be bounded below by an additional parameter $\delta > 0$.

Algorithm 1 Belief Propagation with Homotopy Continuation

```
# Parameters
$dt$: time step
$\delta$: lower bound on time step
$\epsilon$: stopping threshold
$max\_iter$: maximum number of iterations
$adaptive\_step$: indication of whether to use an adaptive time stepping scheme
def HC_BP($dt$, $\delta$, $\epsilon$, $max\_iter$, $adaptive\_step$)
    $\mu_0^{(0)} \leftarrow$ initial set of messages
    $t \leftarrow -dt$
    while $t < 1$:
        # Update time and initialize msgs
        $t \leftarrow \min(t + dt, 1)$
        $\mu_t^{(0)} \leftarrow$EXTRAPOLATE_MSGS($\{\mu_t\}$, $\{t\}$) if $t > 0$ else $\vec{1}$

        # Fixed point iteration
        for $n$ in range($max\_iter$):
            $\mu_t^{n+1} \leftarrow H(\mu_t^{(n)}, t)$
            if $dist(\mu_t^{(n+1)}, \mu_t^{(n)}) < \epsilon$:
                break

        # Check stopping criteria
        if $dist(\mu_t^{(n+1)}, \mu_t^{(n)}) \geq \epsilon$:
            $t \leftarrow t - dt$
            $dt$, $stop \leftarrow$ UPD_TIME_STEP($dt$, $\delta$, $adaptive\_step$)
            if $stop$:
                return $\mu_t$, $t$
    return $\mu_t$, $t$

def UPD_TIME_STEP($dt$, $\delta$, $adaptive\_step$)
    if $adaptive\_step$:
        $dt \leftarrow dt/2$
        $stop \leftarrow True$ if $dt < \delta$ else $False$
    else:
        $stop \leftarrow False$
    return $dt$, $stop$
```

The runtime can be optimized by using extrapolation to initialize the messages at each time step (as suggested by [51]). On the first time step when $t = 0$, the messages are initialized as $\mu^{(0)}_{f\to i} = [1, 1]$. In practice, any initialization can be used since convex combination belief propagation converges regardless of the initialization. Once the algorithm converges with $t < 1$, the resulting approximate fixed point μ_t is stored in an array $\{\mu_t\}$ along with approximate fixed points from all previous time steps. Similarly, the corresponding time steps are also stored in an array denoted by $\{t\}$. In the case when $t > 0$, the messages are initialized by passing the fixed points and times steps into a cubic spline routine. Then extrapolation is used to obtain an approximation of the next fixed point.

Next we provide a concrete example of using Algorithm 1 to approximate the marginals of the Gibbs distribution.

Example 6.2.5. *To evaluate the accuracy of belief propagation with numerical continuation, we apply the algorithm to the graphical model in Example 4.2.4.Let* $dt = 0.1$ *be the time step and let* $\mu^{(0)}_{f\to i} = [1, 1]$ *be the initial set of messages. In Figure 6.4, we show the beliefs obtained with homotopy continuation (fixed point iteration with* H*) along with the beliefs shown in Figure 4.14.*

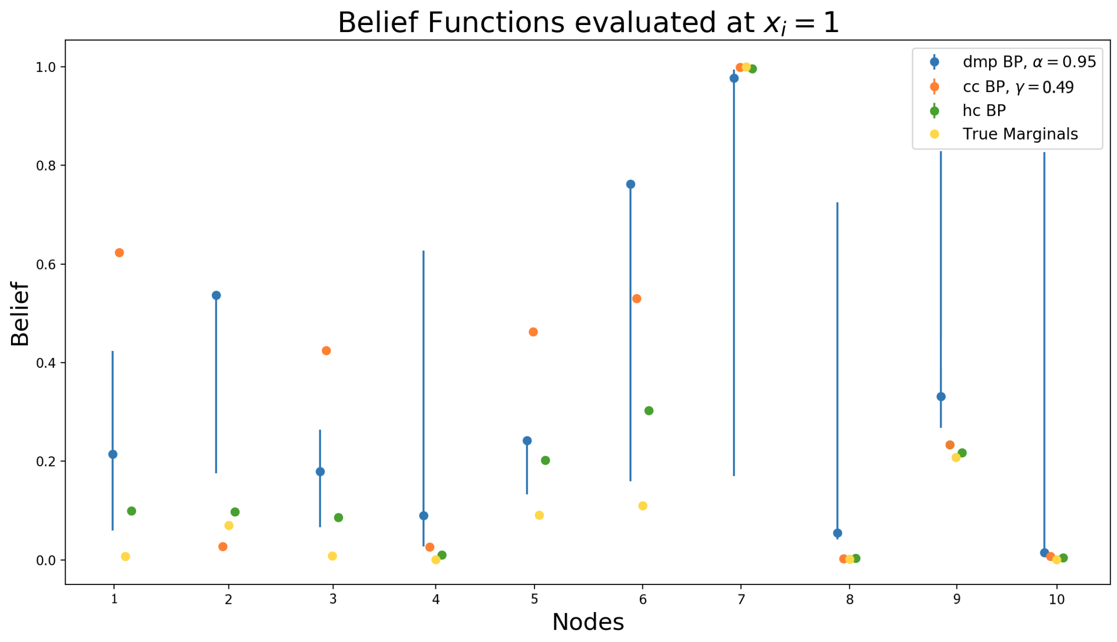


FIGURE 6.4: Beliefs obtained with belief propagation algorithms. Although convex combination belief propagation and the homotopy continuation algorithm both converged, the beliefs obtained with continuation are significantly more accurate.

6.3 Theoretical Analysis

In order for any numerical continuation algorithm to be feasible, there must be a continuous path of fixed points that can be traced as t is varied from 0 to 1. In this section, we provide a rigorous argument that this property holds for arbitrary graphical models. The structure of our argument is to first prove that the homotopy has at least one fixed point for every $t \in [0, 1]$ in Section 6.3.1, then show that the fixed points form a continuous path in Section 6.3.2.

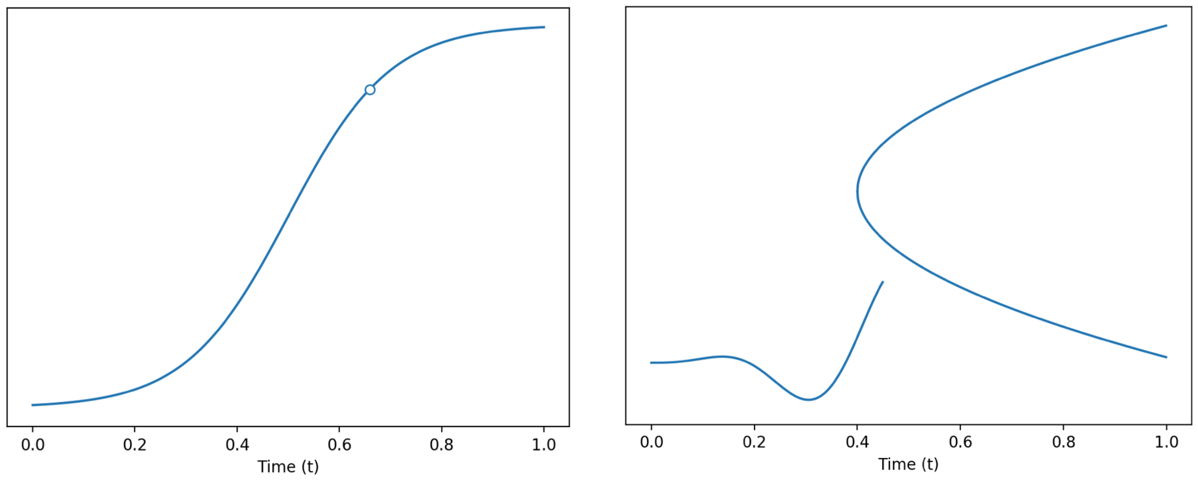


FIGURE 6.5: On the left, we see an example of a path with a hole when $t = 0.65$. A continuation algorithm would fail on this example because there is no point on the path that corresponds to this time step. On the right, we see an example of a path with a discontinuity when $t = 0.5$. A continuation algorithm is likely to fail because the starting point of the next time step is relatively far from the path.

6.3.1 Existence of Fixed Points

The existence of a solution at every time step is a necessary condition for any continuation algorithm to be feasible. Otherwise, the algorithm is doomed to fail because we may reach a time step that does not correspond to a point on the path of solutions (see Figure 6.5). In the next theorem, we prove that the homotopy operator H satisfies this necessary property by using Brouwer's fixed point theorem.

Observation 6.3.1. *Let Δ be the 1-simplex and $N = \dim \mathcal{M}$, then Δ^N is a closed and convex set.*

Theorem 6.3.1. *The homotopy operator H has at least one fixed point for every $t \in [0,1]$.*

Proof. Given any $\mu \in \Delta^N$, then $H(\mu,t) \in \Delta^N$ by

$$\begin{aligned}\sum_{x_i} H(\mu,t)_{f\to i}(x_i) &= \sum_{x_i}\Big((1-t)(\hat{\Theta}\mu)_{f\to i}(x_i) + t(\Theta\mu)_{f\to i}(x_i)\Big)\\ &= (1-t)\sum_{x_i}(\hat{\Theta}\mu)_{f\to i}(x_i) + t\sum_{x_i}(\Theta\mu)_{f\to i}(x_i)\\ &= (1-t)+t\\ &= 1.\end{aligned}$$

Since H maps the closed and convex set Δ^N into itself for any $t \in [0,1]$, H must have at least one fixed point for every $t \in [0,1]$ by Brouwer's fixed point theorem. □

6.3.2 Continuity of Fixed Point

Next we show that the fixed point set of the homotopy operator forms a continuous path. We prove this theorem by defining a set-valued function g that parametrizes the fixed point set. Given any $t \in [0,1]$, this function returns the solution set of the fixed point equation $H(\mu,t) = \mu$. Under this framework, proving that the fixed point set forms a continuous path is equivalent to showing that g is continuous.

Definition 6.3.2. *Let $g : [0,1] \to \mathcal{P}(\mathcal{M})$ be the set-valued function given by*

$$g(t) = \{\mu \in \mathcal{M} : H(\mu,t) = \mu\},$$

where $\mathcal{P}(\mathcal{M})$ denotes the power set of $\mathcal{M}$.

Definition 6.3.3. *A set-valued function $F : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$ is continuous if it is both inner and outer semicontinuous at each $x_0 \in \mathbb{R}^n$ meaning that*

$$\limsup_{x\to x_0} F(x) \subset F(x_0) \quad \textit{and} \quad F(x_0) \subset \liminf_{x\to x_0} F(x).$$

Next we establish that g is continuous by using that each fixed point is continuously dependent upon t. It is natural to use the implicit function theorem to obtain this result. However, an underlying assumption of this classical theorem is that the Jacobian of the corresponding system of equations is nonsingular. In our case, this condition would be difficult to verify for arbitrary graphical models. Instead we can circumvent this condition by using a generalization of the implicit function theorem to non-differentiable functions (see Theorem 6.3.2). In this generalization, local strict monotonicity is sufficient in one dimension. In more general systems, it has been proven that the local injectivity is necessary and sufficient [55].

Theorem 6.3.2. *Suppose that $F : D \subset \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a continuous map with*

$$F(x_0, y_0) = 0.$$

Assume there exist open neighborhoods $A \subset \mathbb{R}^n$ and $B \subset \mathbb{R}^m$ of x_0 and y_0, respectively. Then if $F(\cdot, y) : A \to \mathbb{R}^n$ is locally one-to-one for all $y \in B$, there exist open neighborhoods $A_0 \subset A$ and $B_0 \subset B$ of x_0 and y_0, respectively, such that the

$$F(x, y) = 0$$

has a unique solution $x = Gy \in A_0$ and the mapping $G : B_0 \to \mathbb{R}^n$ is continuous [55].

Theorem 6.3.3. *g is a continuous set-valued function.*

Proof. The set-valued function g being inner semi-continuous is equivalent to $g^{-1}(\mathcal{O})$ being open for every open set $\mathcal{O} \subset \mathcal{M}$. Choose any open set $\mathcal{O} \subset \mathcal{M}$ and without loss assume that $g^{-1}(\mathcal{O}) \neq \emptyset$, hence there exists at least one $\mu \in \mathcal{M}$ such that $H(\mu, t) = \mu$ for some $t \in [0, 1]$. Given any pair $(\mu_i, t_i) \subset \mathcal{O} \times [0, 1]$ satisfying the equation

$$H(\mu_i, t_i) - \mu_i = 0, \tag{6.4}$$

the mapping $F(\cdot, t_i) := H(\mu, t_i) - \mu$ is locally one-to-one and so Theorem 6.3.2 implies that there exists open neighborhoods $A_i \subset \mathcal{O}$ and B_i of μ_i and t_i, respectively, such that

$$H(\mu, t) - \mu = 0$$

has a unique solution $\mu = \tilde{g}_i(t) \in A_i$ for all $t \in B_i$ and the mapping $\tilde{g} : B \to \mathcal{M}$ is continuous. Given that $\tilde{g}_i^{-1}$ is continuous, then the inverse image $\tilde{g}_i^{-1}(A_i)$ must be open. Thus, this implies that $g^{-1}(\mathcal{O})$ must also be open because it can be written in the form

$$g^{-1}(\mathcal{O}) = \bigcup_i \tilde{g}_i^{-1}(A_i).$$

The set-valued function g being outer semi-continuous is equivalent to its graph $\mathcal{G}(g) = \{(t, \mu) : H(\mu, t) = \mu\}$ being closed [17]. Given any sequence $\big\{(t_n, \mu_n)\big\} \subset \mathcal{G}(g)$ with $(t_n, \mu_n) \to (t, \mu)$ being some limit point, it's clear that $t \in [0, 1]$ because the unit interval is closed and μ is a fixed point of H by

$$\big\|H(\mu, t) - \mu\big\| \leq \big\|H(\mu, t) - H(\mu_n, t_n)\big\| + \|\mu_n - \mu\| \to 0$$

as $n \to \infty$ because H is continuous and $\mu_n \to \mu$ by assumption. □

Corollary 6.3.4. *The homotopy operator H has a continuous path of fixed points as t is varied from 0 to 1.*

6.4 Numerical Experiments

In this section, we present two numerical experiments that compare the performance of the belief propagation algorithms. In particular, our focus is to consider difficult problem instances where damped belief propagation fails to converge. We measure the quality of the beliefs return by each algorithm by computing the mean square error (MSE) between the beliefs and true marginals. In addition, we introduce several other metrics to evaluate the overall performance of the algorithms.

6.4.1 Experimental Settings

Let $G = (V, E)$ be an undirected graph with $V = \{1, \ldots, 10\}$. Let $X = (X_1, \ldots, X_{10})$ be a random vector and let $\Omega = \{-1, 1\}$ be the set of outcomes for each random variable. The probability of a configuration of the random vector is given by

$$\mathbb{P}(X = x) = \frac{1}{Z} \exp\bigg(- \sum_{i \in V} x_i y_i - \sum_{\{i,j\} \in E} \lambda_{ij} x_i x_j \bigg).$$

Each y_i is uniformly sampled from Ω and λ_{ij} is independently sampled from a uniform distribution. The magnitude of this parameter controls how strongly neighboring states are coupled and the sign determines whether neighbors prefer to be aligned or misaligned.

For each belief propagation algorithm, we use the initialization $\mu_{ij}^{(0)} = [1, 1]$ for all $\{i, j\} \in E$ and update the messages in parallel. Let $N = 10^3$ be the maximum number of iterations. Let $\epsilon = 10^{-5}$ be a threshold that indicates when the messages have sufficiently converged. In damped belief propagation, we use the damping factor $\alpha = 0.9$ to prioritize convergence since we consider difficult problem instances. In convex combination belief propagation, we use the damping factor $\gamma = 0.49$ and set the weights uniformly. Since our continuation algorithm utilizes the message passing operators from damped and convex combination belief propagation, we use the same damping factors and weights in the homotopy operator.

We compare the performance of our algorithm to another homotopy continuation algorithm referred to as *self-guided* belief propagation (see [51]). The main idea behind this method is to use the time parameter to gradually incorporate the pairwise potentials. We also analyze the performance of these algorithms with respect to different time stepping schemes. In this section, a fixed time stepping scheme is used in both algorithms. Then we provide a side-by-side comparison of using fixed versus adaptive time stepping scheme in Section 6.5.

6.4.2 Varying the Coupling Factor

Belief propagation is known to fail when the corresponding system of equations has multiple fixed points with some being attractive while others being repulsive. In this case, the dynamics of message passing are often unstable for any damping factor. We can simulate this scenario by sampling the coupling factors λ_{ij} from a distribution centered about zero. As the magnitude of the coupling factors increases, the system becomes more unstable as the repulsive and attractive forces intensify.

We consider models where the coupling factors are sampled from a uniform distribution where $\lambda_{ij} \sim \text{Unif}(-\sigma, \sigma)$ with $\sigma \geq 0$. In this experiment, the magnitude of σ is varied from 0 to 5 with increments of $\Delta\sigma = 0.5$. For each value of σ, we generate 100 Markov random fields by sampling $\lambda_{ij} \sim \text{Unif}(-\sigma, \sigma)$. In addition, we consider models where the underlying graph is a complete graph and an Erdös-Renyi graph in which pairs of nodes are connected by an edge with probability $p = 0.5$.

In this experiment, we compare the performance of the homotopy continuation algorithms with the fixed time step $dt = 0.1$. The performance of each algorithm is determined by the accuracy of the resulting beliefs, runtime, and rate of convergence or average stopping time (depending on whether the algorithm uses continuation). The runtime is measured by the number of iterations until convergence for damped and convex combination belief propagation. In the case of the continuation algorithms, the runtime is the average number of iterations until the algorithm stops. We show the results of this experiment in terms of these performance metrics in Figures 6.6 and 6.7.

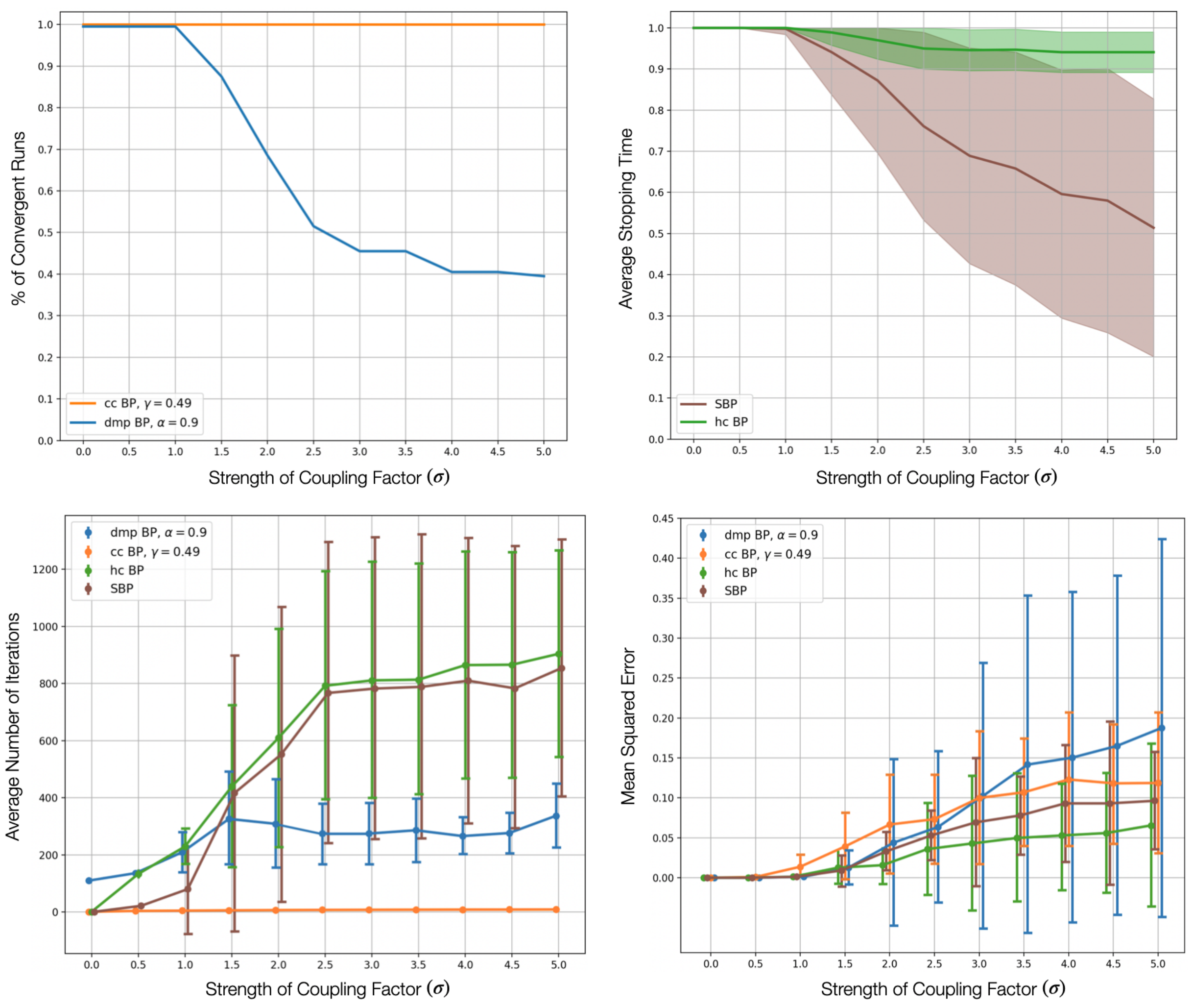


FIGURE 6.6: Erdos-Renyi Graphs with $p = 0.5$

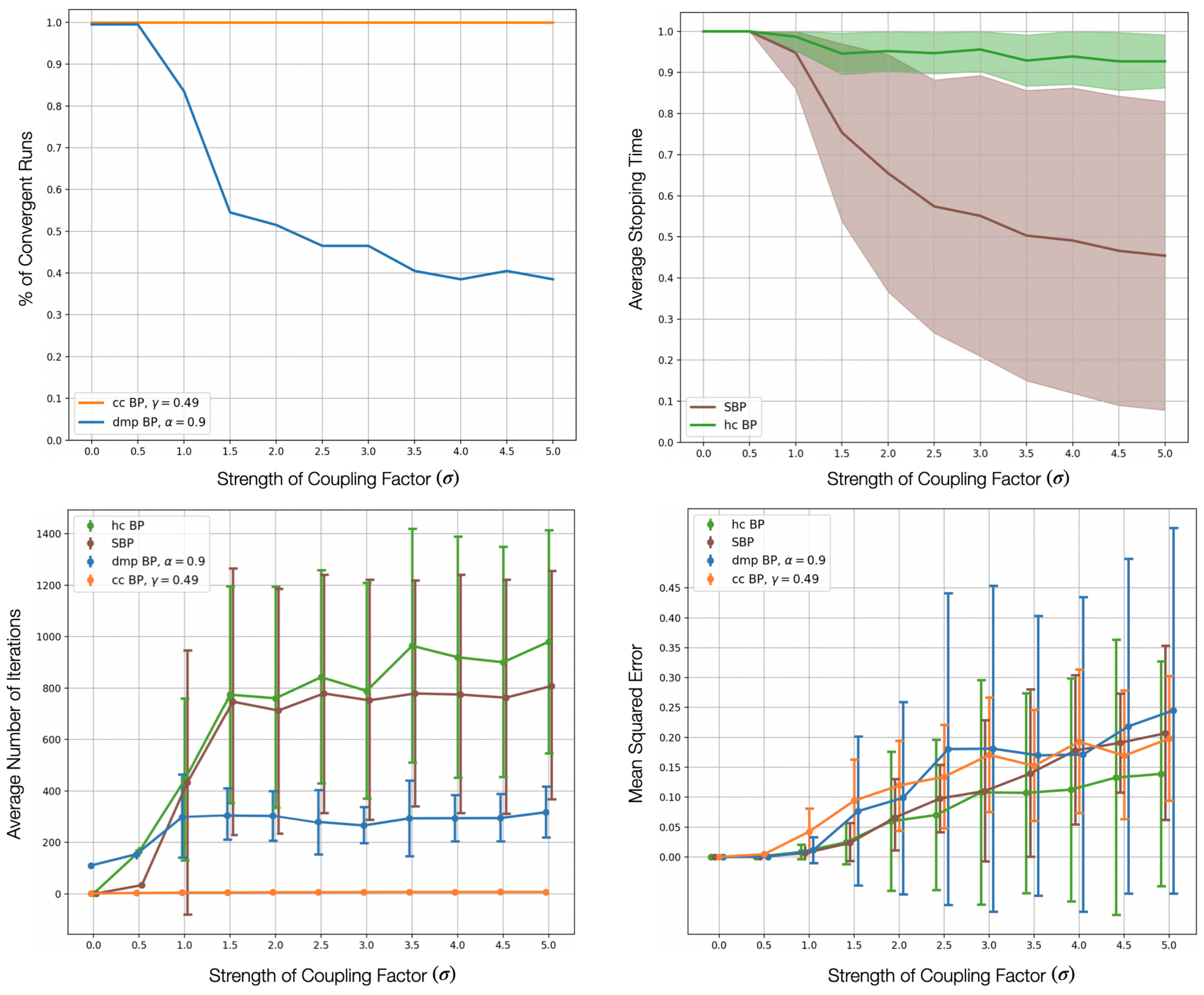


FIGURE 6.7: Complete Graphs

6.4.3 Varying the Graph Connectivity

Belief propagation is also known to either fail to converge or return a poor approximation when the topology of the graph is complex. In particular, the performance of the algorithm suffers when the graph contains cycles because this structure creates message passing feedback loops. In this experiment, we compare the performance of the belief propagation algorithms while gradually increasing the connectivity of the graph.

We consider Markov random fields where the underlying graph is an Erdös-Renyi random graph. The edge appearance probability p is varied from 0 to 1 with increments of $\Delta p = 0.1$. For each value of p, we obtain 100 problem instances by generating an Erdös-Renyi graph and sampling

$\lambda_{ij} \sim \text{Unif}(-5, 5)$ for each problem instance. Similar to the previous experiment, we use the fixed time step $dt = 0.1$ in the homotopy continuation algorithms. Lastly, the results of this experiment in terms of the previously defined performance metrics are shown in Figure 6.8.

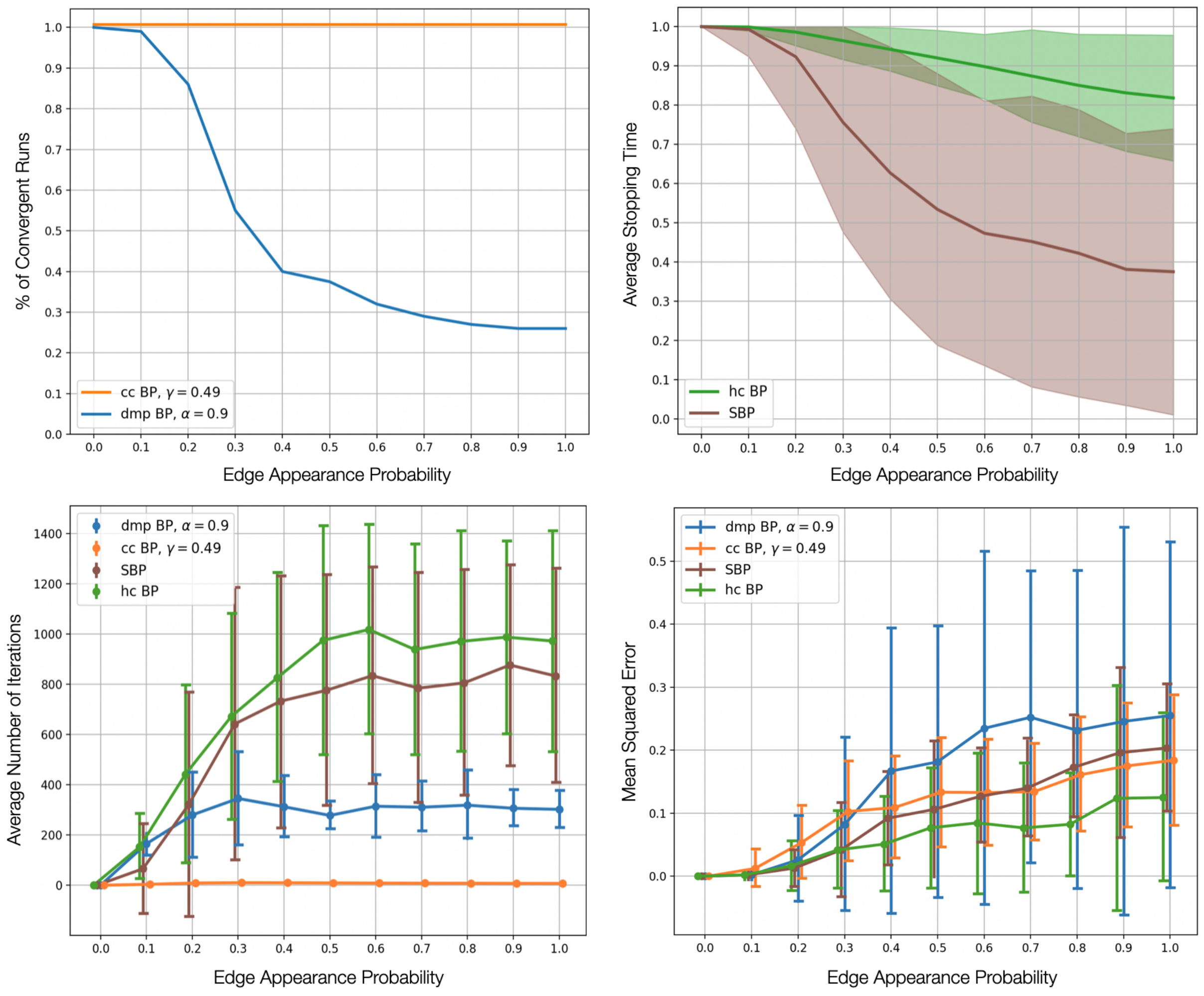


FIGURE 6.8: Erdos-Renyi Graphs with $p = 0.5$

6.5 Performance Enhancement

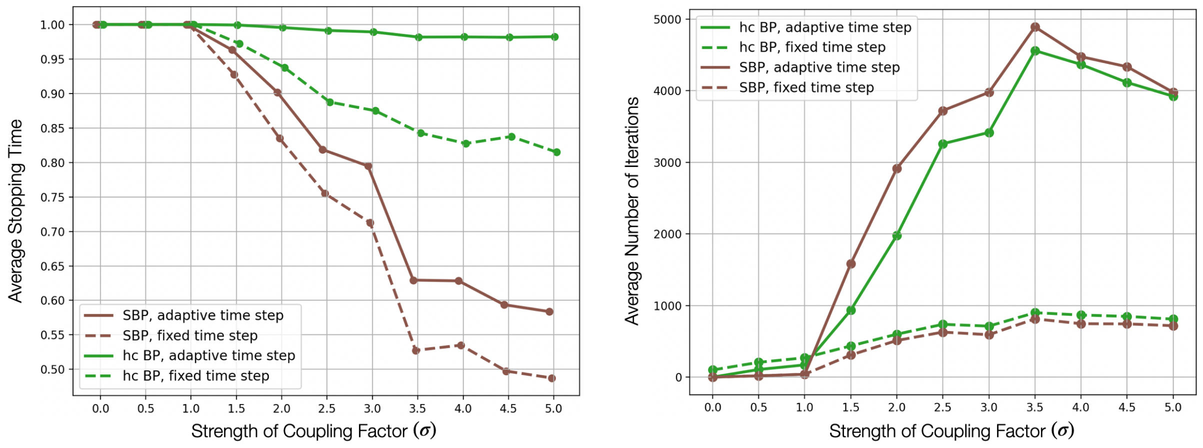
In this section, we present numerical techniques that enhance the performance of the homotopy continuation algorithm in terms of the accuracy and runtime. We describe a series of experiments that compare the performance of our algorithm with a fixed versus adaptive time stepping routine in Section 6.5.1. Then Section 6.5.2 discusses experiments where the algorithm is stopped at a time steps $t' < 1$. This practice significantly improves the runtime with a slight decrease in the accuracy of the beliefs.

6.5.1 Adaptive Time Step

Determining a good step size is a potential challenge associated with any homotopy continuation algorithm. The method is more likely to reach a time close to $t = 1$ when the step size is small. However, there is often a trade-off with the runtime because a large number of iterations may be required. Even if the algorithm converges quickly at each time step, the total number of iterations can still be very large. For example, when the step size is $dt = 10^{-n}$, the number of iterations is bounded below by 10^n.

The main purpose of an adaptive time stepping routine is to obtain good results without explicitly determining an optimal fixed step size. The results in the previous section were obtained with a good step size, which is determined by repeatedly running the algorithm with various time steps. In this experiments, we use a step size that has been predetermined to be too big, then compare the performance of the continuation algorithm with a fixed versus adaptive time stepping scheme.

Let $dt = 0.25$ be the initial time step for time stepping schemes. Let $\delta = 0.01$ be the lower bound on the time step in the adaptive time stepping routine. Now we repeat the exact same experiment described in Section 6.6. The only difference is that we compare the performance of the homotopy continuation algorithms with a fixed versus adaptive time stepping scheme. The results of this experiment in terms of the previously defined performance metrics are shown in Figure 6.9.



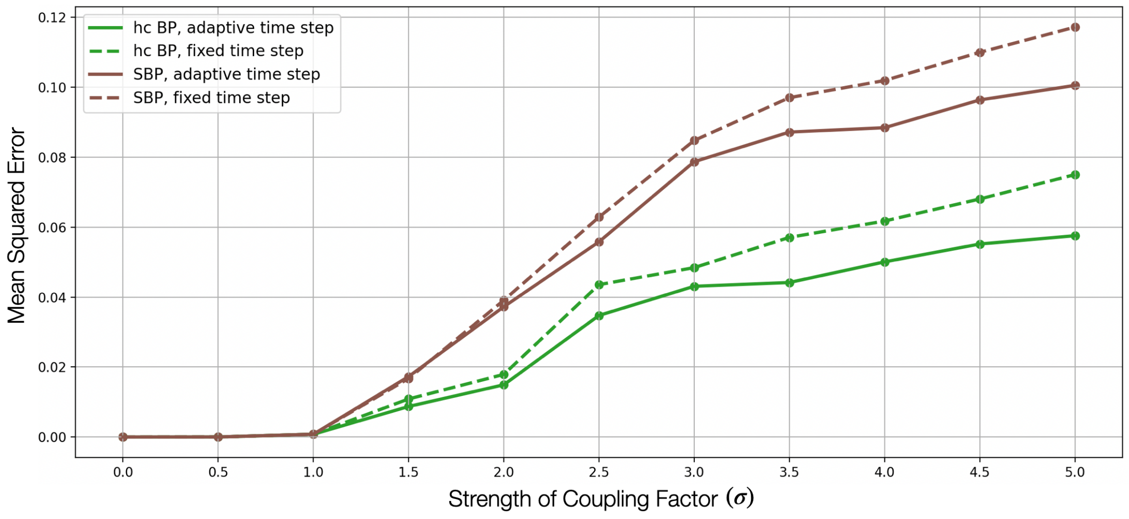


FIGURE 6.9: Erdos-Renyi Graphs with $p = 0.5$

6.5.2 Stopping Early

Although our algorithm obtains a good approximation, these experiments highlight that one disadvantage of numerical continuation methods is that they may require a large number of iterations. In this particular application, the runtime may suffer due to fixed point iteration becoming unstable when $t \approx 1$. We address this potential issue with the simple solution of stopping early which drastically reduces the runtime while still resulting in a good approximation.

In difficult problem instances where loopy belief propagation fails to converge, we have observed that the number of iterations until convergence at each time step often grows exponentially as $t \to 1$ (see Figure 6.10). Meanwhile, the approximation error between the beliefs and true marginals improves. Thus, there is often a trade-off between the approximation error and runtime when the algorithm is applied to a difficult problem instance. In order to understand the nature of this trade-off, we perform an experiment where the stopping time τ is varied from 0.5 to 1 with increments of $d\tau = 0.05$.

We generated 100 Markov random fields with each consisting of a distinct Erdös-Renyi graph with 10 nodes and potentials $\lambda_{ij} \sim \text{Unif}(-5, 5)$. Then we performed our homotopy continuation algorithm in addition to convex combination and damped belief propagation as a comparison. We measure the performance of the algorithm by tracking the total number of iterations that were required to reach each stopping time between 0.5 and 1. In addition, we also compute the mean square error and convergence rate (i.e. the percentage of runs that reach a given stopping time).

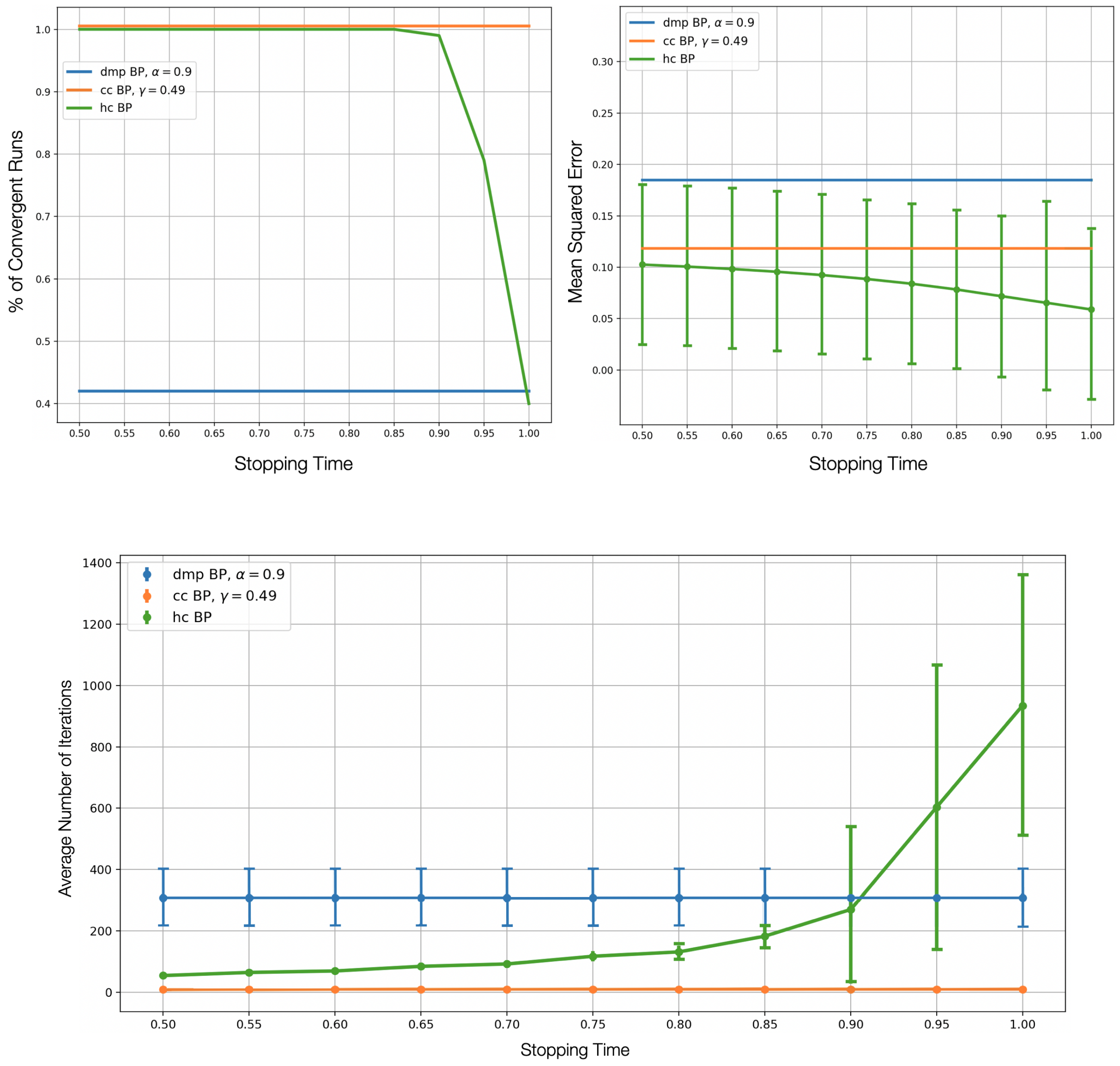


FIGURE 6.10: Erdos-Renyi Graphs with $p = 0.5$

The nature of this relationship is shown in Figure 6.10. We see that the approximation error gradually decreases, while the runtime exponentially increases. Thus, stopping early drastically improves the runtime, which only resulting in a marginally less accurate approximation

6.6 Max-Product Algorithm

In this section, we present the max-product version of the homotopy continuation algorithm. The message passing operators in this algorithm are defined analogously. In addition, the main results from the previous section hold in this version of the algorithm.

Definition 6.6.1. *The message passing operator $Q : \mathcal{M} \to \mathcal{M}$ in max-product belief propagation with damping is given by*

$$Q\mu = (1 - \alpha)P\mu + \alpha\mu$$

with $\alpha \in [0, 1]$.

Definition 6.6.2. *The message passing operator $\tilde{H} : \mathcal{M} \times [0, 1] \to \mathcal{M}$ in max-product belief propagation with homotopy continuation is*

$$\tilde{H}(\mu, t) = (1 - t)\,\hat{Q}\mu + t\,Q\mu$$

with $t \in [0, 1]$.

The continuation algorithm is performed in the exact same manner as the sum-product version. The messages are initialized with $\mu^{(0)} \in \mathcal{M}$ and the time parameter is set to $t = 0$. Then fixed point iteration is performed with the homotopy operator $\tilde{H}(\cdot, t)$. Similar to the sum-product case, an underlying assumption of this algorithm is that the fixed points form a continuous path as t is varied from 0 to 1. The same result holds in this section by using that $\tilde{H}$ is also a bounded and continuous operator.

Corollary 6.6.3. *The solution set of the fixed point equation $\tilde{H}(\mu, t) = \mu$ form a continuous path as t is varied from 0 to 1.*

Proof. This result follows by adapting the arguments provided in Theorems 6.3.1 and 6.3.3. $\square$

6.7 Conclusion

In this chapter, we present a new belief propagation algorithm that utilizes convex combination belief propagation and homotopy continuation in a unified framework. Convex combination belief propagation obtains an initial approximation, then homotopy continuation improves the accuracy of this approximation. This algorithm is supported by several theoretical results, the first being that convex combination belief propagation is guaranteed to converge to a unique fixed point on arbitrary graphical models. This result ensures that the homotopy continuation algorithm is guaranteed to converge to a unique initial solution on the first time step. In addition, there is a continuous path of fixed points that emanates from this initial fixed point and converges to a fixed point of traditional belief propagation. We have also shown that this algorithm obtains a good approximate across many difficult problem instances.

One important future direction is to understand the convergence behavior of this algorithm. This is a complex issue because it is also dependent upon the magnitude of the time step. Although the algorithm converges at a much higher rate than traditional belief propagation, there is no guarantee of convergence. In the cases when the algorithm fails to converge, it generally reaches a time step very close to 1. Thus, we believe that there may exist certain graphical models (containing loops) in which the algorithm is guaranteed to converge.

CHAPTER 7

Application: Approximate Inference in the QMR Network

This chapter discusses using belief propagation with homotopy continuation to perform inference in a medical diagnostic network called the QMR network. Section 7.1 broadly introduces the QMR network, then provides a concise overview of related works. Then Section 7.2 focuses on a technical introduction to the QMR network and the inference problem. In Section 7.3, we discuss using belief propagation to perform inference and present numerical experiments in Section 7.4. Lastly we provide local stability analysis in Section 7.5.

7.1 Introduction

Diagnostic inference is both a difficult and important problem that has attracted much attention in the last few decades. The ultimate goal is to design an intelligent system that assists in diagnosing patients who present with a set of observed findings (or symptoms) in a clinical setting. The computational engine behind this system must incorporate both statistical and expert knowledge into a systematic framework that performs inference. Bayesian networks are a natural fit for this task due to their ability to encode causal relationships and statistical information into a computational model. In addition, they are highly interpretable which is a vital quality in this type of medical application.

This chapter focuses on a medical diagnostic network called the Quick Medical Reference (QMR) network. This Bayesian network models relationships between diseases and findings on a directed bipartite graph that consists of a set of disease and finding nodes. The edges are directed from disease to finding nodes in order to reflect the causal relationship between these variables. The QMR network is a probabilistic model where each disease and finding pair has a corresponding conditional probability that represents the likelihood of the disease causing the finding to be present. This probability captures that findings may be highly correlated with certain diseases. The inference task is then to estimate the conditional probability of individual diseases being present given a set of observed findings. The outcome of this calculation is a probabilistic explanation of the cause behind the observed findings. Exact inference is computationally intractable because this problem has been proven to be NP-hard [16]. Heckerman (1989) used the unique structure of the QMR network to derive a more efficient method called the quickscore algorithm that performs exact inference [36]. However, this algorithm is limited to small cases because the runtime is exponential in the number of positive findings.

Due to the computational complexity of this problem, most efforts have centered around developing approximation algorithms. There was progress made on this problem in the early 1990s by using more classical techniques such as stochastic simulation and heuristic search-based methods. Shwe and Cooper (1990) developed an algorithm known as "likelihood-weighted sampling" that uses importance sampling to estimate the marginals of each disease [73]. Sampling-based methods are notoriously slow since determining effective proposal distributions is often difficult [71]. Shwe and Cooper improve the convergence rate of their method by using Markov blanket scoring and self-importance sampling where samples are drawn from an adaptive proposal distribution. Henrion (1991) developed a search-based algorithm that reduces the search space by using heuristics such as hypotheses with more than a few diseases are improbable and powerful pruning rules applicable to multi-layer networks [38]. Although search-based methods obtain excellent results in small networks, this approach generally doesn't scale well to large networks.

More modern approaches to this problem have focused on methods that are applicable to large-scale networks. Jaakola and Jordan (1999) developed a variational method that uses convex duality to obtain upper and lower bounds on the distribution of interest (see [45], [46]). Their experiments show that their method is more accurate and converges much faster than likelihood-weighted sampling. Yu et al. (2007) used a Lagrangian relaxation algorithm that involves minimizing an upper bound via sub-gradient descent (see [87], [88]). Their method is fast and applicable to very large cases (e.g. 1000 positive findings). Murphy et al. (2013) published an empirical study on approximate inference with loopy belief propagation where they evaluate its performance on several difficult inference tasks, including a large-scale QMR network with 600 diseases and 4000 findings [62]. They found that the algorithm computed very accurate estimates in every problem except two cases where loopy belief propagation oscillated.

The developments in this chapter are most closely related to the work by Murphy et al. Similarly, we also use belief propagation to perform inference, but this work differentiates itself by incorporating homotopy continuation. We present numerical experiments which show that the algorithm obtains accurate estimates and converges on every problem. Prior to performing these experiments, we did not anticipate that the algorithm would always converge. This observation has lead to a conjecture that this inference problem may have additional structure which guarantees that belief propagation with homotopy continuation always converges. At the end of this chapter, we include a section on local stability analysis that provides some insight into this behavior. Our initial our analysis focuses on the structure of the Jacobian which leads to an understanding of how the topology of the graph affects stability. Then we analyze the contents of the Jacobian which leads to an understanding of how the parameters in the QMR network affect the stability.

7.2 QMR-Network

This section presents the QMR network and describes the inference task. In Sections 7.2.1 and 7.2.2, we formally define the QMR network and describe the underlying assumptions in this model. Section 7.2.3 introduces the inference task and motivates the need for approximation algorithms. In addition, we provide a concrete example of a simple QMR network that models a few diseases and findings.

7.2.1 Probabilistic Framework

The QMR network is a directed bipartite graph $\mathcal{G} = (\mathcal{D} \cup \mathcal{F}, \mathcal{E})$, where $\mathcal{D} = \{1, \ldots, n\}$ is a set of disease nodes and $\mathcal{F} = \{n+1, \ldots, m\}$ is a set of finding nodes. The set $\mathcal{E}$ is a collection of directed edges $(i, j) \in \mathcal{E}$ such that $i \in \mathcal{D}$ and $j \in \mathcal{F}$, where the direction of the edge reflects that findings are dependent upon diseases. Each disease $i \in \mathcal{D}$ has a set of children nodes denoted by $Ch(i) = \{ j \, : \, (i, j) \in \mathcal{E} \} \subset \mathcal{F}$. Similarly, each finding $j \in \mathcal{F}$ has a set of parent nodes $Pa(j) = \{ i \, : \, (i, j) \in \mathcal{E} \} \subset \mathcal{D}$.

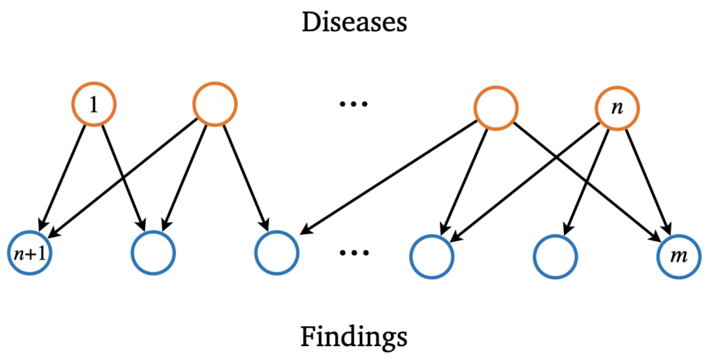


Figure 7.1: Example of an arbitrary QMR network. Orange circles represent diseases and blue circles represent findings. Edges are directed from diseases to findings as indicated by the arrows.

Let $D = (D_1, \ldots, D_n)$ and $F = (F_{n+1}, \ldots, F_m)$ be random vectors that represent the set of disease and finding nodes, respectively. Let $\Omega = \{0, 1\}$ be the set of outcomes of each random variable. Assume the convention that $D_i = 1$ (or $F_i = 1$) indicates the presence of a disease (or finding), while $D_i = 0$ (or $F_i = 0$) indicates the absence of a disease (or finding). A configuration of the random vector is denoted by $(d, f) = (d_1, \ldots, d_n, f_{n+1}, \ldots, f_m) \in \Omega^m$. The likelihood of a configuration is given by the joint distribution

$$\mathbb{P}(F = f, D = d) = \frac{1}{Z} \prod_i \mathbb{P}(F_i = f_i | \operatorname{Pa}(D)) \prod_j \mathbb{P}(D_i = d_j).$$

There are a number of underlying assumptions in this model that allow the joint distribution to be written in this form. Before discussing these assumptions, we present a concrete example of a simple QMR network that we develop throughout this section.

Example 7.2.1. *Let $\mathcal{G} = (\mathcal{D} \cup \mathcal{F}, \mathcal{E})$ be a bipartite graph with $\mathcal{D} = \{1, 2\}$ and $\mathcal{F} = \{3, 4, 5\}$ as shown in Figure 7.2. The cold and flu are represented by disease nodes $1, 2 \in \mathcal{D}$, respectively. When a patient has the flu, some possible findings (in this model) include a high temperature, sore throat and runny nose which are represented by finding nodes $3, 4, 5 \in \mathcal{F}$, respectively. Inference is challenging because different diseases can cause similar findings. In this example, we see that a patient with a cold may also present with a sore throat and runny nose.*

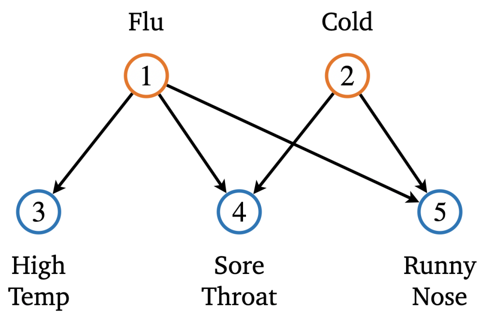


FIGURE 7.2: Simple QMR network from Example 7.2.1.

7.2.2 Assumptions in the Model

There are several underlying assumptions in this model whose purpose is to make probabilistic inference tractable.

1. *Marginal Independence.* The distribution over diseases is

$$\mathbb{P}(D = d) = \prod_{i=1}^{n} \mathbb{P}(D_i = d_i). \tag{7.1}$$

2. *Conditional Independence.* The conditional distribution over findings is

$$\mathbb{P}(F|D) = \prod_{i=n+1}^{m} \mathbb{P}(F_i = f_i|D = d). \tag{7.2}$$

Both of these assumptions are inherent to the bipartite structure of the network. For example, the marginal independence assumption is implied by the lack of edges between disease nodes. This assumption drastically reduces the complexity of the model since the probability of any configuration of the random vector D can be computed from individual disease priors.

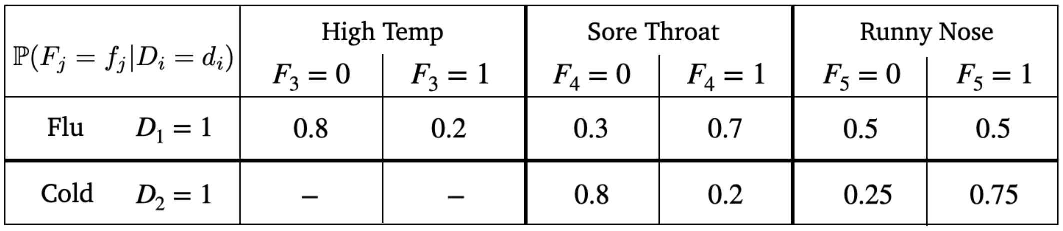
Although the conditional independence assumption also simplifies the joint distribution, the state of each individual finding is conditioned upon the state of the random vector D. This imposes a computational challenge because we must specify 2^n distributions corresponding to each finding node. To overcome this limitation, Pearl and Kim (1983) introduce the notion of *causal* independence in which the parents of a finding contribute independently to its state (see [37], [48]).

3. *Causal Independence.* This relationship is expressed with the leaky-OR model,

$$\begin{aligned}\mathbb{P}(F_i = 0|D = d) &= \mathbb{P}(F_i = 0|D = 0) \prod_{j \in Pa(i)} \mathbb{P}(F_i = 0|D_j = d_j) \\ &= (1 - p_{i0}) \prod_{j \in Pa(i)} (1 - p_{ij})^{d_j}. \end{aligned} \tag{7.3}$$

The *leak* probability $p_{i0} := \mathbb{P}(F_i = 1|D = 0)$ is the probability that a finding is caused by a means other than a disease included in the network. The inhibit probability $(1 - p_{ij}) := \mathbb{P}(F_i = 0|D_j = 1)$ is the probability that a finding is absent despite disease $j \in \mathcal{D}$ being present. Under this set of assumptions, the probability of any configuration of the random vector can be computed from the disease priors, conditional probabilities p_{ij}, and leak probabilities p_{i0}.

Example 7.2.2. *Next we further develop the QMR network from Example 7.2.1 by defining the disease priors, conditional probability table, and leak probability. Let $\mathbb{P}(D_i = 1) = 10^{-3}$ be the disease prior for all $i \in \mathcal{D}$ and let $p_{i0} = 0.01$ be the leak probability for all $i \in \mathcal{F}$. Lastly, the conditional probabilities p_{ij} are shown in the conditional probability table (CPT) shown in Figure 7.3.*



$\mathbb{P}(F_j = f_j \| D_i = d_i)$		High Temp		Sore Throat		Runny Nose	
		$F_3 = 0$	$F_3 = 1$	$F_4 = 0$	$F_4 = 1$	$F_5 = 0$	$F_5 = 1$
Flu	$D_1 = 1$	0.8	0.2	0.3	0.7	0.5	0.5
Cold	$D_2 = 1$	–	–	0.8	0.2	0.25	0.75

Figure 7.3: Conditional Probability Table (CPT)

7.2.3 Probabilistic Inference

Probabilistic inference in the QMR network involves computing posterior marginal distributions, which refer to the marginal distributions conditioned on some observed findings. Let F^+ and F^- be vectors that consist of *observed* positive and negative findings. In general, the number of observed findings tends to be much smaller than the total number of findings in the network. *Unobserved* findings have no effect on the posterior marginals, so these nodes are discarded from the network prior to performing inference.

The main objective is to compute the posterior marginal distribution of each disease. These quantities can be computed as

$$\begin{aligned}\mathbb{P}(D_i = d_i \,|\, F^+, F^-) &= \frac{1}{Z} \sum_{d_{\mathcal{D}\setminus i}} \mathbb{P}(D = d \,|\, F^+, F^-) \\ &\propto \sum_{d_{\mathcal{D}\setminus i}} \mathbb{P}(F^+ \,|\, D = d)\, \mathbb{P}(F^- \,|\, D = d) \prod_{i=1}^{n} \mathbb{P}(D_i = d_i).\end{aligned}$$

Although this is a difficult computational objective, there are some simplifications that can be made by exploiting the structure of the leaky-OR model. In this model, a finding is negative if none of its parent diseases cause it to be present and the leak does not cause it to be present. This implies that $\mathbb{P}(F^- | \, D)$ and $\mathbb{P}(D)$ both factorize over the diseases (see Equations (7.1) and (1.3)), so the negative evidence can be absorbed into the disease priors. This simplification can be carried out in linear time by performing the update

$$\mathbb{P}(D_i) \leftarrow \mathbb{P}(D_i | F^-) \tag{7.4}$$

with normalization.

For the remainder of this chapter, we assume that negative evidence has already been absorbed into the disease priors. In this case, the main objective is to compute

$$\mathbb{P}(D_i = d_i \,|\, F^+) \propto \sum_{d_{\mathcal{D}\setminus i}} \mathbb{P}(F^+ \,|\, D = d) \prod_{i=1}^{n} \mathbb{P}(D_i = d_i). \tag{7.5}$$

This calculation involves summing over all possible configurations of the disease vector which grows exponentially in the number of diseases. However, this objective can be further simplified by exploiting factorizations within this expression. Heckerman used this approach to develop an exact inference method called the quickscore algorithm. Although the runtime of quickscore is exponential in the number of positive findings, this is a significant improvement over Equation 7.5 because the number of positive findings is generally significantly less than the number of diseases in the network [36]. However, the quickscore algorithm is limited to roughly 20 positive findings.

Example 7.2.3. *Next we perform exact probabilistic inference in the QMR network from Examples 7.2.1 and 7.2.2. Let $F_4 = 1$ and $F_3 = 0$ be observed findings, then we use the quickscore algorithm to compute the marginals shown in Figure 7.4.*

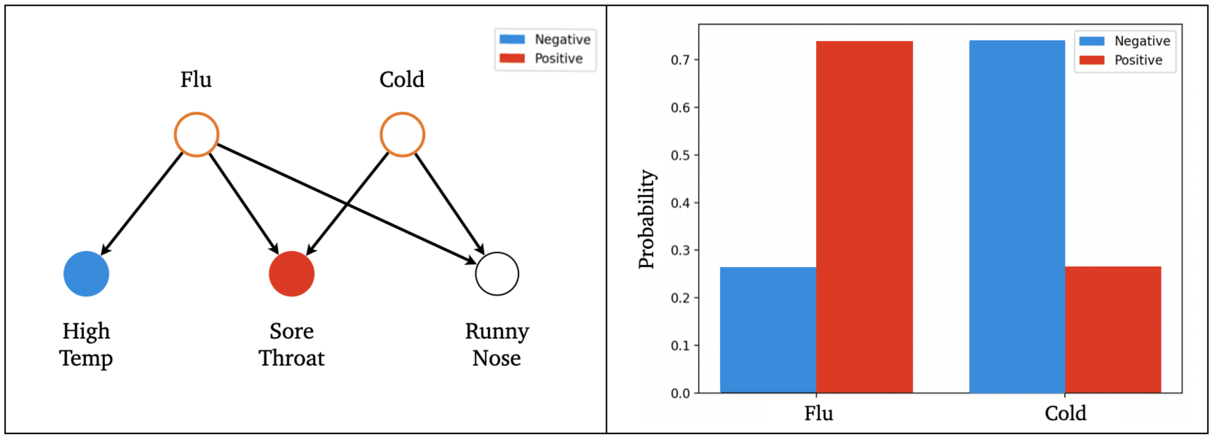


FIGURE 7.4: On the left, we see an illustration of observed findings from a QMR network. The blue node indicates that the patient does not have a high temperature (i.e. negative finding). The red node indicates that the patient has a sore throat (i.e. positive finding). The node corresponding to a runny nose is left uncolored to represent that it is unobserved. On the right, we see the exact posterior marginals computed with the quickscore algorithm.

7.3 Efficient Belief Propagation

Next we discuss using belief propagation to perform approximate inference in the QMR network. Section 7.3.1 describes transforming this network into a factor graph, then Sections 7.3.2 and 7.3.3 present efficient message passing operators for this application.

7.3.1 Factor Graph Model

Next we transform this network into a factor graph, then perform inference [23]. This transformation involves creating a factor node i' corresponding to each node i in the Bayesian network. This factor node i' is then connected to node i and the parents of node i (see Figure 7.5).

Let $G = (V \cup K, E)$ be an undirected bipartite graph with a set of variables nodes V and factor nodes K. We assume that G is the equivalent factor graph representation of the QMR network $\mathcal{G} = (\mathcal{D} \cup \mathcal{F}, \mathcal{E})$. Under this assumption, the variables nodes are $V = \mathcal{D} \cup \mathcal{F}$ and factor nodes are $K = (\mathcal{D} \cup \mathcal{F})'$, where the notation X' denotes a copy of the set X. The set E is a collection of undirected edges $\{i, g\} \in E$ such that $i \in V$ and $g \in K$.

Example 7.3.1. *Let $\mathcal{G} = (\mathcal{D} \cup \mathcal{F}, \mathcal{E})$ be the QMR network from Examples 7.2.1-7.2.3 in the previous section. Let $G = (V \cup K, E)$ be the equivalent factor graph representation, where $V = \{1, \dots, 5\}$ and $K = \{1', \dots, 5'\}$. Each factor node $i' \in K$ is a copy of node $i \in \mathcal{D} \cup \mathcal{F}$ from the original Bayesian network. These nodes are connected by an edge so that $\{i, i'\} \in E$. In addition, there is also an edge drawn between i' and the parents of node i so that $\big\{\{i', j\} : j \in Pa(i)\big\} \subset E$.*

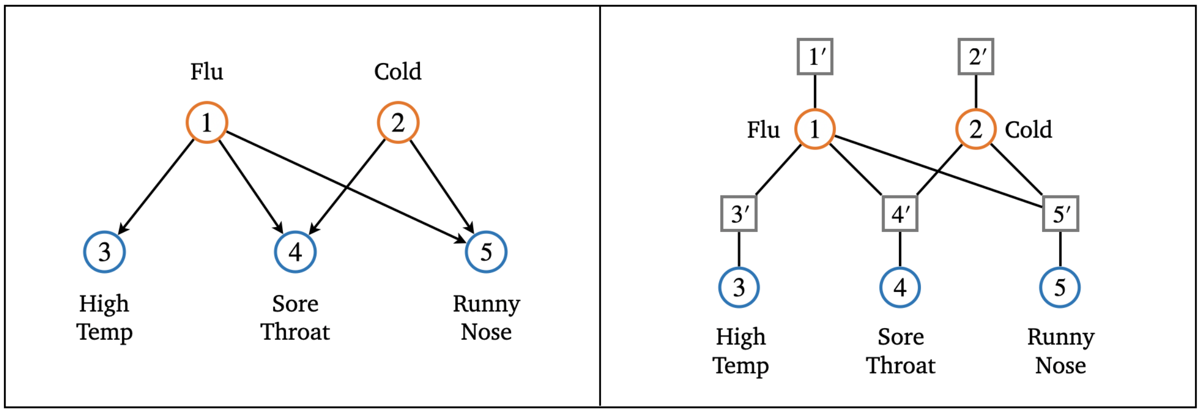


Figure 7.5: On the left, we see the QMR network from the previous section. On the right, we see an equivalent factor graph representation of this network.

In order to fully transform the QMR network into a factor graph, we must also specify the potential functions which compose the joint distribution. Due to the structure of the joint distribution (see Equation 7.2.1), this model consists of two types of potential functions, namely those corresponding to diseases and findings. Let $i' \in K$ is a factor node that corresponds to a finding, then the potential function $\Psi_{i'}$ is given by

$$\Psi_{i'}\big(f_i, d_{\mathcal{D}(i')}\big) = \begin{cases} (1-p_{i0}) \prod\limits_{j\in\mathcal{D}(i')} (1-p_{ij})^{d_j}, & \text{if } f_i = 0 \\ 1-(1-p_{i0}) \prod\limits_{j\in\mathcal{D}(i')} (1-p_{ij})^{d_j}, & \text{if } f_i = 1 \end{cases}$$

where $\mathcal{D}(i') = \{i : i \in \mathcal{N}(i') \cap \mathcal{D}\}$ is the set of disease nodes directly connected to $i' \in K$. When the factor $j' \in K$ corresponds to a disease, the potential function is given by

$$\Psi_{j'}(d_{j'}) = \mathbb{P}(D_{j'} = d_{j'}).$$

7.3.2 Traditional Message Passing

Although belief propagation dramatically improves the computational complexity of performing inference, it has a bottleneck that can be problematic. When the factor i' corresponds to a finding, the message passing operator S contains a sum over all possible states of the vector $x_{\mathcal{N}(i')}$ with $i' \in K$. In the case of the QMR network, the complexity of this operation is $\mathcal{O}\big(2^{\mathcal{N}(i')}\big)$ since each random variable may be either positive or negative. However, we can derive more efficient message passing equations by exploiting the structure of the leaky-OR model.

Proposition 7.3.2. *Let $i' \in K$ be a factor node which corresponds to the finding node $i \in V$, then the message passing operator S can be simplified as*

$$(S\nu)_{i'\to i}(0) = (1-p_{i0}) \prod_{j\in\mathcal{D}(i')} \big(1 - p_{ij}\nu_{j\to i'}(1)\big)$$
$$(S\nu)_{i'\to i}(1) = 1 - (S\nu)_{i'\to i}(0).$$

Proof. See appendix. □

Proposition 7.3.3. *Let $i' \in K$ be a factor node which corresponds to the finding node $i \in V$. Assume that node $j \in \mathcal{N}(i')$ corresponds to a disease node, then the message passing operator S*

can be simplified as

$$(S\nu)_{i'\to j}(0) = \kappa\, \nu_{i\to i'}(1)\Big(1-(1-p_{i0})\prod_{k\backslash j}\big(1-p_{ik}\nu_{k\to i'}(1)\big)\Big)$$
$$(S\nu)_{i'\to j}(1) = \kappa\, \nu_{i\to i'}(1)\Big(1-(1-p_{i0})(1-p_{ij})\prod_{k\backslash j}\big(1-p_{ik}\nu_{k\to i'}(1)\big)\Big),$$

where κ is used to denote the normalization factor.

Proof. See appendix. ☐

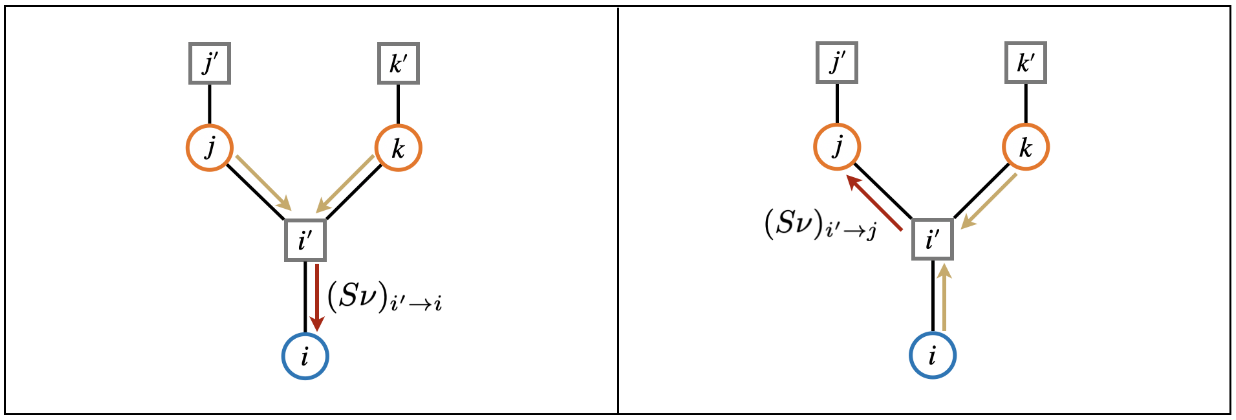


FIGURE 7.6: Illustration of messages sent by a factor that corresponds to a finding. This factor node sends messages to the corresponding variable node and the neighboring disease nodes as shown on the left and right. The indices used in this illustration match the notation in Propositions 7.3.2 and 7.3.3.

7.3.3 Convex Combination Message Passing

Next we present two simplifications of the message passing operator $\hat{S}$. These simplifications are analogous to the ones introduced in Propositions 7.3.2 and 7.3.3. Similarly, the runtime can also be reduced from being exponential to linear in the size of a neighborhood of the factor.

Proposition 7.3.4. *Let $i' \in K$ be a factor node which corresponds to the finding node $i \in V$, then the message passing operator $\hat{S}$ can be simplified as*

$$(\hat{S}\nu)_{i'\to i}(0) = (1-p_{i0})\prod_{j\in\mathcal{D}(i')}\big(1-p_{ij}\nu_{j\to i'}(1)^{w_{j\to i'}}\big)$$
$$(\hat{S}\nu)_{i'\to i}(1) = 1-(\hat{S}\nu)_{i'\to i}(0).$$

where $\sum_{j \in \mathcal{N}(i')} w_{j \to i'} = 1$ *for all* $i' \in K$ *and* $i \in \mathcal{N}(i')$*.*

Proof. This result holds by adapting the argument used to prove Proposition 7.3.2. □

Proposition 7.3.5. *Let* $i' \in K$ *be a factor node which corresponds to the finding node* $i \in V$*. Assume that node* $j \in \mathcal{N}(i')$ *corresponds to a disease node, then the message passing operator* $\hat{S}$ *can be simplified as*

$$(\hat{S}\nu)_{i' \to j}(0) = \kappa\, \nu_{i \to i'}(1)\Big(1 - (1 - p_{i0}) \prod_{k \setminus j} \big(1 - p_{ik}\nu_{k \to i'}(1)^{w_{k \to i'}}\big)\Big)$$

$$(\hat{S}\nu)_{i' \to j}(1) = \kappa\, \nu_{i \to i'}(1)\Big(1 - (1 - p_{i0})(1 - p_{ij}) \prod_{k \setminus j} \big(1 - p_{ik}\nu_{k \to i'}(1)^{w_{k \to i'}}\big)\Big)$$

where κ *denotes a normalization factor and* $\sum_{k \in \mathcal{N}(i'/j)} w_{k \to i'} = 1$ *for all* $i' \in K$ *and* $j \in \mathcal{N}(i') \setminus i$*.*

Proof. This result holds by using the argument used to prove Proposition 7.3.3. □

In Chapter 6, we discuss the limitations of convex combination belief propagation, namely that the accuracy may suffer on certain problems. The QMR network is an extreme example where this algorithm returns very inaccurate approximations. In fact, this is the exact problem that inspired the development of the homotopy continuation algorithm. Next we provide an example of a simple QMR network and inference problem where convex combination belief propagation returns a very poor approximation.

Example 7.3.6. *Let* $\mathcal{G} = (\mathcal{D} \cup \mathcal{F}, \mathcal{E})$ *be a QMR network with* $\mathcal{D} = \{1, \ldots, 5\}$ *and* $\mathcal{F} = \{6, \ldots, 25\}$*. The edges are generated randomly so that a given disease and finding are connected with probability 0.5. Let* $\mathbb{P}(D_i = 1) = 10^{-3}$ *be the disease prior and let* $p_{j0} = 0.01$ *be the leak probability. The conditional probabilities associated to the findings are sampled as* $p_{ji} \sim$ *Unif*$(0, 1)$*.*

To generate a problem, we do not set the observed findings by sampling from the joint distribution. All of the findings would be negative due to the small disease priors, and inference would be trivial. Instead we randomly select two diseases to be positive, then generate positive and negative findings by sampling the conditional distributions. Using this methodology, we obtain a sample where the hidden state of the disease vector is $D = (0, 0, 0, 1, 1)$ *and the observed findings consist of 10 positive and 5 negative findings.*

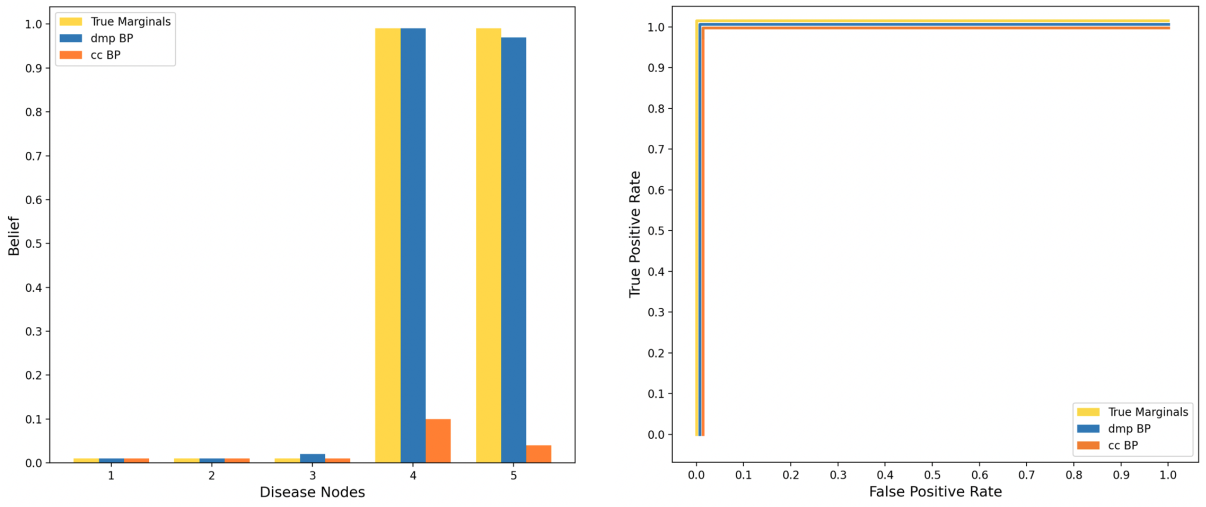


FIGURE 7.7: On the left, we see the beliefs obtained with damped and convex combination belief along with the true marginals. On the right, we see the ROC curve corresponding to each algorithm. Damped belief propagation (dmp BP) provides a good approximation and accurately predicts the hidden state of the diseases. In contrast, convex combination belief propagation (cc BP) obtains a poor approximation. It is interesting to see that although the beliefs are a poor approximation, the corresponding ROC curve is optimal.

We perform inference by transforming the network into a factor graph, then apply both damped and convex combination belief propagation to this example. In convex combination belief propagation, we use $\gamma = 0.49$ *and set the weights uniformly. We used the initial set of messages* $\mu^{(0)}_{g\to i} = (1,1)$ *and perform fixed point iteration with* $\hat{\Theta}$ *and* Θ_α *where* $\alpha = 0.5$*. In order to determine the accuracy of the resulting beliefs, we use quickscore to compute the true marginals. The results are shown in Figure 7.7, which shows the belief of each disease being positive (i.e.* $b_i(1)$ *for all* $i \in \mathcal{D}$*).*

Although convex combination belief propagation fails in certain settings, the algorithm redeems itself by being a vital component of the homotopy continuation algorithm. Despite convex combination belief propagation returning very poor results on the QMR network, the homotopy continuation algorithm has exceptional performance on this inference task. This algorithm obtains significantly more accurate results that convex combination belief propagation and exhibits much better convergence properties than traditional belief propagation. In fact, we conjecture that belief propagation with homotopy continuation may be guaranteed to converge on this problem. This conjecture is based on the experimental results in the next section.

7.4 Experiments

Next we describe a series of experiments that compare the performance of the three belief propagation algorithms discussed in this thesis. We apply each algorithm to a large number of problems, then evaluate the results with several performance metric. Each experiment is designed to push the limits of damped belief propagation, while also providing insight on how the parameters affect the inference task.

7.4.1 Experimental Settings

The main idea behind these experiments is to vary some parameter which influences the difficulty of performing inference. In order to maintain a controlled experiment, every parameter (except the one being varied) is kept constant across all problems. The purpose of this section is to describe how problems are generated and define the performance metrics used to evaluate the algorithms.

Problem Generation

Let $\mathcal{G} = (\mathcal{D} \cup \mathcal{F}, \mathcal{E})$ be a directed bipartite graph with the vertex sets $\mathcal{D} = \{1, \dots, 100\}$ and $\mathcal{F} = \{101, \dots, 300\}$. The edge set $\mathcal{E}$ is generated by connecting any given $i \in \mathcal{D}$ and $j \in \mathcal{F}$ with probability $p = 0.5$. The number of disease and finding nodes are kept constant in each problem, whereas edges are randomly generated for every problem. Each Bayesian network $\mathcal{G}$ is transformed into a factor graph $G = (V \cup K, E)$ in order to perform belief propagation.

The disease priors are initialized as $\mathbb{P}(D_i = 1) = 10^{-3}$ for all $i \in \mathcal{D}$. Let $p_{j0} = 0.01$ be the leak probability for all $j \in \mathcal{F}$. Let $p_{ji} \sim \text{Unif}(0.5, 1)$ be the conditional probability associated with each disease and finding pair. These probabilities are then used to generate the findings corresponding to a problem. We do not set the findings by directly sampling from the joint distribution because all of the findings would be negative due to the small disease priors. Instead we randomly select two diseases to be positive, then generate findings by sampling the conditional distributions.

Let N^+ and N^- be the number of observed positive and negative findings. N^+ must be bounded because quickscore is used to compute the true marginals for each problem. Since this algorithm becomes unstable when $N^+ > 20$, the number of observed positive findings is limited to $N^+ = 18$.

Although the observed negative findings do not impose a computational challenge, they do influence the performance of belief propagation. Inference becomes more trivial as the number of observed negative findings increases. Since we are interested in difficult problem instances, the number of observed negative findings is limited to $N^- = 10$.

Belief Propagation Parameters

For each belief propagation algorithm, we use the initialization $\mu_{g\to i}^{(0)} = (1, 1)$ and update the messages in parallel. Let $M = 10^3$ be the maximum number of iterations. Let $\epsilon = 10^{-2}$ be a threshold that indicates when the messages have sufficiently converged. In damped belief propagation, we use the damping factor $\alpha = 0.9$ to prioritize convergence over runtime since we consider difficult problems. In convex combination belief propagation, we use $\gamma = 0.49$ and set the weights uniformly. Since our continuation algorithm utilizes the message passing operators from damped and convex combination belief propagation, we use the same damping factors and weights in the homotopy operator. Let $dt = 0.1$ be the time step used in the homotopy continuation algorithm.

Performance Metrics

The performance of each algorithm is determined by the accuracy of the beliefs, rate of convergence, and average number of iterations until convergence. The accuracy of the beliefs is measured by the Kullback–Leibler (KL) divergence and average top-10 accuracy. Given that each problem is generated by two hidden diseases, we are interested in seeing whether the resulting beliefs can identify which diseases generated the observed findings. The average top-10 accuracy is the rate at which the beliefs corresponding to the hidden diseases are among the top-10 highest beliefs.

7.4.2 Varying Number of Negative Findings

It is well-known that small disease priors (typically on the order of 10^{-3}) tend to cause damped belief propagation to oscillate [62]. In this experiment, we use the number of observed negative findings N^- to indirectly vary the magnitude of the priors, as opposed to varying the interval from which the priors are sampled. Given that negative findings are absorbed into the priors (see Equation 7.4), a large number of negative findings results in smaller priors.

In this experiment, we vary N^- from 0 to 20 in increments of $\Delta N^- = 2$. For each value of N^-, we generate 100 distinct problems where all other parameters were initialized with the default settings described in Section 7.4.1. The performance of each belief propagation algorithm is determined by using the metrics described in Section 7.4.1.

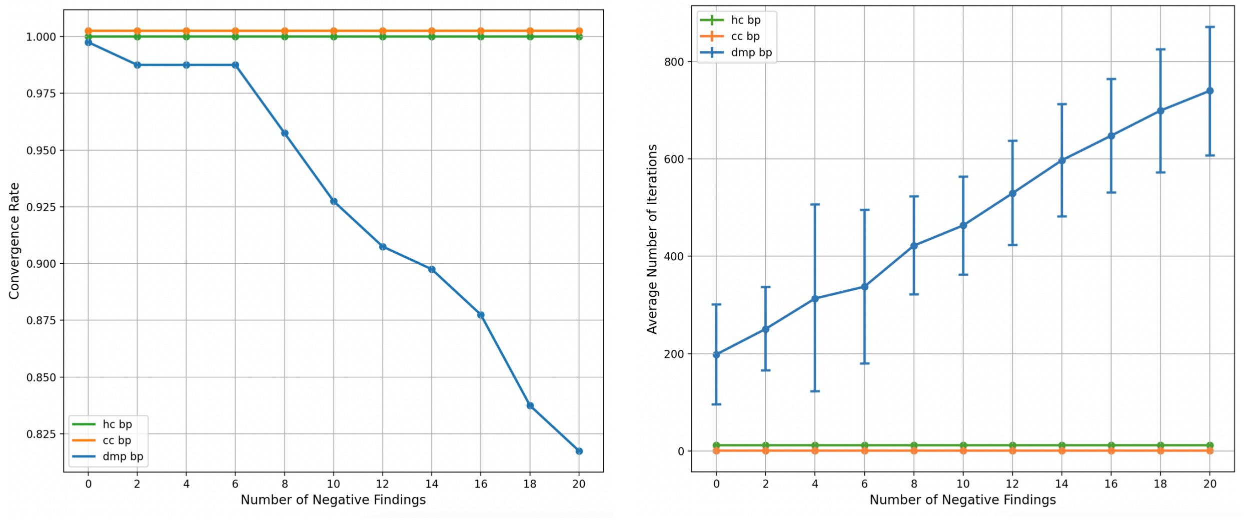


FIGURE 7.8: Convergence and run time of belief propagation algorithms

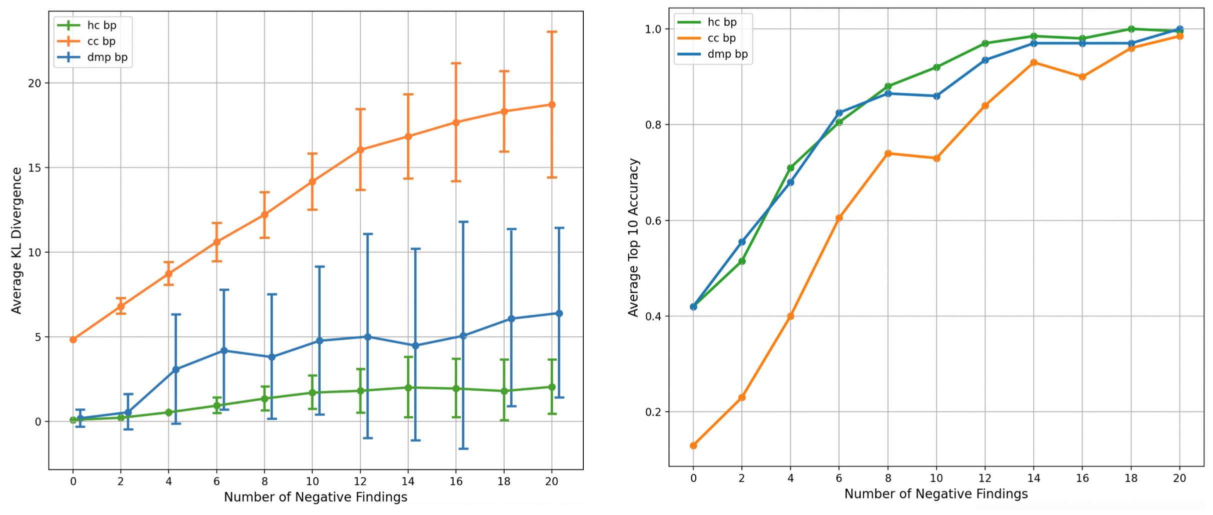


FIGURE 7.9: Accuracy of the belief propagation algorithms

The results shown in Figures 7.8 and 7.9 provide insight on how the inference task as a whole is affected by the number of negative findings. As expected, the convergence rate of damped belief propagation suffers as the number of negative findings increases. In conjunction, the runtime increases linearly with respect to this parameter. It is surprising to see that both the runtime and convergence rate of the homotopy continuation algorithm appear unaffected when this parameter is significantly varied.

An interesting outcome of this experiment is the effect upon the KL divergence and average 10-top accuracy. Given that belief propagation is more likely oscillate when there are a large number of negative findings, it is not surprising that the KL divergences increase as the number of negative findings increase. In contrast, the average top-10 accuracy improves as the number of negative findings increases. Intuitively, this outcome makes sense because more observed findings provides more useful information in determining which diseases generated the observation. This result is interesting because approximating the marginals becomes more challenging, while inferring which positive diseases generated the problem becomes more trivial.

7.4.3 Varying the Inhibit Probabilities

Next our aim is to understand how the inhibit probabilities affect the difficulty of the inference task. Murphy et al. conjectured that small inhibit probabilities may cause oscillations in the QMR network. The theory is that a problem with many positive findings would be very untypical in this parameter regime. As a result, belief propagation may oscillate since determining a probabilistic explanation is difficult.

We tested this hypothesis by varying the interval from which the inhibit probabilities were sampled. Let m be an upper bound on the inhibits so that $(1 - p_{ij}) \sim \text{Unif}(0, m)$. In this experiment, we vary m from 0.1 to 1 with increments of $\Delta m = 0.1$. For each value of m, we generate 100 distinct problems and carried out the experiment in the same manner as the previous ones. The results of this experiment in terms of the performance metrics are shown in Figures 7.10 and 7.11.

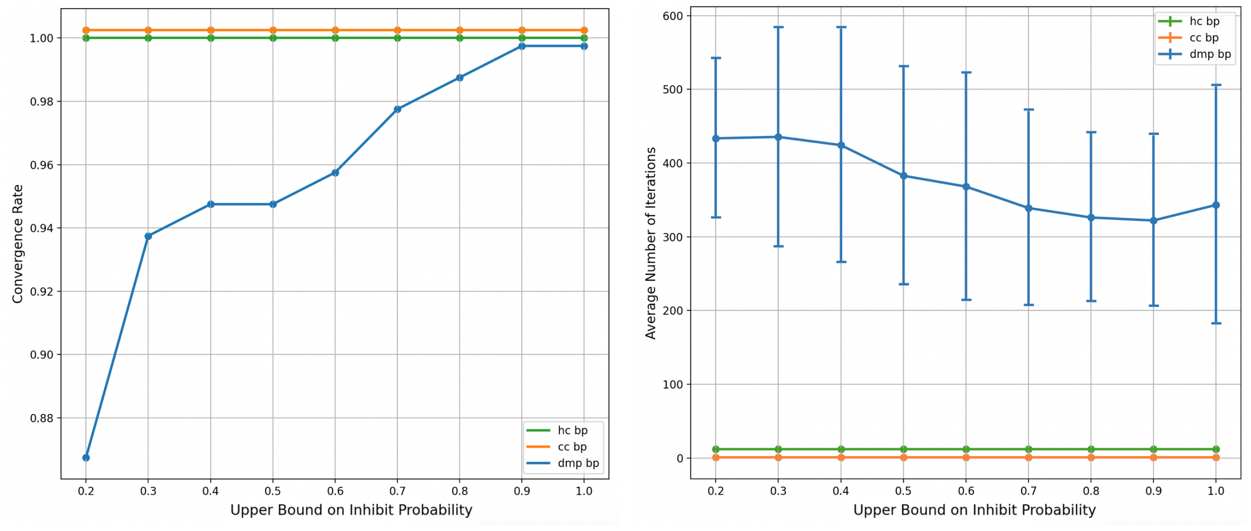


Figure 7.10: Convergence and run time of belief propagation algorithms

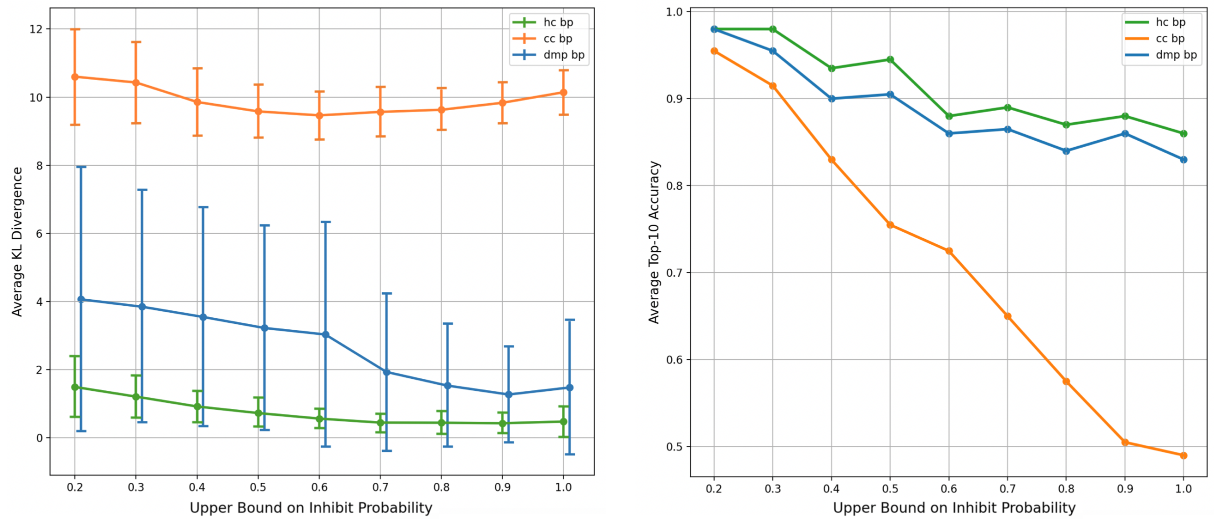


Figure 7.11: Accuracy of belief propagation algorithms

The results of this experiment confirm the hypothesis posed by Murphy et al., namely that small inhibit probabilities cause oscillations. Although this is a theory that may intuitively explain why this phenomena occurs, this explanation lacks a mathematical basis. One outcome of this chapter is that we relate the magnitude of the priors to the stability of belief propagation fixed points. We show that small priors cause fixed points to be less likely to satisfy a local stability condition.

7.4.4 Varying the Connectivity of the Graph

Belief propagation is also known to either fail to converge or return a poor approximation when the graph contains many cycles. In this experiment, we compare the performance of the belief propagation algorithms while gradually increasing the connectivity of the graph. We generate graphs with the Erdös-Renyi random graph model and vary the edge appearance probability p from 0 to 1 in increments of $\Delta p = 0.1$.

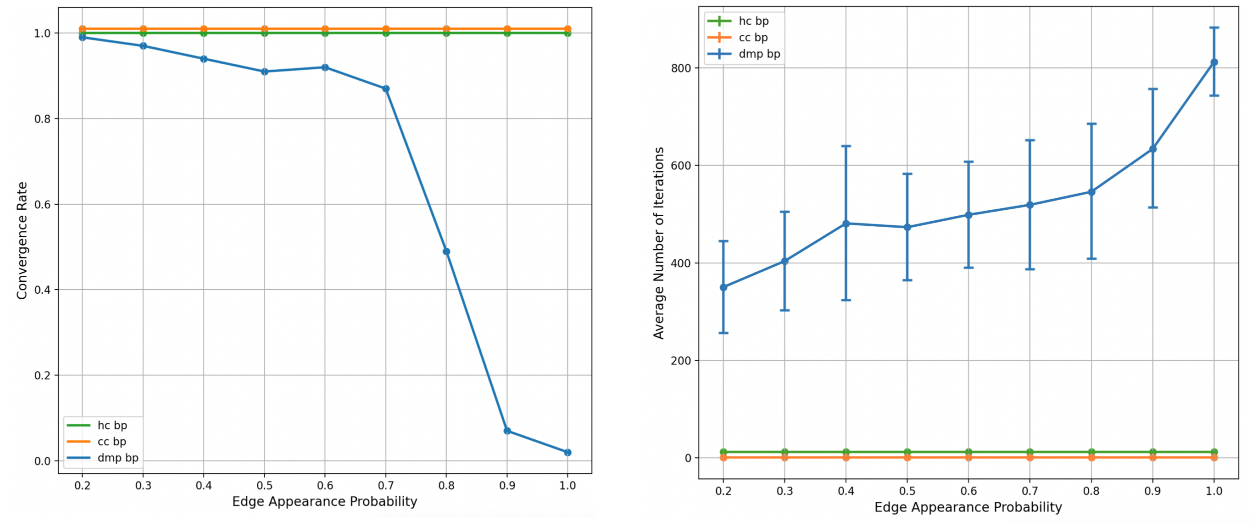


FIGURE 7.12: Convergence and run time of belief propagation algorithms

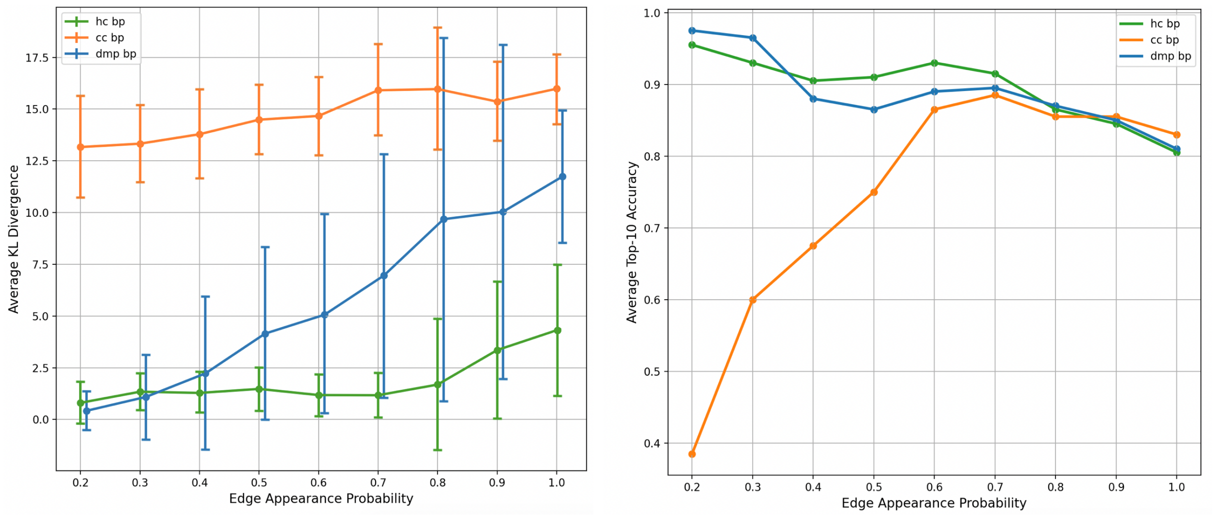


FIGURE 7.13: Accuracy of belief propagation algorithms

The results shown in Figures 7.12 and 7.13 confirm that loopy belief propagation is more likely to fail on graphs with complex topology. In Figure 7.12, we see that the algorithm converged for nearly all 100 problem instances when $p < 0.6$. As the edge appearance probability grows closer to 1, the algorithm rarely converges as expected. In contrast, we see that the homotopy continuation algorithm converges on all 100 problem instances for every value of p. This outcome is surprising since there is no theoretical guarantee that the algorithm converges.

The accuracy of the belief propagation algorithms is shown in Figure 14. As the edge appearance probability increases, the KL divergence also increases for all of the algorithms. It is interesting to see that although the continuation algorithm always converges, the KL divergence also increases with respect to the edge appearance probability. We hypothesize that there are many fixed points when p is large. Since the graph is very connected, there could be several highly probable ways to explain the observed findings. As a result, we see that the top-10 accuracy decreases because the algorithm converged to a fixed point which corresponds to an alternative explanation of the observed findings.

7.5 Local Stability Analysis

The main objective of this section is to analyze the local stability of belief propagation fixed points in the case of the QMR network. In Sections 7.5.3 and 7.5.4, our analysis focuses on the structure of the Jacobian which leads to an understanding of how the topology of the graph affects stability. Section 7.5.5 focuses on the contents of the Jacobian which leads to an understanding of how the parameters in the QMR network affects stability.

7.5.1 Convergence Conjecture

The most surprising outcome of our numerical experiments is that belief propagation with homotopy continuation converged[1] on every problem. We have performed additional experiments on larger networks in more difficult parameter regimes and observed the same convergence phenomena. These

[1]Note the homotopy continuation algorithm is said to *converge* if it converges on every time step, including the final time step when $t = 1$.

results raise the question of whether the QMR network is a special case where our continuation algorithm always converges.

Conjecture 7.5.1. *Belief propagation with homotopy continuation is guaranteed to converge when performing approximate inference in the QMR network.*

This conjecture is interesting because our algorithm is almost certainly not guaranteed to converge on arbitrary graphical models. In Chapter 6, we use this algorithm to perform approximate inference in the spin glass models from statistical physics [33]. This graphical model is generally used as a test case for alternative local message passing algorithms since loopy belief propagation often oscillate on this problem. In our experiments with spin glass models, belief propagation with homotopy continuation generally converges up to the time step $t = 0.95$, but the algorithm certainly does not always converge or even reach the final time step. One possible explanation is that the algorithm fails when it tracks a path of fixed points that becomes unstable. For our purposes, it is important to distinguish between global versus local stability.

Definition 7.5.2. *A fixed point $x^\star$ of an operator $F : X \to X$ is* ***globally stable*** *if $F^{(n)}(x_0) \to x^\star$ as $n \to \infty$ for any $x_0 \in X$.*

Definition 7.5.3. *A fixed point $x^\star$ of an operator $F : X \to X$ is* ***locally stable*** *if there exists an $\epsilon > 0$ such that $F^{(n)}(x_0) \to x^\star$ as $n \to \infty$ for any $x_0 \in X$ such that $\|x^\star - x_0\| < \epsilon$.*

One possible explanation of Conjecture 7.5.1 is that belief propagation fixed points are locally stable in the case of the QMR network. Under this assumption, numerical homotopy continuation would converge (in theory) because this implies the existence of a small enough time step such that the current approximate fixed point lies in the basin of attraction of a fixed point from the next time step. It is possible that belief propagation fixed points are locally stable, but the experiments in Section 7.4 suggest that the fixed points are not globally stable since damped belief propagation did not always converge. Thus, our analysis focuses on local stability of belief propagation fixed points.

7.5.2 Overview of General Approach

Our general approach is to consider belief propagation as a dynamical system, then utilize ideas from the theory of dynamical systems to understand the stability of this system. From this perspective, the conventional approach is to analyze the eigenvalues of the Jacobian of the operator. One well-known result is that a belief propagation fixed point is locally stable if the Jacobian at this point has all eigenvalues with modulus strictly smaller than one [60]. Conversely, a fixed point is unstable if at least one eigenvalue has a modulus strictly greater than 1. In the case of damped belief propagation, the stability criteria is much less strict as shown in Figure 7.14.

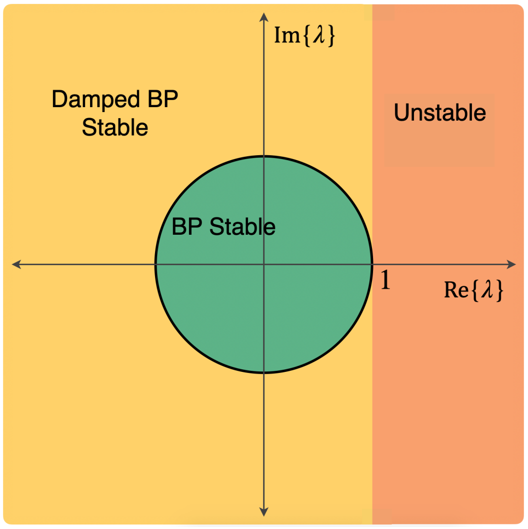


FIGURE 7.14: Eigenvalue spectrum of the Jacobian of the operator from loopy belief propagation. Loopy belief propagation without damping is locally stable when all eigenvalues are in the green region. Damped belief propagation is locally stable when all eigenvalues lie in the green and yellow regions. Both algorithms are unstable when the eigenvalues lie in the orange region[2].

Although this approach (in theory) leads to a conclusive understanding of when a fixed point is locally stable, this technique is difficult to apply to high-dimensional systems because the Jacobian is also very high dimensional. In the case of belief propagation, the dimension of the Jacobian is $\mathcal{O}(|E|)$, so this type of analysis is intractable. However, the unique structure of the QMR network can be exploited to drastically simplify the Jacobian and computation of its eigenvalues. One of the main results in this section is that this calculation can be reduced to computing eigenvalues of

[2]This figure was originally created by Christian Knoll in his PhD thesis [49].

a submatrix within the Jacobian with dimension $\mathcal{O}(|\mathcal{F}^+|)$, where $|\mathcal{F}^+|$ is the number of observed positive findings.

Our approach to obtaining this result is based upon a paper from 2012 by Martin, Lasgouttes, and Furtlehner [60]. In this work, they study the local stability of belief propagation with multiple fixed points. One of their main contributions is establishing a connection between the structure of the Jacobian and topology of the graph underlying the model. We apply some of their ideas to prove that the Jacobian consists of nested triangular matrices in the case of the QMR network. This fact can then be leveraged to drastically reduce the complexity of computing its eigenvalues.

7.5.3 Oriented Line Graphs

Let $\mathcal{L}(G) = (\mathcal{V}, \mathcal{E})$ be a directed graph referred to as the *oriented line graph* of the factor graph $G = (V \cup K, E)$. The vertex set is $\mathcal{V} := E$, where node $gi \in \mathcal{V}$ corresponds to the edge $\{g, i\} \in E$. The set $\mathcal{E}$ is a collection of directed edges in which gi is connected to hj if $j \in \mathcal{N}(g)$ with $i \neq j$ and $h \in \mathcal{N}(j)$ with $h \neq g$ such that $i, j, g, h \in V$. Thus, the entries in the adjacency matrix $\mathcal{A}$ are

$$\mathcal{A}^{gi}_{hj} := \mathbb{1}_{\{j \in \mathcal{N}(g),\, j \neq i\}} \mathbb{1}_{\{h \in \mathcal{N}(j),\, h \neq g\}}.$$

Oriented line graphs are a natural tool for studying stability because they provide an equivalent graphical representation of the non-zero entries in the Jacobian. This is an important realization because the non-zero entries play a critical role in determining when a fixed point is locally stable.

Each entry in the Jacobian of the operator Θ is a partial derivative of the form:

$$\frac{\partial(\hat{\Theta}\mu)_{g\to i}(x_i)}{\partial \mu_{h \to j}(x_j)} = \frac{\partial}{\partial \mu_{h\to j}(x_j)} \left(\kappa \sum_{x_{\mathcal{N}(g)}} \Psi_g(x_{\mathcal{N}(g)}) \prod_{j \in \mathcal{N}(g) \setminus i} \prod_{f \in \mathcal{N}(j) \setminus g} \mu_{f \to j}(x_j) \right),$$

where $\{g, i\}$, $\{h, j\} \in E$ and κ is a normalization factor. The partial derivative is non-zero if and only if the message $\mu_{h\to j}(x_j)$ appears in the update $(\Theta\mu)_{g\to i}(x_i)$. Due to the structure of the message passing operator, the partial derivative being non-zero depends entirely on the topological relationship between the edges $\{g, i\}, \{h, j\} \in E$ in the factor graph or, equivalently, vertices $gi, hj \in \mathcal{V}$ in the oriented line graph.

Observation 7.5.4. *The partial derivative of the message passing operator* Θ *given by*

$$\frac{\partial(\Theta\mu)_{g\to i}(x_i)}{\partial\mu_{h\to j}(x_j)} \tag{7.6}$$

is non-zero if and only if $j \in \mathcal{N}(g)$ *with* $i \neq j$ *and* $h \in \mathcal{N}(j)$ *with* $h \neq g$*.*

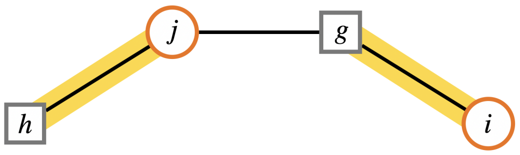


FIGURE 7.15: Edges involved in the partial derivative in Equation 7.6. This pair of edges corresponds to a non-zero partial derivative and an edge in the oriented line graph.

Topology of Oriented Line Graphs

The condition for the partial derivatives being non-zero is identical to the condition that specifies when vertices in the oriented line graph are connected. Thus, we can determine the structure of the Jacobian by equivalently analyzing the topology of this graph. One of the main results in this section is that the oriented line graph has modular structure when the factor graph corresponds to a QMR network. Note that we use the term *modular* to refer to a network that is composed of distinct components (i.e. subgraphs). Modularity is a useful property because it can be leveraged to prove that the Jacobian is a composition of nested triangular matrices. This is a useful property of the Jacobian because it drastically reduces the complexity of computing eigenvalues.

Next we provide a concrete example of determining the oriented line graph of a simple QMR network in order to describe its modular structure.

Example 7.5.5. *Let $G = (V \cup K, E)$ be the factor graph*[3] *of the QMR network from Example 7.3.1. In order to determine the oriented line graph of this factor graph, we first draw the vertices of the oriented line graph which correspond to edges in the factor graph (see Figure 7.16).*

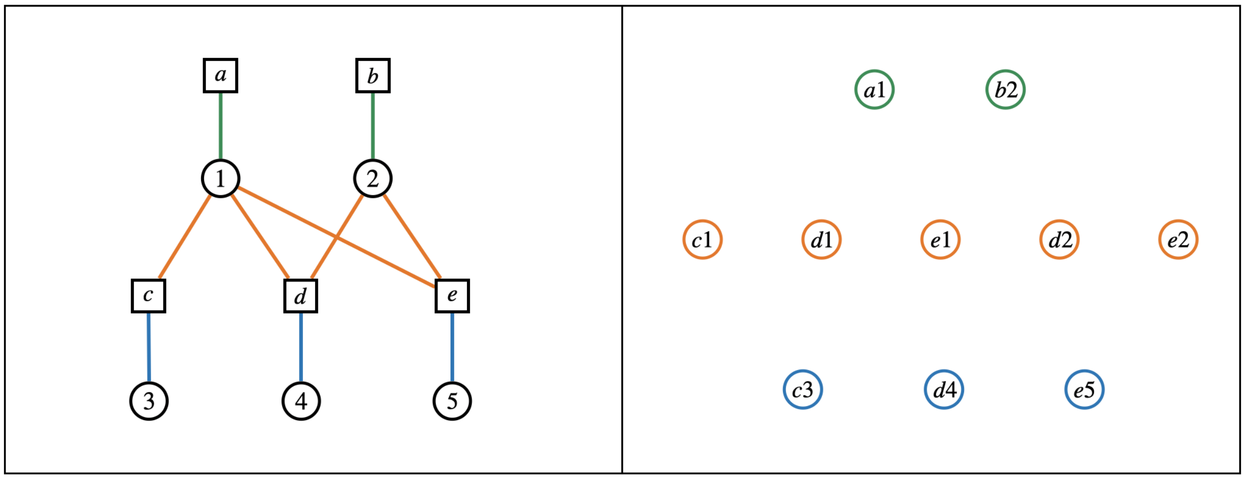


FIGURE 7.16: On the left, we see the factor graph from Example 7.3.1. On the right, we see the vertices of the corresponding oriented line graph. The colors represent the correspondence between types of edges in the factor graph and types of vertices in the oriented line graph.

The QMR network is a directed bipartite graph which leads to a multi-level factor graph consisting of four levels of nodes. This structure translates to three distinct types of nodes in the oriented line graph. Due to structure of the factor graph, it is only possible to have connections between certain types of vertices in the oriented line graph, which results in the modular structure shown in Figure 7.17.

[3]Note that we use different labels for the factor nodes.

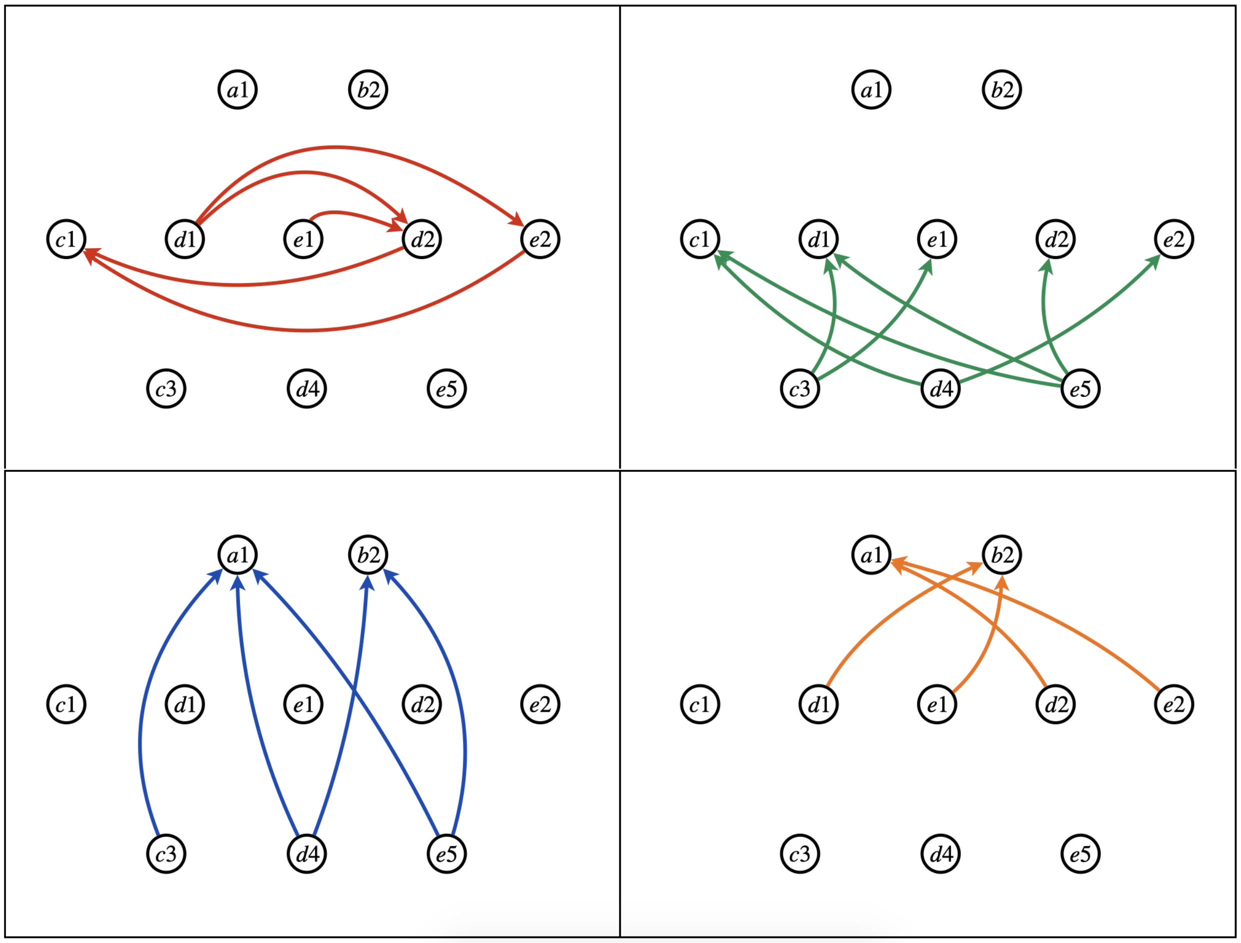


FIGURE 7.17: Modular structure of an oriented line graph. Each cell contains a subgraph (or module) and the oriented line graph is the superposition of these subgraphs.

In this example, we see that the modular structure of the oriented line graph stems from the QMR network being bipartite. The modularity of this graph results in an adjacency matrix that consists of nested triangular block matrices, where the blocks are formed by the three different types of vertices in the oriented line graph.

Definition 7.5.6. *Let $\mathcal{V}_1$ be the set of vertices in an oriented line graph that correspond to edges between disease variable nodes and factor nodes.*

Definition 7.5.7. *Let $\mathcal{V}_2$ be the set of vertices in an oriented line graph that correspond to edges between disease variable nodes and finding factor nodes.*

Definition 7.5.8. *Let $\mathcal{V}_3$ be the set of vertices in an oriented line graph that correspond to edges between finding variable nodes and factor nodes.*

Note that the three different types of vertices are highlighted in Figure 7.16. The set $\mathcal{V}_1$ corresponds to the green vertices, $\mathcal{V}_2$ corresponds to orange, and $\mathcal{V}_3$ corresponds to blue.

Next we provide a simple example of computing the adjacency matrix of the oriented line graph in Example 7.5.5. The purpose of this example is to show that the adjacency matrix consists of nested triangular matrices. Although this is a simple example of a more general result to be presented later, we start with this simple case because it illustrates most of the underlying ideas without getting involved with the additional technicalities of the general case.

Example 7.5.9. *Let $\mathcal{L}(G) = (\mathcal{V}, \mathcal{E})$ be the oriented line graph from Example 7.5.5. The adjacency matrix $\mathcal{A}$ of this graph is given by*

$$\mathcal{A} = \begin{array}{c|cc|ccccc:ccc} & a1 & b2 & c1 & d1 & e1 & d2 & e2 & c3 & d4 & e5 \\ a1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline c1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ d1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ e1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ d2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline c3 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ d4 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ e5 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}$$

where the blue characters indicate how the matrix is indexed by the vertices of the graph. The solid lines within this matrix specify a partition of $\mathcal{A}$ into four submatrices. Two of these submatrices are identically zero, so the adjacency matrix is a lower triangular block matrix,

$$\mathcal{A} = \begin{array}{c|cc} & \mathcal{V}_1 & \mathcal{V}_2 \cup \mathcal{V}_3 \\ \mathcal{V}_1 & 0 & 0 \\ \mathcal{V}_2 \cup \mathcal{V}_3 & A & B \end{array}.$$

The dashed lines in the adjacency matrix specify a partition of B into four submatrices. Similarly, B is also a lower triangular block matrix,

$$B = \begin{array}{c c} & \begin{array}{cc} \mathcal{V}_2 & \mathcal{V}_3 \end{array} \\ \begin{array}{c} \mathcal{V}_2 \\ \mathcal{V}_3 \end{array} & \begin{pmatrix} B_{11} & 0 \\ B_{21} & 0 \end{pmatrix} \end{array}.$$

This example shows that the adjacency matrix of the oriented line graph from Example 7.5.5 consists of two nested triangular matrices. Although this is a simple example, every oriented line graph corresponding to the factor graph of a QMR network has this exact same structure. Next we prove this fact by first defining the set $\mathcal{E}_{k.\ell}$ which consist of edges from vertices in $\mathcal{V}_k$ to $\mathcal{V}_\ell$ with $k, \ell \in \{1, 2, 3\}$. Then we prove that only four of these subsets are non-empty, which then implies that the oriented line graph of a QMR network is modular.

Definition 7.5.10. *Let $\mathcal{L}(G) = (\mathcal{V}, \mathcal{E})$ be an oriented line graph and define the set*

$$\mathcal{E}_{k,\ell} = \big\{ \{gi, hj\} \in \mathcal{E} \, : \, gi \in \mathcal{V}_k, \, hj \in \mathcal{V}_\ell \big\}$$

with $k, \ell \in \{1, 2, 3\}$.

Lemma 7.5.11. *Given any oriented line graph $\mathcal{L}(G) = (\mathcal{V}, \mathcal{E})$ of the factor graph $G = (V \cup K, E)$ of a QMR network, then $\mathcal{E}_{1,1} \cup \mathcal{E}_{1,2} \cup \mathcal{E}_{1,3} \cup \mathcal{E}_{2,3} \cup \mathcal{E}_{3,3} = \varnothing$.*

Lemma 7.5.12. *Given any oriented line graph $\mathcal{L}(G) = (\mathcal{V}, \mathcal{E})$, then $\mathcal{E} = \mathcal{E}_{2,1} \cup \mathcal{E}_{2,2} \cup \mathcal{E}_{3,1} \cup \mathcal{E}_{3,2}$ and there exists an $\mathcal{L}(G)$ in which these subsets are non-empty.*

Proof. Lemma 7.5.11 implies the first equality and Example 7.5.5 provides an example of an oriented line graph where these sets are non-empty. □

Next we prove that the oriented line graph of a factor graph corresponding to a QMR network consists of two nested triangular matrices. The main idea behind this argument is to define a partition of the matrix, then use Lemmas 7.5.11 and 7.5.12 to show that certain submatrices in this partition are always zero matrices. Note that the proof of this result bears a close resemblance to Example 7.5.9.

Theorem 7.5.1. *The adjacency matrix $\mathcal{A}$ of the oriented line graph of a factor graph corresponding to a QMR network is a lower triangular block matrix with the form:*

$$\mathcal{A} = \begin{array}{c c} & \begin{array}{cc} \mathcal{V}_1 & \mathcal{V}_2 \cup \mathcal{V}_3 \end{array} \\ \begin{array}{c} \mathcal{V}_1 \\ \mathcal{V}_2 \cup \mathcal{V}_3 \end{array} & \begin{pmatrix} 0 & 0 \\ A & B \end{pmatrix} \end{array}, \tag{7.7}$$

where the blue characters specify the partition used to obtain this block matrix. B is also a lower triangular block matrix with the form:

$$B = \begin{array}{c c} & \begin{array}{cc} \mathcal{V}_2 & \mathcal{V}_3 \end{array} \\ \begin{array}{c} \mathcal{V}_2 \\ \mathcal{V}_3 \end{array} & \begin{pmatrix} B_{11} & 0 \\ B_{21} & 0 \end{pmatrix} \end{array}. \tag{7.8}$$

Proof. To begin, we define a partition of the adjacency matrix $\mathcal{A}$ in the exact same manner as Example 7.5.5. Let the upper left block be indexed by $\mathcal{V}_1 \times \mathcal{V}_1$ and the lower right block be indexed by $(\mathcal{V}_2 \cup \mathcal{V}_3) \times (\mathcal{V}_2 \cup \mathcal{V}_3)$. Given that $\mathcal{E}_{1,1} = \varnothing$ by Lemma 7.5.11, this implies that the upper left block is a zero matrix. Similarly, the upper right block must also be a zero matrix because both $E_{1,2}$ and $E_{1,3}$ are also empty by Lemma 7.5.11. These two facts imply that the adjacency matrix $\mathcal{A}$ must be lower triangular and that the identity in Equation 7.7 holds.

Given the partition of $\mathcal{A}$, the submatrix B is indexed by the set $(\mathcal{V}_2 \cup \mathcal{V}_3) \times (V_2 \cup \mathcal{V}_3)$. Now we define a partition of B such that the upper left block B_{11} is indexed by $\mathcal{V}_2 \times \mathcal{V}_2$ and the lower right block is indexed by $\mathcal{V}_3 \times \mathcal{V}_3$. Given that both $\mathcal{E}_{2,3}$ and $\mathcal{E}_{3,3}$ are empty by Lemma 7.5.11, the upper and lower right blocks are zero matrices. The lower right block is also a zero matrix due to the set also being empty. This implies that B is be lower triangular and also verifies the identity in Equation 7.8. □

7.5.4 Jacobian of Belief Propagation

The main objective of this section is to draw a direct connection between the structure of the Jacobian and the adjacency matrix of the corresponding oriented line graph. This relationship can then be leveraged to infer that the Jacobian also consists of nested triangular matrices which significantly reduces the computational complexity of computing its eigenvalues.

Structure of Jacobian

Let $\mathcal{J}_\Theta$ denote the Jacobian of the message passing operator Θ from loopy belief propagation. Recall that each entry in $\mathcal{J}_\Theta$ is a partial derivative of the form:

$$\frac{\partial(\Theta\mu)_{g\to i}(x_i)}{\partial\mu_{h\to j}(x_j)} = \frac{\partial}{\partial\mu_{h\to j}(x_j)}\left(\kappa\sum_{x_{\mathcal{N}(g)}}\Psi_g(x_{\mathcal{N}(g)})\prod_{j\in\mathcal{N}(g)\setminus i}\ \prod_{f\in\mathcal{N}(j)\setminus g}\mu_{f\to j}(x_j)\right).$$

The Jacobian is indexed the set of edges in the factor graph. One important realization is that the Jacobian can be equivalently indexed by vertices in the corresponding oriented line graph. This observation is useful because it draws a connection between the Jacobian $\mathcal{J}_\Theta$ and adjacency matrix $\mathcal{A}$ of the corresponding oriented line graph.

The precise relationship between these matrices is that individual entries in the adjacency matrix correspond to 2×2 blocks in the Jacobian. Given any entry $\mathcal{A}_{gi,hj}$ in the adjacency matrix, this entry corresponds to the following 2×2 block in the Jacobian,

$$\mathcal{J}_{\{g,i\},\{h,j\}} := \begin{pmatrix} \dfrac{\partial(\Theta\mu)_{g\to i}(0)}{\partial\mu_{h\to j}(0)} & \dfrac{\partial(\Theta\mu)_{g\to i}(0)}{\partial\mu_{h\to j}(1)} \\ \\ \dfrac{\partial(\Theta\mu)_{g\to i}(1)}{\partial\mu_{h\to j}(0)} & \dfrac{\partial(\Theta\mu)_{g\to i}(1)}{\partial\mu_{h\to j}(1)} \end{pmatrix}. \tag{7.9}$$

$\mathcal{A}_{gi,hj}$ acts as an indicator function that indicates whether the 2×2 block in Equation 7.9 has any non-zero entries. This relationship is useful because it implies that the 2×2 block is a zero matrix when $\mathcal{A}_{gi,hj}=0$ due to Observation 7.5.4.

Lemma 7.5.13.

$$\mathcal{J}_{\{g,i\},\{h,j\}} = \begin{pmatrix} \dfrac{\partial(\Theta\mu)_{g\to i}(0)}{\partial\mu_{h\to j}(0)} & \dfrac{\partial(\Theta\mu)_{g\to i}(0)}{\partial\mu_{h\to j}(1)} \\ \\ \dfrac{\partial(\Theta\mu)_{g\to i}(1)}{\partial\mu_{h\to j}(0)} & \dfrac{\partial(\Theta\mu)_{g\to i}(1)}{\partial\mu_{h\to j}(1)} \end{pmatrix}\mathcal{A}_{gi,hj}.$$

Proof. This fact is a direct consequence of Observation 7.5.4. □

The result in Lemma 7.5.13 draws an important connection between the structure of the Jacobian and topology of the oriented line graph. There are two corollaries that follow from this Lemma. First, each entry in the adjacency matrix can be identified with a 2×2 block in the Jacobian. Since these matrices are indexed by equivalent sets, the Jacobian can be partitioned in the exact same manner described in Theorem 7.5.1. Second, Lemma 7.5.13 draws a correspondence between zero-valued entries in the adjacency matrix and zero submatrices in the Jacobian. This relationship can be leveraged to prove that the Jacobian has the exact same nested triangular structure as the adjacency matrix.

Theorem 7.5.2. *The Jacobian $\mathcal{J}_\Theta$ of the operator Θ is a lower triangular block matrix with the form*

$$\mathcal{J}_\Theta = \begin{pmatrix} 0 & 0 \\ C & D \end{pmatrix} \tag{7.10}$$

where D is also a lower triangular block matrix with the form

$$D = \begin{pmatrix} D_{11} & 0 \\ D_{21} & 0 \end{pmatrix} \tag{7.11}$$

Proof. The Jacobian $\mathcal{J}_\Theta$ is indexed by the edges in the corresponding factor graph. Since edges in the factor graph are identical to vertices in its oriented line graph, the Jacobian can equivalently be indexed by these vertices. Let $\mathcal{J}_\Theta$ be partitioned into the 2×2 block matrix given by

$$\mathcal{J}_\Theta = \begin{pmatrix} A & B \\ C & D \end{pmatrix},$$

where A is a composition of 2×2 submatrices indexed by $\mathcal{V}_1 \times \mathcal{V}_1$ and D is a composition of 2×2 submatrices indexed by $(\mathcal{V}_2 \cup \mathcal{V}_3) \times (\mathcal{V}_2 \cup \mathcal{V}_3)$. Note that this is the exact same partition of the adjacency matrix $\mathcal{A}$ used in Theorem 7.5.1.

The matrix A is composed of the submatrices $\mathcal{J}_{\{g,i\},\{h,j\}}$ such that $gi, hj \in \mathcal{V}_1$. Given that $\mathcal{A}_{gi,hj} = 0$ for all $gi, hj \in \mathcal{V}_1$ by Theorem 7.5.1, this implies that A is a zero matrix. In addition, B is also a zero matrix by a similar argument. Thus, these two facts imply that the Jacobian is lower block triangular and verify the identity in Equation 7.10.

Let be D partitioned into a 2×2 block matrix such that

$$D = \begin{pmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{pmatrix},$$

where D_{11} and D_{22} are compositions of 2×2 submatrices indexed by $\mathcal{V}_2 \times V_2$ and $\mathcal{V}_3 \times \mathcal{V}_3$, respectively. D_{12} and D_{22} are both zero matrices because the corresponding blocks in Equation 7.8 from Theorem 7.5.1 are zero matrices. □

Eigenvalues of Jacobian

Next we prove the main result of this section, namely that the complexity of computing eigenvalues of the Jacobian can be drastically simplified. The key to proving this result is to use that the Jacobian is a triangular block matrix. In this case, the eigenvalues of the entire matrix are a subset of eigenvalues of a submatrix within this matrix.

Definition 7.5.14. *Let A be an $n \times n$ real-valued matrix and let $\sigma(A)$ denote the set of eigenvalues of this matrix.*

Lemma 7.5.15. *Let M be the following lower block triangular matrix*

$$M = \begin{pmatrix} M_{11} & 0 \\ M_{21} & M_{22} \end{pmatrix},$$

then $\sigma(M) \subset \sigma(M_{11}) \cup \sigma(M_{22})$.

Corollary 7.5.16. $\sigma(\mathcal{J}_\Theta) \subset \sigma(D_{11}) \cup \{0\}$*, where D_{11} is the matrix from Theorem 7.5.2.*

Proof. Given that $\mathcal{J}_\Theta$ is lower triangular, then $\sigma(\mathcal{J}_\Theta) \subset \sigma(D) \cup \{0\}$ by Lemma 7.5.15. Similarly D is also lower triangular and so $\sigma(\mathcal{D}) \subset \sigma(D_{11}) \cup \{0\}$. These two facts imply the final result

$$\sigma(\mathcal{J}_\Theta) \subset \sigma(D) \cup \{0\} \subset \sigma(D_{11}) \cup \{0\}.$$

□

The main implication of this result is that the eigenvalues of the Jacobian can be obtained by computing the eigenvalues of the submatrix D_{11}. This is a significant improvement of the computational complexity of this problem because the Jacobian has dimension $\mathcal{O}(|E|)$ and D_{11} has dimension $\mathcal{O}(|\mathcal{F}^+|)$. For example, if we consider the QMR network that Murphy et al. discuss in [62], then $\dim(D_{11}) \leq 2 \cdot 20$ since they consider problems in which quickscore is applicable. In contrast, the dimension of the Jacobian is roughly $\dim(\mathcal{J}_\Theta) \approx 3640$. Since their paper does not report the number of edges between diseases and findings, we assume that any disease and finding pair is connected with probability 0.1 in this estimate. In our experiments, this probability is 0.5 which leads to $\dim(\mathcal{J}_\Theta) \approx 13,240$ in this calculation.

7.5.5 Local Stability Condition

The objective of this section is to use Corollary 7.5.16 to derive a local stability condition. We compute the entries of the Jacobian, then use Gershgorin's circle theorem to obtain an upper bound on the eigenvalues.

Computation of Jacobian

Next we compute the entries in the Jacobian. The

The main focus of this subsection is compute the Jacobian's entries. Since the ultimate is goal to bound the eigenvalues of the Jacobian, the calculation can be simplified by only computing entries in the submatrix D_{11} since $\sigma(\mathcal{J}_\Theta) \subset \sigma(D_{11})$ by Corollary 7.5.16. Each entry in this submatrix is given by

$$\frac{\partial(\Theta\mu)_{i'\to j}(x_j)}{\partial\mu_{h'\to\ell}(x_\ell)},$$

where $i', h' \in K$ correspond to findings and $j, \ell \in V$ correspond to diseases.

The analysis can be simplified by making the following change of variables:

$$\lambda_{i'\to j} := \mu_{i'\to j}(1) - \mu_{i'\to j}(0).$$

Some elementary algebraic manipulations show that the message update can be written as

$$\tilde{\lambda}_{i'\to j} := \frac{p_{ij} \prod_{k\backslash j} \left(1 - p_{ik}\phi_{k\to i'}\right)}{2 - (2 - p_{ij}) \prod_{k\backslash j} \left(1 - p_{ik}\phi_{k\to i'}\right)},$$

where $\phi_{k\to i'} := \prod_{g\backslash i'} \frac{\lambda_{g\to k} + 1}{2}$.

Under this change of variables, each entry in the Jacobian is given by

$$\frac{\partial \tilde{\lambda}_{i'\to j}}{\partial \lambda_{h'\to \ell}} = c_{ij}\, p_{ih} \prod_{k\backslash \{j,h\}} \left(1 - p_{ik}\phi_{k\to i'}\right) \tag{7.12}$$

where c_{ij} is given by

$$c_{ij} = \frac{p_{ij}}{\left(2 - (2 - p_{ij}) \prod_{k\backslash j} (1 - p_{ik}\phi_{k\to i'})\right)^2}. \tag{7.13}$$

Stability Condition

Next we derive a local stability condition that provides insight on how the parameters of the QMR network affect belief propagation fixed points. Recall that a belief propagation fixed point is locally stable if the Jacobian at this point has all eigenvalues with modulus strictly less than one [60]. Conversely, a fixed point is unstable if at least one eigenvalue has a modulus strictly greater than 1 (see Figure 7.14).

Thus, we derive a local stability condition by obtaining an upper bound on the modulus of the eigenvalues of the Jacobian. The key to deriving this bound is to use Gershgorin's circle theorem (see Theorem 7.5.3), which provides an upper bound on the eigenvalues of a matrix in terms of sums over individual rows [29].

Theorem 7.5.3. *Let A be an $n \times n$ matrix with entries denoted by a_{ij}. Then the eigenvalues of A lie within at least one of the Gershgorin discs $\mathscr{D}(a_{ii}, R_i) = \{x \in \mathbb{C} : |x - a_{ii}| < R_i\}$, where R_i is given by*

$$R_i = \sum_{i \neq j} |a_{ij}|.$$

Proof. See [29] for the proof. □

Now this theorem can be immediately applied to the submatrix D_{11}. One important detail is that all of the diagonal entries of D_{11} are zero because the factor graph does not contain any self-loops.

Lemma 7.5.17. *Let c_{ij} be the constant in Equation 7.13, then $c_{ij} \leq 1$.*

Proof. Let $\alpha_{kj} = \prod_{k \setminus j} (1 - p_{ik}\phi_{k \to i'})$ so that

$$c_{ij} = \frac{p_{ij}}{\big(2 - (2 - p_{ij})\alpha_{kj}\big)^2}.$$

The denominator of this expression is bounded below by

$$\begin{aligned}\big(2 - (2 - p_{ij})\alpha_{kj}\big)^2 &= 4 - 2(2 - p_{ij})\alpha_{kj} + (2 - p_{ij})^2\alpha_{kj} \\ &\geq 4 - 2(2 - p_{ij}) + (2 - p_{ij})^2\alpha_{kj} \\ &= 2p_{ij} + (2 - p_{ij})^2\alpha_{kj}^2.\end{aligned}$$

Consider the case when $p_{ij} \geq 1/2$, then the denominator can be further bounded below by

$$2p_{ij} + (2 - p_{ij})^2\alpha_{kj}^2 \geq 1 + (2 - p_{ij})^2\alpha_{kj}^2 \geq 1.$$

$$\implies c_{ij} \leq p_{ij} \leq 1 \qquad \text{if} \quad p_{ij} \geq \frac{1}{2}.$$

In the case when $p_{ij} < 1/2$, the desired inequality holds by

$$2p_{ij} + (2 - p_{ij})^2\alpha_{kj}^2 \geq 2p_{ij}$$

$$\implies c_{ij} \leq \frac{p_{ij}}{2p_{ij}} \leq \frac{1}{2} \qquad \text{if} \quad p_{ij} < \frac{1}{2}.$$

□

Theorem 7.5.4. *Let μ be a belief propagation fixed point, then this fixed point is locally stably if*

$$\max_{i' \in \mathcal{F}^+} \max_{j \in \mathcal{D}(i')} \left\{ c_{ij} \sum_{h \in \mathcal{F}(j)} p_{ih} \prod_{k \setminus \{j,h\}} \left(1 - p_{ik} \phi_{k \to i'}\right) \right\} < 1 \tag{7.14}$$

where $\mathcal{F}(j) \subset K$ denotes the set of findings connected to j and $c_{ij} \leq 1$.

Proof. This statement holds by applying Theorem 7.5.3 to the Jacobian computed in the previous section. Then Lemma 7.5.17 proves that $c_{ij} \leq 1$ □

APPENDIX A

Efficient Message Passing Equations

This appendix includes a derivation of efficient belief propagation message passing equations that can be used to perform inference in the QMR network.

Lemma A.0.1. *Let $f(x) = (f_1(x_1), \ldots, f_n(x_n))$ be a vector-valued function with $f_i : \{0,1\} \to [0,1]$ and assume that $f_i(0) + f_i(1) = 1$ for all $i = 1, \ldots, n$. Then*

$$\sum_{x \in \mathscr{P}(n)} \prod_{i=1}^{n} f_i(x_i) = \prod_{i=1}^{n} \sum_{x_i} f_i(x_i) = 1,$$

where $\mathscr{P}(n)$ denotes the power set of binary sequences with length n.

Proof. We use induction to prove this claim. The base case when $n = 1$ holds trivially, so assume that the claim is true in the case when f is an n-dimensional vector. Then the inductive step holds by

$$\begin{aligned}
\sum_{x \in \mathscr{P}(n+1)} \prod_{i=1}^{n+1} f_i(x_i) &= \sum_{\substack{x \in \mathscr{P}(n+1) \\ x_{n+1}=0}} \prod_{i=1}^{n+1} f_i(x_i) + \sum_{\substack{x \in \mathscr{P}(n+1) \\ x_{n+1}=1}} \prod_{i=1}^{n+1} f_i(x_i) \\
&= f_{n+1}(0) \sum_{x \in \mathscr{P}(n)} \prod_{i=1}^{n} f_i(x_i) + f_{n+1}(1) \sum_{x \in \mathscr{P}(n)} \prod_{i=1}^{n} f_i(x_i) \\
&= \big(f_{n+1}(0) + f_{n+1}(1)\big) \sum_{x \in \mathscr{P}(n)} \prod_{i=1}^{n} f_i(x_i) \\
&= \Bigg(\sum_{x_{n+1}} f_{n+1}(x_{n+1})\Bigg) \sum_{x \in \mathscr{P}(n)} \prod_{i=1}^{n} f_i(x_i)
\end{aligned}$$

Next we use the inductive hypothesis to obtain the final result

$$\begin{aligned}
&= \Big(\sum_{x_{n+1}} f_{n+1}(x_{n+1})\Big) \prod_{i=1}^{n} \sum_{x_i} f_i(x_i) \\
&= \prod_{i=1}^{n+1} \sum_{x_i} f_i(x_i) \\
&= 1.
\end{aligned}$$

□

Lemma A.0.2. *Let $f(x) = (f_1(x_1), \dots, f_n(x_n))$ be a vector-valued function with $f_i : \{0,1\} \to [0,1]$ and assume that $f_i(0) + f_i(1) = 1$ for all $i = 1, \dots, n$. Given any vector $q = (q_1, \dots, q_n) \in \mathbb{R}^n$, then*

$$\sum_{x \in \mathscr{P}(n)} \prod_{i=1}^{n} (1 - q_i)^{x_i} f_i(x_i) = \prod_{i=1}^{n} \big(1 - q_i f_i(1)\big)$$

where $\mathscr{P}(n)$ denotes the power set of binary sequences with length n.

Proof. We use induction to prove this claim and begin with the base case $n = 1$ which holds by

$$\begin{aligned}
\sum_{x \in \mathscr{P}(1)} (1 - q_i)^{x_i} f_i(x_i) &= f_1(0) + (1 - q_1) f_1(1) \\
&= 1 - f_1(1) + (1 - q_1) f_1(1) \\
&= 1 - q_1 f_1(1).
\end{aligned}$$

Now suppose the claim holds when f is an n-dimensional vector-valued function, then the inductive step holds by

$$\begin{aligned}
\sum_{x\in\mathscr{P}(n+1)} \prod_{i=1}^{n+1}(1-q_i)^{x_i} f_i(x_i) &= \sum_{x\in\mathscr{P}(n+1)} \prod_{\substack{i\\x_i=0}} (1-q_i)^{x_i} f_i(x_i) \prod_{\substack{i\\x_i=1}} (1-q_i)^{x_i} f_i(x_i) \\
&= \sum_{x\in\mathscr{P}(n+1)} \prod_{\substack{i\\x_i=0}} f_i(0) \prod_{\substack{i\\x_i=1}} (1-q_i) f_i(1) \\
&= f_{n+1}(0) \sum_{x\in\mathscr{P}(n)} \prod_{\substack{i\backslash n+1\\x_i=0}} f_i(0) \prod_{\substack{i\\x_i=1}} (1-q_i) f_i(1) \\
&+ (1-q_{n+1}) f_{n+1}(1) \sum_{x\in\mathscr{P}(n)} \prod_{\substack{i\\x_i=0}} f_i(0) \prod_{\substack{i\backslash n+1\\x_i=1}} (1-q_i) f_i(1) \\
&= \big(1-q_{n+1}f_{n+1}(1)\big) \sum_{x\in\mathscr{P}(n)} \prod_{i=1}^{n}(1-q_i)^{x_i} f_i(x_i) \\
&= \prod_{i=1}^{n+1} \big(1-q_i f_i(1)\big),
\end{aligned}$$

where the final line holds by using the inductive hypothesis. □

Proposition A.0.3. *Let $i' \in K$ be a factor node which corresponds to the finding node $i \in V$, then the message passing operator S can be simplified as*

$$\begin{aligned}
(S\nu)_{i'\to i}(0) &= (1-p_{i0}) \prod_{j\in\mathcal{D}(i')} \big(1-p_{ij}\nu_{j\to i'}(1)\big) \\
(S\nu)_{i'\to i}(1) &= 1-(S\nu)_{i'\to i}(0).
\end{aligned}$$

Proof. By definition, the message passing operator is

$$(S\nu)_{i'\to i}(\tau) = \kappa \sum_{d_{\mathcal{D}(i')}} \Psi_{i'}(\tau, d_{\mathcal{D}(i')}) \prod_{j\in\mathcal{D}(i)} \nu_{j\to i'}(d_j),$$

where κ is used to denote the normalization factor. Consider the case when $\tau = 0$, then the message passing operator can be simplified as

$$\begin{aligned}
(S\nu)_{i' \to i}(0) &= \kappa \sum_{d_{\mathcal{D}(i')}} \Psi_{i'}(0, d_{\mathcal{D}(i')}) \prod_j \nu_{j \to i'}(d_j) \\
&= \kappa \sum_{d_{\mathcal{D}(i')}} (1 - p_{i0}) \prod_j (1 - p_{ij})^{d_j} \prod_j \nu_{j \to i'}(d_j) \\
&= \kappa \, (1 - p_{i0}) \sum_{d_{\mathcal{D}(i')}} \prod_j (1 - p_{ij})^{d_j} \nu_{j \to i'}(d_j) \\
&= (1 - p_{i0}) \prod_{j \in \mathcal{D}(i')} \big(1 - p_{ij} \nu_{j \to i'}(1)\big).
\end{aligned}$$

Next, consider the case when $\tau = 1$ and the claim holds by

$$\begin{aligned}
(S\nu)_{i' \to i}(1) &= \kappa \sum_{d_{\mathcal{D}(i')}} \Psi_{i'}(1, d_{\mathcal{D}(i')}) \prod_j \nu_{j \to i'}(d_j) \\
&= \kappa \sum_{d_{\mathcal{D}(i')}} \big(1 - \Psi_{i'}(0, d_{\mathcal{D}(i')})\big) \prod_j \nu_{j \to i'}(d_j) \\
&= \kappa \sum_{d_{\mathcal{D}(i')}} \prod_j \nu_{j \to i'}(d_j) - \kappa \sum_{d_{\mathcal{D}(i')}} \Psi_{i'}(0, d_{\mathcal{D}(i')}) \prod_j \nu_{j \to i'}(d_j) \\
&= \kappa - \kappa \sum_{d_{\mathcal{D}(i')}} \Psi_{i'}(0, d_{\mathcal{D}(i')}) \prod_j \nu_{j \to i'}(d_j),
\end{aligned}$$

where the simplification on the last line holds by Lemma A.0.1. Now we conclude by using the definition of the operator so that

$$\begin{aligned}
&= \kappa - \kappa \, (S\nu)_{i' \to i}(0) \\
&= \kappa \, \big(1 - (S\nu)_{i' \to i}(0)\big).
\end{aligned}$$

Then final result is attained by setting $\kappa = 1$.

$\square$

Proposition A.0.4. *Let $i' \in K$ be a factor node which corresponds to the finding node $i \in V$. Assume that node $j \in \mathcal{N}(i')$ corresponds to a disease node, then the message passing operator S can be simplified as*

$$(S\nu)_{i'\to j}(0) = \kappa\, \nu_{i\to i'}(1)\Big(1 - (1-p_{i0}) \prod_{k\backslash j} \big(1 - p_{ik}\nu_{k\to i'}(1)\big)\Big)$$
$$(S\nu)_{i'\to j}(1) = \kappa\, \nu_{i\to i'}(1)\Big(1 - (1-p_{i0})(1-p_{ij}) \prod_{k\backslash j} \big(1 - p_{ik}\nu_{k\to i'}(1)\big)\Big)$$

where κ is used to denote the normalization factor.

Proof. By definition, the message passing operator is

$$\begin{aligned}(S\nu)_{i'\to j}(\tau) &= \kappa \sum_{\substack{d_{\mathcal{D}(i')}\\ d_j=\tau}} \sum_{f_i\in\{0,1\}} \Psi_{i'}(f_i, d_{\mathcal{D}(i')})\, \nu_{i\to i'}(f_i) \prod_{k\backslash j} \nu_{k\to i'}(d_k)\\ &= \kappa\, \nu_{i\to i'}(1) \sum_{\substack{d_{\mathcal{D}(i')}\\ d_j=\tau}} \Psi_{i'}(1, d_{\mathcal{D}(i')}) \prod_{k\backslash j} \nu_{k\to i'}(d_k),\end{aligned}$$

which holds by using that all observed findings are positive because negative findings are absorbed into the prior. Then this expression can be further simplified as

$$\begin{aligned}&= \kappa\, \nu_{i\to i'}(1) \sum_{\substack{d_{\mathcal{D}(i')}\\ d_j=\tau}} \big(1 - \Psi_{i'}(0, d_{\mathcal{D}(i')})\big) \prod_{k\backslash j} \nu_{k\to i'}(d_k)\\ &= \kappa\, \nu_{i\to i'}(1)\Big(\sum_{\substack{d_{\mathcal{D}(i')}\\ d_j=\tau}} \prod_{k\backslash j} \nu_{k\to i'}(d_k) - (1-p_{i0}) \sum_{\substack{d_{\mathcal{D}(i')}\\ d_j=\tau}} \prod_{k} (1-p_{ik})^{d_k} \prod_{k\backslash j} \nu_{k\to i'}(d_k)\Big)\\ &= \kappa\, \nu_{i\to i'}(1)\Big(1 - (1-p_{i0}) \sum_{\substack{d_{\mathcal{D}(i')}\\ d_j=\tau}} \prod_{k} (1-p_{ik})^{d_k} \prod_{k\backslash j} \nu_{k\to i'}(d_k)\Big),\end{aligned}$$

where the last line holds by Lemma A.0.1.

Now assume that $\tau = 0$, then the first simplified equation holds by

$$\begin{aligned}
(S\nu)_{i'\to j}(0) &= \kappa\, \nu_{i\to i'}(1)\Bigg(1 - (1-p_{i0}) \sum_{\substack{d_{\mathcal{D}(i')}\\ d_j=0}} \prod_{k} (1-p_{ik})^{d_k} \prod_{k\setminus j} \nu_{k\to i'}(d_k)\Bigg) \\
&= \kappa\, \nu_{i\to i'}(1)\Bigg(1 - (1-p_{i0}) \sum_{\substack{d_{\mathcal{D}(i')}\\ d_j=0}} \prod_{k\setminus j} (1-p_{ik})^{d_k}\, \nu_{k\to i'}(d_k)\Bigg) \\
&= \kappa\, \nu_{i\to i'}(1)\Bigg(1 - (1-p_{i0}) \prod_{k\setminus j} \big(1 - p_{ik}\nu_{k\to i'}(1)\big)\Bigg),
\end{aligned}$$

where the last line holds by Lemma A.0.2.

Now assume that $\tau = 1$, then the second simplified equation holds by

$$\begin{aligned}
(S\nu)_{i'\to j}(1) &= \kappa\, \nu_{i\to i'}(1)\Bigg(1 - (1-p_{i0}) \sum_{\substack{d_{\mathcal{D}(i')}\\ d_j=1}} \prod_{k} (1-p_{ik})^{d_k} \prod_{k\setminus j} \nu_{k\to i'}(d_k)\Bigg) \\
&= \kappa\, \nu_{i\to i'}(1)\Bigg(1 - (1-p_{i0})(1-p_{ij}) \sum_{\substack{d_{\mathcal{D}(i')}\\ d_j=0}} \prod_{k\setminus j} (1-p_{ik})^{d_k}\, \nu_{k\to i'}(d_k)\Bigg) \\
&= \kappa\, \nu_{i\to i'}(1)\Bigg(1 - (1-p_{i0})(1-p_{ij}) \prod_{k\setminus j} \big(1 - p_{ik}\nu_{k\to i'}(1)\big)\Bigg),
\end{aligned}$$

where the last line holds by Lemma A.0.2. $\square$

Bibliography

[1] S. Au and J. Beck. A new adaptive importance sampling scheme for reliability calculations. *Structural Safety*, 21(2):135–158, 1999.

[2] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988.

[3] A. Becker and D. Geiger. A sufficiently fast algorithm for finding close to optimal junction trees. In *UAI*, 1996.

[4] V. Berinde. *Iterative Approximation of Fixed Points.* Springer, 2007.

[5] C. Berrou and A. Glavieux. Near optimum error correcting coding and decoding: Turbo-codes. *IEEE Trans. Commun.*, 44:1261–1271, 1996.

[6] A. Blake and A. Zisserman. *Visual Reconstruction.* The MIT Press, 1987.

[7] E. Boros and P. Hammer. Network flows and minimization of quadratic pseudo-boolean functions. Technical report, RRR 17-1991, RUTCOR Research Report, 1991.

[8] E. Boros and P. Hammer. Pseudo-boolean optimization. *Discrete applied mathematics*, 123(1-3):155–225, 2002.

[9] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[10] T. Bullmore and D. Bassett. Brain graphs: graphical models of the human brain connectome. *Annual review of clinical psychology*, 7:113–40, 2011.

[11] J. Chen and M. Fossorier. Near optimum universal belief propagation based decoding of low-density parity check codes. *IEEE Transactions on communications*, 50(3):406–414, 2002.

[12] S. Chen, H. Tong, Z. Wang, S. Liu, M. Li, and B. Zhang. Improved generalized belief propagation for vision processing. *Mathematical Problems in Engineering*, 2011.

[13] J. Chua and F. Felzenszwalb. Scene grammars, factor graphs, and belief propagation. *Journal of the ACM*, 19, 2020.

[14] G. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[15] G. Cross and A. Jain. Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:25–39, 1983.

[16] P. Dagum and M. Luby. Approximating probabilistic inference in bayesian belief networks is np-hard. *Artificial intelligence*, 60(1):141–153, 1993.

[17] A. Daniilidis and C. Pang. Continuity and differentiability of set-valued maps revisited in the light of tame geometry. *Journal of the London Mathematics Society*, 83:637–658, 2011.

[18] R. Devaney. *An Introduction To Chaotic Dynamical Systems, Second Edition.* Avalon Publishing, 1989.

[19] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, 2004.

[20] P. Felzenszwalb and B. Svaiter. Diffusion methods for classification with pairwise relationships. *Quarterly of Applied Mathematics*, 77:793–810, 2019.

[21] M. Fossorier, M. Mihaljevic, and H. Imai. Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. *IEEE Transactions on communications*, 47(5):673–680, 1999.

[22] B. Frey. *Graphical models for machine learning and digital communication.* MIT press, 1998.

[23] B. Frey. Extending factor graphs so as to unify directed and undirected graphical models. In *UAI*, 2003.

[24] B. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[25] B. Frey and D. MacKay. A revolution: Belief propagation in graphs with cycles. In *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1997.

[26] A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin. *Bayesian Data Analysis, third edition.* Chapman & Hall, 2013.

[27] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[28] H. Georgii. *Gibbs measures and phase transitions.* de Gruyter, 2011.

[29] S. Gerschgorin. Über die abgrenzung der eigenwerte einer matrix. *Izv. Akad. Nauk. USSR Otd. Fiz.-Mat. Nauk*, 7:749–754, 1931.

[30] J. Gibbs. *Elementary Principles in Statistical Mechanics: Developed with Especial Reference to the Rational Foundation of Thermodynamics.* Yale University Press, 1902.

[31] A. Granas and J. Dugundji. *Fixed Point Theory.* Springer, 2003.

[32] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society: Series B (Methodological)*, 51(2):271–279, 1989.

[33] A. Grim and F. Felzenszwalb. Belief propagation algorithms on factor graphs with numerical homotopy continuation. *Preprint*, 2022.

[34] G. Grimmett. *Probability on graphs: random processes on graphs and lattices*, volume 8. Cambridge University Press, 2018.

[35] T. Hazan and A. Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. *IEEE Transactions on Information Theory*, 56(12):6294–6316, 2010.

[36] D. Heckerman. A tractable inference algorithm for diagnosing multiple diseases. In *UAI*, 1989.

[37] D. Heckerman and J. Breese. Causal independence for probability assessment and inference using bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 26(6):826–831, 1996.

[38] M. Henrion. Search-based methods to bound diagnostic probabilities in very large belief nets. In *UAI*, 1991.

[39] T. Heskes. Stable fixed points of loopy belief propagation are local minima of the bethe free energy. In *NIPS*, 2002.

[40] T. Heskes. On the uniqueness of loopy belief propagation fixed points. In *Neural Computations*, volume 16, pages 2397–2413, 2004.

[41] M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013.

[42] B. Huber and B. Sturmfels. A polyhedral method for solving sparse polynomial systems. *Mathematics of computation*, 64(212):1541–1555, 1995.

[43] A. Ihler, J. Fisher III, A. Willsky, and D. Chickering. Loopy belief propagation: convergence and effects of message errors. *Journal of Machine Learning Research*, 6(5), 2005.

[44] Hiroshi Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE transactions on pattern analysis and machine intelligence*, 25(10):1333–1336, 2003.

[45] T. Jaakkola and M. Jordan. Variational probabilistic inference and the qmr-dt network. *Journal of Artificial Intelligence Ressearch*, 10:291–322, 1999.

[46] M. Jordan, Z. Ghahramani, T. Jaakkola, and L Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

[47] M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. *arXiv preprint arXiv:1301.2281*, 2013.

[48] J. Kim and J. Pearl. A computational model for causal and diagnostic reasoning in inference engines. *Proc. 8th Int. Joint Conf. on Artificial intelligence*, 1983.

[49] C. Knoll. *Understanding the Behavior of Belief Propagation.* PhD thesis, Graz University of Technology, 2019.

[50] C. Knoll, D. Mehta, T. Chen, and F. Pernkopf. Fixed points of belief propagation—an analysis via polynomial homotopy continuation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(9):2124–2136, 2017.

[51] C. Knoll, Ad. Weller, and F. Pernkopf. Self-guided belief propagation–a homotopy continuation method. *arXiv preprint arXiv:1812.01339*, 2018.

[52] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

[53] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts-a review. *IEEE transactions on pattern analysis and machine intelligence*, 29(7):1274–1279, 2007.

[54] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *NIPS*, 2011.

[55] S. Kumagai. An implicit function theorem. *Journal of Optimization Theory and Applications*, 31:285–288, 1980.

[56] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*, 2001.

[57] S. Li. Markov random field models in computer vision. In *European Conference on Computer Vision*, pages 361–370. Springer, 1994.

[58] S. Li. *Markov Random Field Modeling in Image Analysis.* Springer Science & Business Media, 2009.

[59] A. Markov. *Wahrscheinlichkeitsrechnung*. 1912.

[60] V. Martin, M. Lasgouttes, and C. Furtlehner. Local stability of belief propagation algorithm with multiple fixed points. In *Frontiers in Artificial Intelligence and Applications*, volume 241, pages 180–191, 2012.

[61] M. McCoy and T. Wu. *The Two-Dimensional Ising Model*. Harvard University Press, 2013.

[62] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, 1999.

[63] L. Onsager. Crystal statistics. i. a two-dimensional model with an order-disorder transition. *Physical Review*, 65:117–149, 1944.

[64] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: networks of Plausible Inference*. Morgan Kauffman, 1988.

[65] J. Pearl. *Causality: Models*. Cambridge University Press, 2009.

[66] R. Peierls. On ising's model of ferromagnetism. 1936.

[67] W. Petryshyn. Construction of fixed points of demicompact mappings in hilbert space. *Journal of Mathematical Analysis and Applications*, 14(12):276–284, 1966.

[68] N. Peyrard and S. et al. Givry. Exact and approximate inference in graphical models: Variable elimination and beyond. *ArXiv*, abs/1506.08544, 2015.

[69] M. Pretti. A message-passing algorithm with damping. *Journal of Statistical Mechanics: Theory and experiment*, 2005.

[70] G. Rebane and J. Pearl. The recovery of causal poly-trees from statistical data. *Proceedings, 3rd Workshop on Uncertainty in AI*, page 222–228, 1987.

[71] B. Ripley. *Stochastic Simulation*. John Wiley & Sons, Inc., 1987.

[72] T. Roosta, M. Wainwright, and S. Sastry. Convergence analysis of reweighted sum-product algorithms. *IEEE Transactions on Signal Processing*, 56(9):4293–4305, 2008.

[73] M. Shwe and G. Cooper. An empirical analysis of likelihood-weighting simulation on a large, multiply-connected belief network. *Computers and biomedical research, an international journal*, 24:453–75, 1991.

[74] J. Sun, N. Zheng, and H. Shum. Stereo matching using belief propagation. *IEEE Transactions on pattern analysis and machine intelligence*, 25(7):787–800, 2003.

[75] S. Tatikonda. Convergence of the sum-product algorithm. In *Proceedings 2003 IEEE Information Theory Workshop*, pages 222–225, 2003.

[76] S. Tatikonda and M. Jordan. Loopy belief propagation and gibbs measures. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, page 493–500. Morgan Kaufmann Publishers Inc., 2002.

[77] M. Wainwright, T. Jaakkola, and A. Willsky. Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching. In *AISTATS*, 2003.

[78] M. Wainwright, T. Jaakkola, and A. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7):2313–2335, 2005.

[79] Y. Watanabe and K. Fukumizu. Graph zeta function in the bethe free energy and loopy belief propagation. In *NIPS*, 2009.

[80] Y. Weiss. Belief propagation and revision in networks with loops. *Technical Report*, 1997.

[81] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural computation*, 12(1):1–41, 2000.

[82] Y. Weiss and W. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13:2173–2200, 2001.

[83] M. Welling. On the choice of regions for generalized belief propagation. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, page 585–592. AUAI Press, 2004.

[84] S. Wright. Correlation and causation. *Journal of Agriculture Research*, 20:162–177, 1921.

[85] J. Yedidia, W. Freeman, and Y. Weiss. Generalized belief propagation. *Advances in neural information processing systems*, 13, 2000.

[86] J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

[87] F. Yu, F. Tu, H. Tu, and K. Pattipati. Multiple disease (fault) diagnosis with applications to the qmr-dt problem. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics)*, volume 2, pages 1187–1192, 2003.

[88] F. Yu, F. Tu, H. Tu, and K. Pattipati. A lagrangian relaxation algorithm for finding the map configuration in qmr-dt. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 37(5):746–757, 2007.

[89] A. Yuille. A double-loop algorithm to minimize the bethe free energy. In *EMMCVPR*, 2001.

[90] A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Comput.*, 15(4):915–936, 2003.

[91] Y. Zhang, M. Brady, and S. Smith. Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *IEEE Transactions on Medical Imaging*, 20(1):45–57, 2001.

[92] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. *Technical Report CMU-CALD-02-107*, 2002.