UNIVERSITY OF CALIFORNIA,
IRVINE

Reasoning and Decisions in Probabilistic Graphical Models – A Unified Framework

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Qiang Liu

Dissertation Committee:
Prof. Alexander Ihler, Chair
Prof. Rina Dechter
Prof. Padhraic Smyth

2014

DEDICATION

To my grandpa,
my wife and our parents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

First, I would like to express my highest appreciation and thanks to my advisor Alex Ihler, for his great advice and enormous support through my graduate career. Alex has been a great mentor, teacher and friend, who guided me along my research journey, not only pointing out important directions, providing countless support, but also offering a great amount of freedom for me to explore new exciting topics. I have been greatly benefited from the hundreds of discussions with Alex, when he can always rapidly point out the fallacies in my ideas, and suggest ways to fix them. As a non-native English speaker, I greatly appreciate Alex's extremely patient guidance and help on academic writing and presentation, as well as other perspectives needed for success beyond research. For this thesis in particular, Alex devoted a significant amount of time and energy to give comments and suggestions, edit and revise the text word by word, and play around with latex magics, including creating this latex template. Without him, this thesis and most of my papers would have been impossible. Alex's strong sense of responsibility, hard working practices, and commitment to perfection have made a profound impact on me and have challenged me to do the same. Although I can only absorb a small part of the wisdom he shared over the years, I have hugely benefited.

I would like to thank the two other members of my committee – Padhraic Smyth and Rina Dechter – for their time and comments, and for their constant support throughout my Ph.D. journey. Back when I was a student in the MCSB program, Padhraic provided me a first research opportunity in computer science, introducing me into the area of machine learning while allowing me to explore my interests freely. As I got more involved in probabilistic inference, I took Rina's class on belief networks, and was greatly inspired by her work; in fact, Chapter 5 was a direct result of a course project I did while in Rina's class. I deeply appreciate Padhraic's and Rina's constant encouragements and their generous help over the years.

I have also greatly benefited from my interaction and collaboration with other professors at UCI, including Max Welling, Mark Steyvers, Bogi Anderson, David van Dyk and Xiaohui Xie, to name only a few. The first year of my Ph.D. was supported by the Mathematical, Computational and Systems Biology (MCSB) program, where I benefited from the guidance and support of Qin Nie, Fred Wang, Arthur Lander and Jack Xin. Another two years of my research was supported by a Microsoft Research (MSR) graduate fellowship, as part of which I enjoyed a summer internship at MSR with Chris Meek and John Platt. I am also indebted to Denny Zhou of MSR for inspiring discussions and collaborations, and his countless support and help in both my professional and personal life.

Of course, my graduate experience has been largely shaped by my peers. Arthur Asuncion in Padhraic's group guided and collaborated with me on my first few papers, and constantly provided help and encouragement throughout the years. Andrew Frank was the person that I could refer to for all kinds of problems when I first came to computer science. Thanks to Andew Gelfand for offering mock interviews and providing lots of valuable suggestions on job searches. Outside of UCI, I would like to thank my close friends Jian Peng, Jason Lee

and Hongwei Li for their collaborations and inspiring discussions. There are too many other names to mention, so I will simply acknowledge the machine learning folks on the 4th floor of Donald Bren Hall, my MCSB classmates, as well as all of my Chinese friends over the years.

Finally, I would like to thank my family for a lifetime of love and support. I owe a special debt of gratitude to my wife Yi Zhang and her family, for their understanding and encouragement, especially during strenuous times.

In addition to a Microsoft research fellowship, my graduate work and this thesis have been supported in part by NIH NIAMS AR 44882, a CCBS Opportunity Award, and National Science Foundation awards IIS-1065618, IIS-1254071 and DMS-0928427.

CURRICULUM VITAE

Qiang Liu

EDUCATION

Doctor of Philosophy in Computer Science **2014**
University of California, Irvine Irvine, California

Bachelor of Science in Information and Computational Sciences **2008**
Beihang University China

RESEARCH EXPERIENCE

Graduate Research Assistant **2008–2014**
University of California, Irvine Irvine, California

Research Intern **Summer 2011**
Microsoft Research Redmond Redmond, Washington

SELECTED AWARDS

First Place, Shared Task Challenge in Crowdsourcing at Scale 2013

Microsoft Research Ph.D Fellowship 2011-2013

Notable Paper Award, AI & Statistics Conference 2011

RESEARCH EXPERIENCE

Graduate Research Assistant **2008–2014**
University of California, Irvine Irvine, California

Research Intern **Summer 2011**
Microsoft Research Redmond Redmond, Washington

TEACHING EXPERIENCE

Teaching Assistant **UC Irvine, California**
Machine Learning & Data Mining Fall 2012
Ensemble Learning Summer 2012
Math/Computational bootcamp Summer 2010

WORKSHOP CO-ORGANIZATION

Machine Learning Meets Crowdsourcing	ICML 2013
Crowdsourcing: Theory, Algorithms and Applications	NIPS 2013

ACADEMIC REVIEWING

Journal of Machine Learning Research (JMLR)	2013-2014
Neural Computation	2014
International Conference on Artificial Intelligence and Statistics (AISTATS)	2012-2014
International Conference on Machine Learning (ICML)	2012-2014
Conference on Neural Information Processing Systems (NIPS)	2012-2014
Conference on Uncertainty in Artificial Intelligence (UAI)	2012-2014

REFEREED JOURNAL PUBLICATIONS

Q. Liu, A. Ihler. Variational algorithms for marginal MAP. *Journal of Machine Learning Research (JMLR)*. 2013.

M. Geyfman, V. Kumar, **Q. Liu**, R. Ruiz, W. Gordon, F. Espitia, E. Cam, S.E. Millar, P. Smyth, A. Ihler, J.S. Takahashi, B. Andersen. Brain and muscle Arnt-like protein-1 (BMAL1) controls circadian cell proliferation and susceptibility to UVB-induced DNA damage in the epidermis. *Proc Natl Acad Sci USA doi:10.1073/pnas.120959210*. 2012.

Q. Liu, K.K. Lin, B. Anderson, P. Smyth, A. Ihler. Estimating Replicate Time-Shifts Using Gaussian Process Regression. *Bioinformatics* , 26(6), Mar. 2010, pp. 770-776; doi: 10.1093/bioinformatics/btq022

REFEREED CONFERENCE PUBLICATIONS

D. Zhou, **Q. Liu**, J.C. Platt, C. Meek; Aggregating Ordinal Labels from Crowds by Minimax Conditional Entropy. *International Conference on Machine Learning (ICML)*, June 2014.

W. Ping, **Q. Liu**, A. Ihler; Marginal structured SVM with hidden variables. *International Conference on Machine Learning (ICML)*, June 2014.

Q. Liu, M. Steyvers, A. Ihler. Scoring Workers in Crowdsourcing: How Many Control Questions are Enough? *Advances in Neural Information Processing Systems (NIPS)*. 2013.

Q. Cheng, **Q. Liu**, A. Ihler. Variational Planning for Graph-based MDPs. *Advances in Neural Information Processing Systems (NIPS)*. 2013.

Q. Liu, J. Peng, A. Ihler. Variational Inference for Crowdsourcing. *Advances in Neural Information Processing Systems (NIPS)*. 2012.

Q. Liu, A. Ihler. Belief Propagation for Structured Decision Making. *Uncertainty in Artificial Intelligence (UAI)*. 2012.

Q. Liu, A. Ihler. Distributed Parameter Estimation via Pseudo-likelihood. *International Conference on Machine Learning (ICML)*. 2012.

G. Zweig, J.C. Platt, C. Meek, C.J.C. Burges, A. Yessenalina, **Q. Liu**. Computational Approaches to Sentence Completion. *in ACL 2012, ACL/SIGPARSE*. 2012.

Q. Liu, A. Ihler. Variational algorithms for marginal MAP. *Uncertainty in Artificial Intelligence (UAI)* 2011.

Q. Liu, A. Ihler. Bounding the Partition Function using Hölder's Inequality. *International Conference on Machine Learning (ICML)*. 2011.

Q. Liu, A. Ihler. Learning Scale Free Networks by Reweighted L_1 Regularization. *AI & Statistics*. 2010. (notable paper)

Q. Liu, A. Ihler. Negative Tree Reweighted Belief Propagation. *Uncertainty in Artificial Intelligence (UAI)*. July 2010.

A. Asuncion, **Q. Liu**, A. Ihler, P. Smyth. Particle Filtered MCMC-MLE with Connections to Contrastive Divergence. *International Conference on Machine Learning (ICML)*. June 2010.

A. Asuncion, **Q. Liu**, A. Ihler, P. Smyth. Learning with Blocks: Composite Likelihood and Contrastive Divergence. *AI & Statistics (AISTATS)*. April 2010.

ABSTRACT OF THE DISSERTATION

Reasoning and Decisions in Probabilistic Graphical Models – A Unified Framework

By

Qiang Liu

Doctor of Philosophy in Computer Science

University of California, Irvine, 2014

Prof. Alexander Ihler, Chair

Probabilistic graphical models such as Markov random fields, Bayesian networks and decision networks (a.k.a. influence diagrams) provide powerful frameworks for representing and exploiting dependence structures in complex systems. However, making predictions or decisions using graphical models involve challenging computational problems of optimization and/or estimation in high dimensional spaces. These include *combinatorial optimization* tasks such as maximum *a posteriori* (MAP), which finds the most likely configuration, or *marginalization* tasks that calculate the normalization constants or marginal probabilities. Even more challenging tasks require a hybrid of both: marginal MAP tasks find the optimal MAP prediction while marginalizing over missing information or latent variables, while decision-making problems search for optimal policies over decisions in single- or multi-agent systems, in order to maximize expected utility in uncertain environments.

All these problems are generally NP-hard, creating a need for efficient approximations. The last two decades have witnessed significant progress on traditional optimization and marginalization problems, especially via the development of variational message passing algorithms. However, there has been less progress on the more challenging marginal MAP and decision-making problems.

This thesis presents a unified variational representation for all these problems. Based on our framework, we derive a class of efficient algorithms that combines the advantages of several existing algorithms, resulting in improved performance on traditional marginalization and optimization tasks. More importantly, our framework allows us to easily extend most or all existing variational algorithms to hybrid inference and decision-making tasks, and significantly improves our ability to solve these difficult problems. In particular, we propose a spectrum of efficient belief propagation style algorithms with "message passing" forms, which are simple, fast and amenable to parallel or distributed computation. We also propose a set of convergent algorithms based on proximal point methods, which have the nice form of transforming the hybrid inference problem into a sequence of standard marginalization problems. We show that our algorithms significantly outperform existing approaches in terms of both empirical performance and theoretical properties.

Chapter 1

Introduction

Probabilistic modeling plays a central role in many modern data-oriented applications. Probability theory and Bayesian inference provide the basic tools for modeling uncertainty and quantifying our beliefs about the different possible states of the world, including reasoning about the unknown variables given observed evidence, and even guiding our actions to select decisions to optimize some reward or cost function. While the application of probability theory has a long history, it is only more recently that it has been put to effective use on large scale problems, e.g., those involving many inter-related variables, and these advances are due largely to the development of the framework of *probabilistic graphical models*.

Probabilistic graphical models use graph-based representations to efficiently encode and manipulate relationships and probabilities in high dimensional spaces, often involving hundreds or even many thousands of variables. Graphical models include factorized distributions such as Bayesian networks and Markov random fields, which capture and exploit the conditional independence structure among random variables, as well as more general decision theoretic models such as influence diagrams (a.k.a. decision networks), which in additional to random variables also include structured decision and utility components for modeling high dimensional, sequential, and even multi-agent decision making processes. These models have been widely used in an enormous range of application domains, including computer vision, robotics, natural language processing, computational biology, and many more; see

e.g., Pearl [1988], Darwiche [2009], Koller and Friedman [2009], and Wainwright and Jordan [2008] for general overviews. In general, any area that involves uncertainty and large numbers of interacting variables is likely to benefit from using the graphical model framework.

However, making predictions or decisions using graphical models raises challenging computational problems that involve optimization and/or estimation in high dimensional spaces. For example, predictions with structured, high dimensional objectives are often framed as combinatorial optimization, or *max-inference* tasks – sometimes called maximum *a posteriori* (MAP) estimation – which maximize the joint probability to find a most-likely configuration over the high dimensional space. Other problems, such as evaluating the probability of an event, or computing the model likelihood for selecting better models, are often framed as marginalization, or *sum-inference tasks*, which requires summing over the probabilities in the high dimensional space to calculate marginal probabilities or the normalization constant.

Even more challenging problems involve making robust predictions or sequential decisions with missing or incomplete information, and require hybrids of both the optimization and marginalization operations in high dimensional spaces; these *mixed-inference* tasks form a major focus of this thesis. This hybrid setting includes the *marginal MAP*, or *max-sum inference* task[1] for making predictions with missing information or latent variables; these tasks require marginalizing over the latent variables, while maximizing over the target variables. More generally, the *maximum expected utility* task for decision networks searches for optimal policies for making decisions in single- or multi-agent systems, in order to maximize the expected utility value under the uncertain environment, sometimes with incomplete information sharing.

All these problems are generally NP-hard, creating a tremendous demand for efficient ap-

[1]Due to an unfortunate inconsistency in terminology, in Bayesian network literature the joint max-inference is called most probable explanation (MPE), and the marginal MAP (max-sum inference) is simply called MAP.

proximate algorithms. In this thesis, we consider two different perspectives, corresponding to two major styles of algorithms, for approximate inference: *variable elimination* based methods, such as mini-bucket elimination [Dechter and Rish, 2003], which act by approximately eliminating (maximizing or summing over) the variables one by one, while *variational optimization* methods, including loopy belief propagation Pearl [1988] and mean field Saul and Jordan [1995], are based on approximating an equivalent functional optimization problem over the space of distributions, by minimizing a divergence from the target distribution.

These two styles of approximation algorithms have their own advantages and disadvantages. For example, the mini-bucket method easily interpolates between costly, exact inference and efficient approximations using a single parameter, called the *ibound*; larger *ibound*s are more costly but typically more accurate. Moreover, since the algorithm is non-iterative, it is easy to estimate the computational requirements (memory and time) for any given *ibound*; but conversely, the results cannot be improved without increasing the *ibound*. On the other hand, variational approximation methods rely on iterative numerical optimization to obtain accurate results. These often offer better approximations than mini-bucket with low *ibound*, but the iterative updates can be slow or sometimes even fail to converge, and do not offer the same easy mechansims for identifying better but more computationally expensive approximations. New methods that combine the advantages of both approaches could have significant advantages in practice.

Another significant difference between these two styles of algorithms lies in their application to more difficult, hybrid tasks such as marginal MAP and MEU in decision networks. Although it is fairly straightforward to extend elimination-based methods such as mini-bucket elimination to hybrid tasks, by simply using the correct (sum or max) operators at each elimination step, extending variational approaches to these settings is less obvious and has not been previously studied.

This thesis develops a unified view of hybrid inference problems that allow variational techniques to be applied, and derives a number of new and powerful algorithms for solving hybrid or mixed inference tasks. In addition, we explore the connections between approximate elimination and variational inference methods, developing techniques that can exploit many of the advantages of both approaches, and are applicable to both the traditional max- and sum-inference, as well as hybrid inference tasks such as marginal MAP and MEU.

In particular, we propose a weighted mini-bucket elimination, which generalizes standard mini-bucket, and also connects closely to convex variational methods such as tree-reweighted belief propagation. Our algorithm combines the advantages of both the elimination and variational views: it starts with an elimination-like algorithm, but can iteratively improve the results; it often gives significant improvement within the first few iterations, avoiding the waste of further iterative updates in variational methods.

In addition, based on a novel, general variational representation, we derive a set of efficient "mixed-product" belief propagation algorithms for hybrid inference tasks, including marginal MAP and MEU in decision networks. These algorithms are simple, efficient and significantly advance our ability to solve these challenging problems. We show that our algorithms significantly outperform existing approaches in terms of both empirical performance and theoretical properties.

Outline and Contributions

The general outline of this thesis is depicted in Figure 1.1. Chapters 2-3 present some necessary background on graphical models and inference methods. Then, Chapters 4-7 consist of the novel contributions of the thesis, and Chapter 8 concludes and outlines some open directions. The appendices provide additional proofs and derivations. In more detail:

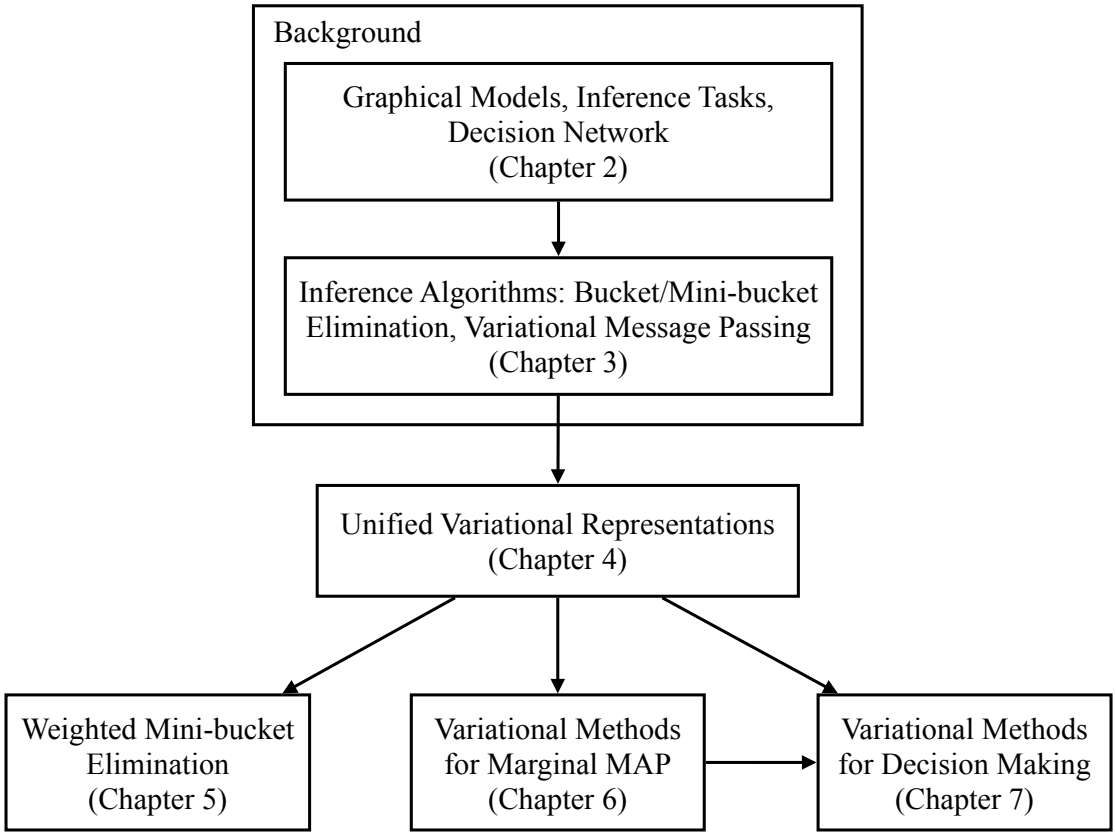


Figure 1.1: Outline and dependence of the thesis.

Chapter 2 provides background on graphical models and inference and decision tasks. We first introduce factorized distributions, including Markov random fields and Bayesian networks, on which we define the standard max- and sum-inference tasks, and the hybrid marginal MAP task. We then introduce influence diagrams, and the related maximum expected utility (MEU) task, which reduces to a hybrid sum-max-sum inference under a particular assumption called perfect recall.

Chapter 3 introduces the two major styles of inference methods: Section 3.1 introduces elimination based methods, including the exact bucket elimination and the approximate mini-bucket elimination algorithms; we also describe the "message passing" viewpoint of variable elimination, presenting exact forward-backward belief propagation (BP) on trees, as well as loopy BP as an approximation heuristic. Section 3.2 introduces various variational inference

methods, all obtained by approximating a basic variational (or dual) form of the log-partition function. We start with the Bethe entropy approximation, which gives a principled derivation for loopy BP; we then introduce several other approximations, including tree reweighted BP, mean field and their variants.

Chapters 4-7 describe the contributions of this thesis.

Chapter 4 develops the dual forms for general "hybrid" exact inference tasks, which extend those known for standard sum-inference and max-inference problems, and provide the main theoretical foundation of subsequent chapters. In particular, we derive dual representations of two unifying viewpoints for hybrid inference: the first, a powered sum inference task that generalizes arbitrary sequences of max, sum, and min operators, and the second, an MEU task in general, cooperative multi-agent systems. Both viewpoints result in exact forms for marginal MAP and MEU with perfect recall. Specific contributions include:

- We define the sequential powered sum, and show that it generalizes many hybrid tasks, including marginal MAP and sum-max-sum inference (MEU with perfect recall).
- We give a variational representation of the sequential powered sum and study its properties, under assumptions of positive and negative weights (or temperatures); these results form the foundation for chapter 5.
- We derive a variational representation for marginal MAP that forms the foundation for Chapter 6.
- We derive a variational representation for general MEU tasks in influence diagrams, including both perfect recall and imperfect recall cases, forming the foundation for Chapter 7.

Chapter 5 introduces a novel weighted mini-bucket elimination algorithm, which generalizes and extends standard mini-bucket, as well as connecting it to the framework of variational inference approximations such as tree-reweighted belief propagation (TRBP). Our

algorithm inherits many of the advantages of, and significantly outperforms, both MBE and TRBP. In detail:

- We propose a novel weighted mini-bucket elimination (MBE) algorithm for approximate inference that generalizes mini-bucket elimination algorithm.
- Our method provides a unified framework for calculating both upper and lower bounds, by using positive and negative weights, respectively. We show that tightening the upper bounds corresponds a convex optimization, while optimizing the lower bounds leads to a challenging non-convex optimization problem.
- We derive iterative tightening methods to find the optimal weight values, as well as the optimal factor reallocation for our weighted MBE bound, with both positive and negative weights. Our algorithm for optimal factor reallocation can also be used to improve standard mini-bucket elimination.
- We draw and discuss the connection between our weighted MBE and convex variational methods, including tree-reweighted belief propagation (TRBP) and conditional entropy decomposition (CED). We show that our weighted MBE bounds represent a large subclass of the TRBP and CED bounds, but with far fewer parameters, enabling efficient, anytime tightening algorithms that provide flexible cost-accuracy trade-offs.
- We experimentally demonstrate the efficiency and flexibility of our weighted MBE method. Compared to standard MBE, our method is much more general and flexible, and provides much better solutions with roughly the same computational complexity. Compared to variational methods such as TRBP and mean field, our method can easily and efficiently use larger clique sizes (controlled by the *ibound*) to provide much better results. In addition, our algorithm often observes significant improvement within just the first one or few iterations, suggesting that running variational message-passing to convergence may be wasteful.

Chapter 6 leverages the variational representation of marginal MAP introduced in Chapter 4, and develops a spectrum of efficient message passing algorithms that both admit appealing theoretical properties and perform well in practice:

- We formalize the concept of A-B trees, which characterize graphs on which marginal MAP problems can be solved efficiently, in the same way that max- and sum-inference are efficient on trees
- We extend both the Bethe and tree reweighted (TRW) entropy approximations to marginal MAP, and derive a novel "mixed-product" belief propagation (BP) that is a hybrid of max-product, sum-product, and special "argmax-product" message updates.
- We derive a class of convergent algorithms for marginal MAP, by maximizing the Bethe or TRW approximation based on the proximal point method; interestingly, this approach takes the form of transforming the marginal MAP problem into a sequence of pure (or annealed) marginalization tasks.
- We provide theoretical conditions under which our mixed-product BP obtains the global optimum; however, these conditions may not be satisfied in practice.
- We provide theoretical conditions under which our mixed-product BP takes on strong local optimality guarantees (optimal up to perturbations on certain embedded subgraphs).
- We extend mixed-product BP to models with higher order cliques, and derive mixed-product factor graph BP and mixed-product junction graph BP.
- We show that expectation-maximization (EM) and variational EM can be treated as applying a mean field-like approximation on our variational form for marginal MAP.
- We present numerical experiments on both simulated and real-world datasets. We show that our methods can provide significantly better solutions than existing baselines, including EM and a state-of-the-art algorithm based on local search methods.

Chapter 7 applies variational approximation techniques to the dual form of maximum expected utility tasks derived in Chapter 4, giving new belief propagation algorithms for optimizing policies in influence diagrams and studying their theoretical and empirical properties. In more detail:

- Applying our novel variational representation to MEU in influence diagrams, we derive a novel MEU-BP algorithm, which works not only in influence diagrams with perfect recall, but also those with imperfect recall.
- We study the theoretical properties of MEU-BP, proving a reparameterizatipon property and a strong local optimality property, which is guaranteed to be stronger than the policy-by-policy optimality given by greedy methods such as the classic single policy update algorithm.
- We derive a convergent version of MEU-BP based on the proximal point method.
- Our methods work by exploiting the factorization structure of the influence diagram; however, the presence of additive utilities may lead to computational difficulties. We propose a method to address this issue, which significantly improves performance with additive utilities.
- We test our methods experimentally on both simulated and real world models, including a distributed detection problem in sensor networks. We show that our algorithms are highly efficient; for example, in influence diagrams with imperfect recall our methods are both far more efficient, and find far better decision policies, than the standard, greedy single-policy update algorithm.

Chapter 8 gives some conclusions and open directions for future research, and the appendices contain several proofs and additional details omitted from the main chapters.

Chapter 2

Graphical Models for Inference and Decision Making

In this chapter, we provide background about inference and decision making in graphical models. Section 2.1 defines various types of graphical models, including Markov random fields, Bayesian networks and factor graphs. Section 2.2 then introduces three major inference tasks on graphical models: maximum a posteriori or MAP (maximization) tasks, marginalization (summation) tasks, and the more general marginal MAP (max-sum) tasks; the importance and the increased difficulty of marginal MAP tasks are emphasized. In Section 2.3, we introduce decision networks (also called influence diagrams), which are generalizations of Bayesian networks used to represent decision-making problems, and the associated maximum expected utility (MEU) task for optimizing decision policies. MEU inference tasks are even more general than marginal MAP; they reduce to an interleaved, sum-max-sum form similar to marginal MAP for decision making in centralized settings under an assumption called perfect recall, but can also be applied to much more general settings, including decentralized "team" decision making with limited information sharing. However, these more general settings also incur additional and fundamental increases in the complexity of the associated inference tasks.

2.1 Probabilistic Graphical Models

Denote by $\boldsymbol{x}$ a random vector $\{x_1, x_2, \ldots, x_n\}$ taking values in space $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. We assume the x_i are discrete variables throughout the thesis. We will use $[n]$ to denote the set of first n positive integers, i.e., $[n] = \{1, \ldots, n\}$. For any subset $\alpha \subseteq [n]$, we write $\boldsymbol{x}_\alpha = \{x_i | i \in \alpha\}$.

It is in general computationally intractable to directly specify or process an arbitrary joint distribution $p(x_1, x_2, \ldots, x_n)$, since it may require $|\mathcal{X}_1| \times \cdots \times |\mathcal{X}_n| - 1 = O(\exp(n))$ parameters, which quickly grows too large in practice. Fortunately, real-world distributions often have special patterns that we can exploit to decrease the computational burden. Probabilistic graphical models are a particular class of distributions that have been well studied in machine learning and statistics. The key idea is to restrict the form of the joint probability to be a product of a set of lower dimensional functions, each of which involves only a small number of variables, that is,

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha), \qquad Z = \sum_{\boldsymbol{x}} \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha), \tag{2.1}$$

where $\mathcal{I}$ is a set of subsets of variables, and $\psi_\alpha : \mathcal{X}_\alpha \to \mathbb{R}^+$ are positive (lower dimensional) functions called *factors*; here we conveniently use the variable scope $\alpha := \text{scope}(\psi)$ to index ψ. The constant Z serves to normalize the distribution, and is known as the partition function. In this case, the number of parameters required is only $O(\exp(\max\{|\alpha| : \alpha \in \mathcal{I}\}))$, which can be much smaller than the $O(\exp(n))$ required for an unstructured distribution. Distributions of form (2.1) are also called Gibbs distributions.

Conditional Independence. In additional to the computational benefits, the importance of the factorization structure (2.1) lies in its connection to conditional dependency. Consider, for example, two variables x_i and x_j that do not co-appear in any $\alpha \in \mathcal{I}$. Denote by $\mathcal{I}_i$ the

set of factors involving i, that is, $\mathcal{I}_i = \{\alpha \in \mathcal{I} \colon i \in \alpha\}$, and similarly for $\mathcal{I}_j$. Then obviously, we have $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$, and therefore

$$p(x_i, x_j | \boldsymbol{x}_{\neg ij}) \propto \prod_{\alpha \in \mathcal{I}_i} \psi_\alpha(\boldsymbol{x}_\alpha) \cdot \prod_{\alpha \in \mathcal{I}_j} \psi_\alpha(\boldsymbol{x}_\alpha),$$

where $\boldsymbol{x}_{\neg ij}$ denotes the variables other than x_i and x_j; the factors that involve neither i nor j are treated as constants and dropped. Similarly, we have

$$p(x_i | \boldsymbol{x}_{\neg ij}) \propto \prod_{\alpha \in \mathcal{I}_i} \psi_\alpha(\boldsymbol{x}_\alpha), \qquad\qquad p(x_j | \boldsymbol{x}_{\neg ij}) \propto \prod_{\alpha \in \mathcal{I}_j} \psi_\alpha(\boldsymbol{x}_\alpha).$$

Comparing the forms of $p(x_i, x_j | \boldsymbol{x}_{\neg ij})$ and $p(x_i | \boldsymbol{x}_{\neg ij})$, $p(x_j | \boldsymbol{x}_{\neg ij})$, we have

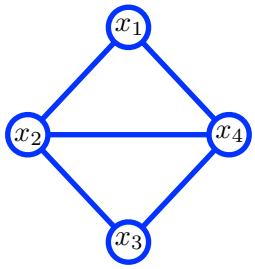
$$p(x_i, x_j | \boldsymbol{x}_{\neg ij}) = p(x_i | \boldsymbol{x}_{\neg ij}) p(x_j | \boldsymbol{x}_{\neg ij}).$$

In other words, x_i and x_j are conditionally independent for fixed $\boldsymbol{x}_{\neg ij}$, that is, $x_i \perp x_j \mid\mid \boldsymbol{x}_{\neg ij}$. Therefore, the factorization assumptions effectively correspond to conditional independence assumptions, which often arise in real world systems.

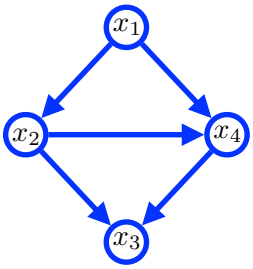
Graph theory provides a convenient notation for visualizing the factorization structure, and also makes it easy to read off all the conditional independence relationships. This leads to different types of *probabilistic graphical models*, depending on the choice of graphical representation.

Markov Random Fields. A Markov random field is a distribution of the form (2.1), whose factorization structure is represented using an undirected graph $G = (V, E)$, called its Markov network, where each node $i \in V$ is associated with a variable x_i, and there exists an edge $(ij) \in E$ if i and j appear in some common subsets in $\mathcal{I}$, that is, $\{i, j\} \subseteq \alpha$ for some $\alpha \in \mathcal{I}$.

In this case, the index set $\mathcal{I}$ forms a subset of the cliques (fully connected subgraphs) of



(a) Markov random field:
$p(\boldsymbol{x}) \propto \psi_{12}\psi_{23}\psi_{24}\psi_{34}\psi_{14}$



(b) Bayesian network:
$p(\boldsymbol{x}) = p(x_1)p(x_2|x_1)p(x_3|x_2,x_4)p(x_4|x_1,x_2)$

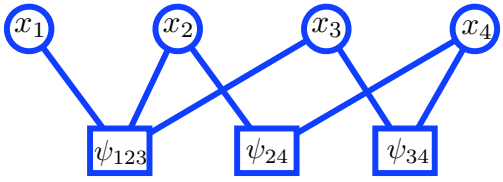
Figure 2.1: Simple examples of Markov random fields and Bayesian networks.

G. With this notation, the conditional independence is associated with the connectivity in the graph: Given any three subsets of variables $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ and $\boldsymbol{x}_S$, we have $\boldsymbol{x}_A \perp \boldsymbol{x}_B || \boldsymbol{x}_S$ if every path from a node in A to a node in B passes through S (the so called *global Markov properties*). In addition, the well-known *Hammersley–Clifford* theorem shows that any distribution with positive probabilities that satisfies the Markov properties with respect to a graph G must have a factorization form of (2.1), whose $\mathcal{I}$ is the set of cliques of G. See Lauritzen [1996] for a comprehensive treatment.

A special class of models that have received considerable attention are pairwise models, which are models that include only singleton and pairwise factors; that is,

$$p(\boldsymbol{x}) \propto \prod_{i \in V} \psi_i(x_i) \prod_{(ij) \in E} \psi_{ij}(x_i, x_j),$$

where each singleton factor ψ_i is associated with the node i, and the pairwise factors correspond to the edges in the graph. See Figure 2.1(a) for a simple example. It is often relatively convenient to specify inference algorithms for pairwise models, and they are well studied to due their relative simplicity. In addition, any graphical model can be reformed into a pairwise model, with some loss of computational efficiency [e.g., Wainwright and Jordan, 2008].



$$p(\boldsymbol{x}) \propto \psi_{123}(x_1, x_2, x_3)\psi_{24}(x_2, x_4)\psi_{34}(x_3, x_4)$$

Figure 2.2: An example factor graph.

Bayesian Networks. Bayesian networks are another common formalism for defining graphical models. Bayesian networks use directed acyclic graphs (DAGs) to represent probabilities that can be written as a product of conditional probabilities; that is,

$$p(\boldsymbol{x}) = \prod_i p(x_i | \boldsymbol{x}_{\mathrm{pa}(i)}),$$

where the parent set $\{\mathrm{pa}(i) \colon i \in [n]\}$ defines a directed acyclic graph $G = (V, E)$: each node $i \in V$ is again associated with a variable x_i, and E is the set of directed edges that points from each parent in $\mathrm{pa}(i)$ to node i, for $\forall i \in V$. Figure 2.1(b) illustrates a simple example of a Bayesian network.

Note that, for Bayesian networks, the distribution is composed of conditional probabilities and hence is normalized (sums to one) by definition. However, in practice it is typical to observe values for some set of "evidence" variables, inducing an un-normalized model over the remaining variables.

Factor Graphs. Another common and useful graphical representation is the factor graph formalism. Given a distribution with factorized form $p(\boldsymbol{x}) \propto \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha)$, its factor graph representation is a bipartite graph $G = (V, \mathcal{I}, E)$ which consists of variable nodes $V = \{i\}$, factor nodes $\mathcal{I} = \{\alpha\}$, and edges between factors and the variables within their scopes, $E = \{(i, \alpha) \in V \times \mathcal{I} \colon i \in \alpha\}$. See Figure 2.2 for an example factor graph.

2.2 Probabilistic Inference Tasks

Given a graphical model $p(\boldsymbol{x})$, the term *inference* refers generically to answering probabilistic queries about the model, such as computing marginal probabilities or maximum *a posteriori* estimates. Although these inference tasks are NP-hard in the worst case, recent algorithmic advances, including the development of variational methods and the family of algorithms collectively called belief propagation, provide approximate or exact solutions for these problems in many practical circumstances. We introduce three common classes of inference tasks, which involve optimization, marginalization and their combination, respectively.

Optimization. *Maximization* or *max-inference* tasks, sometimes called maximum *a posteriori* (MAP) or most probable explanation (MPE) tasks, look for a mode of the joint probability, that is,

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}} \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha),$$

where, since the optimization does not involve the partition function Z, it can be dropped in the second equation. The MAP task is widely used for making predictions in high dimensions. One well known example is image denoising via Markov random fields in computer vision [e.g., Li and Singh, 2009]. In Bayesian network literature, the MAP task corresponds to optimizing the configuration of the unobserved variables given some observed evidence, and is thus often referred to as the most probable explanation (MPE) task. MAP inference also includes classical weighted constraint satisfaction problems (CSPs) as a special case, in which the log of each factor corresponds to a weighted logical constraint.

Marginalization. *Marginalization* or *sum-inference* tasks sum over the configurations to calculate the marginal probabilities of one or a few variables, or to evaluate the model's

normalization constant Z, that is,

$$p(x_i) = \sum_{\boldsymbol{x}_{[n]\setminus\{i\}}} p(\boldsymbol{x}) = \frac{1}{Z} \sum_{\boldsymbol{x}_{[n]\setminus\{i\}}} \prod_{\alpha\in\mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha), \qquad Z = \sum_{\boldsymbol{x}} \prod_{\alpha\in\mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha).$$

Marginal probability distributions are useful in many estimation settings; for example, they can be used to make Bayes-optimal (minimum error rate) predictions on x_i, provide confidence intervals, or detect unusual or anomalous events. In addition, marginalization tasks are required when calculating the data likelihood and its derivative, and thus forms a critical subroutine for learning models from data.

The term *partition function* for Z comes originally from statistical physics. In the Bayesian network setting, un-normalized models ($Z \neq 1$) arise when some of the variables (the "evidence" variables) are assigned observed values. In this case, the normalization constant Z corresponds precisely to the marginal probability of the observed value of the evidence variables. Hence, computing Z is sometimes referred to as the *probability of evidence* task in Bayesian network literature. Additionally, computation of Z includes classical counting constraint satisfaction problems (#CSP) as a special case.

Marginal MAP. Finally, marginal MAP tasks[1] comprise a type of *mixed-inference*, a hybrid of both the marginalization and optimization tasks. Marginal MAP seeks a partial configuration of variables that maximizes their marginal probability, with the remaining variables summed out; that is,

$$\boldsymbol{x}_B^* = \arg\max_{\boldsymbol{x}_B} p(\boldsymbol{x}_B) = \arg\max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} p(\boldsymbol{x}_A, \boldsymbol{x}_B) = \arg\max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} \prod_{\alpha\in\mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha),$$

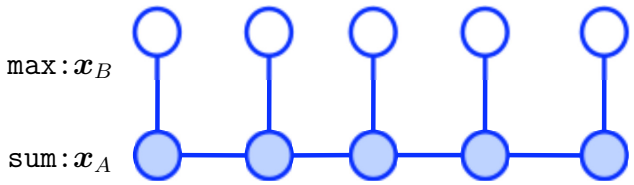
where A, B are disjoint subsets of variables with $A \cup B = [n]$. Obviously, marginal MAP reduces to MAP if $A = \emptyset$ and to marginalization if $B = \emptyset$.

[1] In some Bayesian network literature [e.g., Park and Darwiche, 2004], marginal MAP is simply referred to as MAP, while the joint MAP problem is called MPE.

Marginal MAP plays an essential role in many practical scenarios where there exist hidden variables or uncertain parameters. For example, a marginal MAP problem can arise as a MAP problem on models with unobserved variables whose predictions are not of interest, or as a robust optimization variant of MAP with some unknown or noisily observed parameters marginalized w.r.t. a prior distribution. See Ping et al. [2014], Naradowsky et al. [2012], Pletscher and Ong [2012] for examples of applications that use marginal MAP inference.

Complexity Comparison. These three types of inference tasks are listed in order of increasing difficulty: max-inference is NP-hard (its "decision" version is NP-complete), while sum-inference is #P-complete, and the decision version of mixed inference is NP^{PP}-complete [Park and Darwiche, 2004, de Campos, 2011]. Practically speaking, max-inference tasks have a host of efficient algorithms such as loopy max-product BP, linear programming relaxations, and dual decomposition [see e.g., Koller and Friedman, 2009, Sontag et al., 2011]. Sum-inference is more difficult than max-inference: for example there are models, such as those with binary attractive pairwise potentials, on which sum-inference is #P-complete but max-inference is tractable [Greig et al., 1989, Jerrum and Sinclair, 1993]. In general, the complexity of these tasks can grow exponentially with a quantity called the *induced width*, whose value depends on the structure of the graph. When the graph is highly structured, such as a linear chain or a tree, both max- and sum-inference can be solved efficiently using dynamic programming, in time linear in the graph size; see Chapter 3.

Mixed-inference is significantly harder than either standard max- or sum- inference tasks: for example, marginal MAP can be NP-hard even on tree structured graphs, in which the max- and sum-inference tasks are only linear complexity. An illustrative example is shown in Figure 2.3, where marginal MAP is NP-hard on a simple tree structured graph. The main difficulty arises because the max and sum operators do not commute, which restricts our ability to reorder the calculations in a more computationally efficient manner. Instead, all the summation operations must be completed before *any* max operations; in the worst case,



Marginal MAP:

$$\boldsymbol{x}_B^* = \arg\max_{\boldsymbol{x}_B} p(\boldsymbol{x}_B) = \arg\max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} p(\boldsymbol{x}).$$

Figure 2.3: Illustrating the difficulty of marginal MAP: an example adapted from Koller and Friedman [2009, page 561] in which a marginal MAP query on a tree requires exponential time complexity. The marginalization over $\boldsymbol{x}_A$ destroys the conditional dependency structure in the marginal distribution $p(\boldsymbol{x}_B)$, causing an intractable maximization problem over $\boldsymbol{x}_B$. The exact variable elimination method, which sequentially marginalizes the sum nodes and then maximizes the max nodes, has time complexity $O(\exp(n))$, where n is the chain length.

marginalizing the sum nodes $\boldsymbol{x}_A$ may destroy any conditional independence among the max nodes $\boldsymbol{x}_B$. The lack of conditional independence structure, then, makes it difficult to directly optimize the objective function $\sum_{\boldsymbol{x}_A} p(\boldsymbol{x}_A, \boldsymbol{x}_B)$, even when the sum part alone is tractable (such as when the nodes in A form a tree). For more details, see Section 3.1.

Marginal MAP vs. MAP. Despite of its computational difficulty, marginal MAP plays an essential role in many practical scenarios. The marginal MAP configuration $\boldsymbol{x}_B^*$ is Bayes optimal in the sense that it minimizes the expected error on B, $\mathbb{E}[\mathbf{1}(\boldsymbol{x}_B^* \neq \boldsymbol{x}_B)]$, where $\mathbb{E}[\cdot]$ denotes the expectation under distribution $p(\boldsymbol{x})$. Here, the variables $\boldsymbol{x}_A$ are not included in the error criterion, perhaps because they are "nuisance" hidden variables that are not of direct interest, or are unobserved or inaccurately measured model parameters. In contrast, the joint MAP configuration $\boldsymbol{x}^*$ minimizes the joint error $\mathbb{E}[\mathbf{1}(\boldsymbol{x}^* \neq \boldsymbol{x})]$, but gives no guarantees on the partial error $\mathbb{E}[\mathbf{1}(\boldsymbol{x}_B^* \neq \boldsymbol{x}_B)]$. In practice, perhaps because of the wide availability of efficient algorithms for joint MAP, researchers tend to over-use joint MAP even in cases where marginal MAP would be more appropriate. The following toy example shows that this seemingly reasonable approach can sometimes cause serious problems.

Example 2.1 (Weather Dilemma)**.** *Define* $x_b \in \{\texttt{rainy}, \texttt{sunny}\}$ *to be the weather condition in Irvine, and* $x_a \in \{\texttt{walk}, \texttt{drive}\}$ *whether Alice drives or walks to the school (which depends on the weather). Let the probabilities of* x_b *and* x_a *be*

$p(x_b)$:

`rainy`	0.4
`sunny`	0.6

$p(x_a|x_b)$:

	`walk`	`drive`
`rainy`	1/8	7/8
`sunny`	1/2	1/2

The task is to calculate the most likely weather condition of Irvine, which is obviously `sunny` *according to* $p(x_b)$*. The marginal MAP solution,* $x_b^* = \arg\max_{x_b} p(x_b) =$ `sunny`*, gives the correct answer. However, the full MAP solution,* $[x_a^*, x_b^*] = \arg\max p(x_a, x_b) =$ [`drive`, `rainy`]*, yields the answer* $x_b^* =$ `rainy` *(by dropping the* x_a^* *component), which is obviously wrong. Paradoxically, if* $p(x_a|x_b)$ *is changed (say, to correspond to a different person), the solution returned by the full MAP query could be different.*

In the above example, since no evidence on x_a is observed, the conditional probability $p(x_a|x_b)$ does not provide useful information for x_b, but instead provides misleading information when it is incorporated in the full MAP estimator. The marginal MAP query, on the other hand, eliminates the influence of the irrelevant $p(x_a|x_b)$ by marginalizing (or averaging over) x_a. In general, the marginal MAP and full MAP can differ significantly when the uncertainty in the hidden variables changes as a function of $\boldsymbol{x}_B$.

2.3 Structured Decision Making Under Uncertainty

Intuitively speaking, optimization tasks correspond to finding configurations that maximize a function (for MAP, the joint probability of the model), while marginalization tasks correspond to averaging over random effects. Real world intelligent systems will almost inevitably involve both of these two elementary operations as building blocks of more complicated reasoning. For example, the marginal MAP query is a simple hybrid of optimization and marginalization, in which the function being maximized is a marginal probability (or equiv-

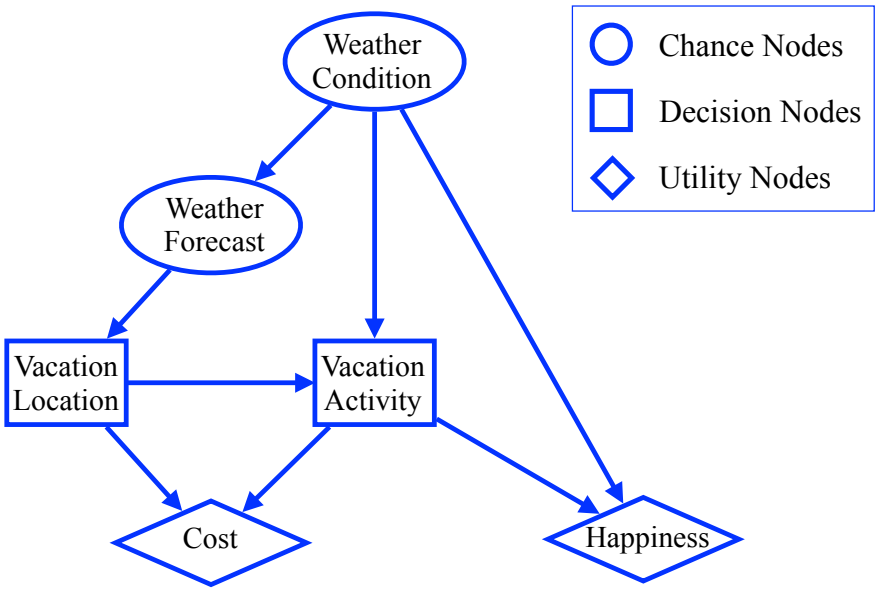


Figure 2.4: A simple influence diagram for deciding vacation activity.

alently, an expectation). More generally, sequential decision making problems may involve several such steps of optimization and expectations, as early actions should optimize the expected reward that will ensue from later actions. Many decision-making tasks can be framed conveniently using decision networks, also known as influence diagrams, which extend the Bayesian network formalism by adding additional decision nodes and reward functions for modeling the decision process.

2.3.1 Influence Diagrams and Maximum Expected Utility

Influence diagrams (IDs) or *decision networks* [Howard and Matheson, 1985] are extensions of Bayesian networks to represent structured decision making problems within uncertain environments. Formally, an influence diagram is defined on a directed acyclic graph $G = (V, E)$, whose nodes $V = R \cup D$ are a union of chance nodes R and decision nodes D, and a utility function $u(\boldsymbol{x})$:

1. Each **chance node** $i \in R$ represents a random variable x_i, associated with a conditional probability table $p_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$, in a way similar to that in Bayesian networks.

2. Each **decision node** $i \in D$ represents a controllable decision variable x_i, whose value is determined by a decision maker via a decision rule (or policy) which determines the value of x_i based on the observation on the values of $\boldsymbol{x}_{\mathrm{pa}(i)}$. In words, the parents $\mathrm{pa}(i)$ denotes the set of information available at the time the decision is made. Therefore, the edges from $\mathrm{pa}(i)$ to i for any decision nodes are called *information arcs*. A decision rule can be represented as a deterministic conditional "probability" $\delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$, where $\delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) = 1$ if x_i is the selected value based on $\boldsymbol{x}_{\mathrm{pa}(i)}$, and zero if otherwise. We call the collection of policies $\boldsymbol{\delta} = \{\delta_i \colon i \in D\}$ a *strategy*.

   It is helpful to allow *soft decision rules* where $\delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$ takes fractional values; these define a *randomized strategy* in which x_i is determined by randomly drawing from conditional probability $\delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$. Denote by Δ^o the set of deterministic strategies and Δ the set of randomized strategies. Note that Δ^o is a discrete set, while Δ is its convex hull.
3. Finally, a **utility function** $u : \mathcal{X} \to \mathbb{R}^+$ measures the reward given an instantiation of $\boldsymbol{x} = [\boldsymbol{x}_R, \boldsymbol{x}_D]$, which the decision maker wants to maximize. It is reasonable to assume some decomposition structure on the utility $u(\boldsymbol{x})$, either additively or multiplicatively:

$$\textit{Additive}: \quad u(\boldsymbol{x}) = \sum_{j \in U} u_j(\boldsymbol{x}_{\beta_j}), \qquad \text{or} \qquad \textit{Multiplicative}: \quad u(\boldsymbol{x}) = \prod_{j \in U} u_j(\boldsymbol{x}_{\beta_j}).$$

   A decomposable utility function can be visualized by augmenting the DAG with a set of leaf nodes U, called *utility nodes*, each with parent set β_j. See Figure 2.4 for a simple example. We should point out that the utility nodes are not uniquely mapped to random or decision variables as are the chance and decision nodes. Instead, they are introduced mainly for the purpose of visualization of the utility dependence, and in this sense are more similar to factor nodes in a factor graph.

Remark. Historically, the definition of IDs requires an additional *perfect recall* assumption that we discuss in Section 2.3.2 below (see e.g., Howard and Matheson [1985], Shachter [1988, 1986]); IDs without perfect recall are typically called *limited memory* IDs, or LIMIDs [Zhang

et al., 1994]. However, throughout this thesis, we will not assume perfect recall for general IDs unless stated explicitly.

Given an influence diagram, we aim to find an optimal strategy that maximizes the *expected utility function* (MEU), that is,

$$\begin{aligned}\mathrm{MEU} &= \max_{\boldsymbol{\delta}\in\Delta} \mathrm{EU}(\boldsymbol{\delta}) = \max_{\boldsymbol{\delta}\in\Delta} \mathbb{E}(u(\boldsymbol{x}) \mid \boldsymbol{\delta}) \\ &= \max_{\boldsymbol{\delta}\in\Delta} \sum_{\boldsymbol{x}} u(\boldsymbol{x}) \prod_{i\in R} p_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \prod_{i\in D} \delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}). \end{aligned} \tag{2.2}$$

where we maximize the expected utility over the set of randomized strategies Δ; this is equivalent to maximizing over the deterministic strategies Δ^o; see Appendix C.1.

Let us write $p(\boldsymbol{x}_R|\boldsymbol{x}_D) = \prod_{i\in R} p(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$ be the (overall) conditional distribution of the random variables conditional on the decision variables. Then the MEU is written as

$$\mathrm{MEU} = \max_{\boldsymbol{\delta}\in\Delta} \sum_{\boldsymbol{x}} p(\boldsymbol{x}_R|\boldsymbol{x}_D) u(\boldsymbol{x}) \prod_{i\in D} \delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}). \tag{2.3}$$

Thus, mathematically, a MEU task is uniquely decided by only two components:

1. The parent sets $\{\mathrm{pa}(i) \colon i \in D\}$ of the decision nodes (that is, the informational arcs).
2. The product $p(\boldsymbol{x}_R|\boldsymbol{x}_D)u(\boldsymbol{x})$, which we call the "utility-augmented" distribution.

Therefore, we can treat MEU as a special sort of "inference" on the utility-augmented distribution $q(\boldsymbol{x}) := p(\boldsymbol{x}_R|\boldsymbol{x}_D)u(\boldsymbol{x})$, which, as we will show in the sequel, includes the more common inference tasks in Section 2.2 as special cases. This general view also allows us to specify a MEU problem without necessarily referring to the semantics of the influence diagram — one can specify $p(\boldsymbol{x}_R|\boldsymbol{x}_D)$ and $u(\boldsymbol{x})$ arbitrarily, e.g., defining $p(\boldsymbol{x}_R|\boldsymbol{x}_D)$ via an undirected MRF, extending the definition of IDs.

2.3.2 Perfect Recall and Sum-Max-Sum Inference

The MEU problem in (2.3) can be solved in closed form if the influence diagram satisfies a *perfect recall assumption* (PRA): that there exists a "temporal" ordering over all the decision nodes, say $\{d_1, d_2, \cdots, d_m\}$, consistent with the partial order defined by the DAG G, such that every decision node observes all the earlier decision nodes and their parents, that is, $\{d_j\} \cup \text{pa}(d_j) \subseteq \text{pa}(d_i)$ for any $j < i$. Intuitively, PRA implies a centralized decision scenario, where a global decision maker sets all the decision nodes in a predefined order, with perfect memory of all the past observations and decisions.

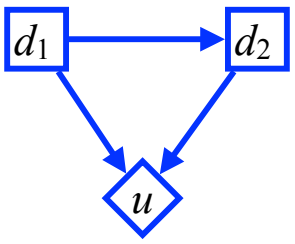
With PRA, the chance nodes can be grouped by when they are observed. Let r_{i-1} ($i = 1, \ldots, m$) be the set of chance nodes that are parents of d_i but not of any d_j for $j < i$; then both the decision and chance nodes are temporally ordered by $[r_0, d_1, r_1, \cdots, d_m, r_m]$. For IDs with PRA, the MEU task can be solved by dynamic programming, which in this case takes on a sum-max-sum form,

$$\text{MEU} = \sum_{x_{r_0}} \max_{x_{d_1}} \sum_{x_{r_1}} \cdots \max_{x_{d_m}} \sum_{x_{r_m}} p(\boldsymbol{x}_R|\boldsymbol{x}_D) u(\boldsymbol{x}), \tag{2.4}$$

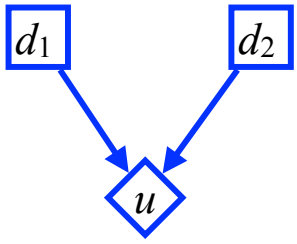
and the optimal strategies are decoded by

$$\delta^*_{d_i}(x_{d_i}|\boldsymbol{x}_{\text{pa}(d_i)}) \propto \mathbf{1}\big[x_{d_i} \in \arg\max_{x_{d_i}} \big\{ \sum_{x_{r_i}} \cdots \max_{x_{d_m}} \sum_{x_{r_m}} p(\boldsymbol{x}_R|\boldsymbol{x}_D) u(\boldsymbol{x}) \big\}\big],$$

where $\mathbf{1}[\cdot]$ is the indicator function. Here the calculation is performed in the reverse temporal ordering, alternating between marginalizing the random variables and maximizing the decision variables. Obviously, (2.4) generalizes the marginalization, optimization, and marginal MAP tasks introduced in Section 2.2, in that it interleaves the sum and max operators in arbitrary orders. The marginal MAP, in particular, can be treated as a *blind decision problem*, where no chance nodes are observed by any decision nodes. Similar to marginal MAP,



(a) Perfect Recall



(b) Imperfect Recall

d_1	d_2	$u(d_1, d_2)$
1	1	3
0	1	1
1	0	1
0	0	2

(c) Utility Function

Figure 2.5: Illustrating perfect recall vs. imperfect recall. In (a) d_2 observes d_1; its optimal decision rule is to follow d_1's state (whatever it is); knowing d_2 will follow, d_1 can choose $d_1 = 1$ to achieve the global optimum. In (b) d_1 and d_2 do not know the other's states; both $d_1 = d_2 = 1$ and $d_1 = d_2 = 0$ (suboptimal) become locally optimal strategies and the optimization problem is multi-modal.

the non-exchangeability of the sum and max operators restricts our ability to reorganize the calculation of (2.4) in a computationally efficient manner (specifically, restricting to elimination orders that are consistent with the "temporal ordering" of the influence diagram; see Section 3.1). This causes the MEU task to have potentially much higher complexity than, for example, sum-inference on a similarly structured model.

However, perfect recall is often not a realistic assumption to make in practice. First, most systems lack enough memory to express arbitrary policies over an entire history of observations: representing an arbitrary policy $\delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$ requires complexity exponential in $|\mathrm{pa}(i)|$, which under PRA is the entire past history of observed random variables and quickly becomes intractable in practice. Second, many real-world scenarios such as team decision making [Detwarasiti and Shachter, 2005] and decentralized sensor networks [Kreidl and Willsky, 2006] are distributed by nature: a team of agents makes decisions separately based on sharing limited information with their neighbors. In these cases, relaxing the perfect recall assumption is very important.

2.3.3 Imperfect Recall and Single Policy Update

General IDs in which the perfect recall assumption has been relaxed have been discussed in, for example, Zhang et al. [1994], Lauritzen and Nilsson [2001], and Koller and Milch [2003], and are commonly referred to as limited memory influence diagrams (LIMIDs). Unfortunately, while relaxing PRA may be necessary from one standpoint, it also causes many additional computational difficulties. First, it is no longer possible to eliminate the decision nodes in a sequential "sum-max-sum" fashion as that in (2.4). Instead, the different decision nodes and their policies can influence each other in a cyclic fashion, formally discussed in Koller and Milch [2003] by defining a relevance graph over the decision nodes; the relevance graph is a tree for IDs with perfect recall, but is usually loopy with imperfect recall. As a consequence, iterative algorithms are usually required to solve LIMIDs. Even more unfortunately, when these iterative updates are performed, the agents' incomplete information often causes them to behave myopically. Ignorance of the system's global statistics leads them to select only locally optimal strategies, and it can be very difficult to make improvements that require coordinated changes in the policies of several agents. This results in the strategy space effectively breaking into many local modes; in fact, this difficulty can be directly viewed as arising from a form of non-convexity in the objective, compared to PRA problems in which it is convex; see Chapter 4, Section 4.3 for more details. Figure 2.5 shows an illustrative example of this phenomenon, in which a graph with imperfect recall has two different local optima, while adding an additional information arc to ensure perfect recall ensures a unique optimum.

The currently most popular algorithms for LIMIDs are based on iterative policy-by-policy improvement, for example, the *single policy update* (SPU) algorithm [Lauritzen and Nilsson, 2001] shown in Algorithm 2.1 steps through all the decision nodes i in some order, and sequentially optimizes δ_i with the other decisions $\boldsymbol{\delta}_{\neg i} = \{\delta_j \colon j \neq i\}$ fixed. If the perfect recall assumption holds, and the updates are made along a reverse temporal order, then one

Algorithm 2.1 Single Policy Update [Lauritzen and Nilsson, 2001] for IDs With Imperfect Recall (LIMIDs)

Input: An influence diagram, as specified by conditional probability $p(\boldsymbol{x}_R|\boldsymbol{x}_D)$ and the utility function $u(\boldsymbol{x})$.
Output: An optimal decision policy $\boldsymbol{\delta} = \{\delta_i \colon i \in D\}$.
Initialize decision policy $\boldsymbol{\delta} = \{\delta_i \colon i \in D\}$ (e.g., randomly).
while not converged **do**
  for all decision nodes $i \in D$ **do**

$$\delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \leftarrow \mathbf{1}\big[x_i \in \arg\max_{x_i} \mathbb{E}(u(\boldsymbol{x})|\boldsymbol{x}_{\{i\}\cup\mathrm{pa}(i)}\,;\,\boldsymbol{\delta}_{\neg i})\big],$$
$$\text{where } \mathbb{E}(u(\boldsymbol{x})|\boldsymbol{x}_{\{i\}\cup\mathrm{pa}(i)};\boldsymbol{\delta}_{\neg i}) = \sum_{x_{V\setminus\{i\}\cup\mathrm{pa}(i)}} p(\boldsymbol{x}_R|\boldsymbol{x}_D)u(\boldsymbol{x}) \prod_{j\in D\setminus\{i\}} \delta_j(x_j|\boldsymbol{x}_{\mathrm{pa}(j)}), \tag{2.5}$$

  end for
end while
Return: Decision policy $\boldsymbol{\delta} = \{\delta_i \colon i \in D\}$.

can show that the SPU reduces to the sum-max-sum rule and returns a globally optimal strategy. However, in the case of imperfect recall, SPU is only guaranteed to return locally optimal strategies, and is often heavily influenced by initialization. In the example shown in Fig. 2.5(b), one can readily see that SPU will converge to either of the two local optima, depending on the initialization.

The issue of local optima can be improved by generalizing SPU to the strategy improvement (SI) algorithm [Detwarasiti and Shachter, 2005], which simultaneously updates subgroups of decisions nodes. However, the time and space complexity of SI grows exponentially with the sizes of the subgroups. We show in Chapter 7 that the novel message-passing algorithms we develop for MEU are able to go beyond this naïve greedy paradigm, giving solutions that are locally optimal in a stronger sense than SPU with comparable or better efficiency.

Chapter 3

Background: Inference Methods

In this thesis we consider two different perspectives, corresponding to two major styles of algorithms, for exact or approximate inference: *variable elimination* methods, which directly eliminate (max or sum over) the variables one by one, and *variational optimization* methods, which are based on the idea of reframing the inference problem as a functional optimization problem that minimizes a KL divergence to the target distribution. Under either perspective, exact inference usually leads to algorithms with complexity that is exponential on the induced-width of the graphical models. However, this is often prohibitively slow for practical applications. Thus, many approximate algorithms have been developed by approximating the exact inference process under one of these two perspectives. For example, the mini-bucket elimination algorithm [Dechter and Rish, 2003] works by directly approximating the marginalization operators in variable elimination; other related methods include Boyen and Koller [1998], Wexler and Meek [2009] and Mauá and de Campos [2012]. Alternatively, the family of variational approximation methods include loopy belief propagation, tree-reweighted BP and mean field methods, each of which rely on approximating the KL divergence minimization problem in various ways.

In this chapter, we introduce general background material about these two types of algorithms: Section 3.1 introduces variable elimination based methods, including the exact bucket elimination and the approximate mini-bucket elimination algorithms; in addition,

these sections define notation for induced graphs and junction graphs. Section 3.2 introduces the basic variational form for the log-partition function, which we then use as a basis to discuss several variational approximation methods, including loopy BP, tree reweighted BP, mean field and their variants.

3.1 Primal View – Elimination Methods

Elimination-based methods such as bucket or variable elimination [Dechter, 1996, 1999, Zhang and Poole, 1996] act by directly eliminating (summing over or maximizing over) the variables, one by one, along a predefined ordering. We first introduce the bucket elimination method [Dechter, 1999] for exact inference in Section 3.1.1, and then the mini-bucket elimination method [Dechter and Rish, 2003] for approximate inference in Section 3.1.2. Finally, in Section 3.1.3, we reform the bucket elimination as forward-backward belief propagation algorithms that pass "messages" over a tree structure, and introduce loopy BP as another type of approximation method, by passing messages over general graphs with loops. We mainly focus the discussion on marginal inference, but the methods' extension to joint optimization and mixed inference can be derived in a straightforward manner, by simply replacing the sum operators with max.

3.1.1 Bucket Elimination for Exact Inference

We use a simple running example to illustrate the general idea of bucket (or variable) elimination. Consider a simple distribution on $\boldsymbol{x} = [x_1, x_2, x_3, x_4]$,

$$p([x_1, x_2, x_3, x_4]) = \frac{1}{Z}\psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3)\psi_{24}(x_2, x_4)\psi_{34}(x_3, x_4).$$

Assume we are interested in sum-inference, specifically calculating the partition function,

$$Z = \sum_{x_4}\sum_{x_3}\sum_{x_2}\sum_{x_1} \psi_{34}(x_3, x_4)\psi_{24}(x_2, x_4)\psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3),$$

for which the brute-force algorithm would be required to sum over a 4 dimensional tensor, with a complexity of $O(d^4)$, where d is the number of possible states of x_i. However, by using the distributive law, the partition function can be written as

$$Z = \sum_{x_4}\sum_{x_3} \psi_{34}(x_3, x_4) \sum_{x_2} \psi_{24}(x_2, x_4) \sum_{x_1} \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3),$$

where the summations can be calculated node by node,

$$\begin{aligned}
\psi_1^{new}(x_2, x_3) &= \sum_{x_1} \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3), \\
\psi_2^{new}(x_3, x_4) &= \sum_{x_2} \psi_{24}(x_2, x_4)\psi_1^{new}(x_2, x_3), \\
\psi_3^{new}(x_4) &= \sum_{x_3} \psi_{34}(x_3, x_4)\psi_2^{new}(x_3, x_4), \\
Z &= \sum_{x_4} \psi_3^{new}(x_4),
\end{aligned} \tag{3.1}$$

which requires a total complexity of only $O(d^3)$, instead of $O(d^4)$.

In general, the calculation can be conveniently organized as a "bucket elimination procedure" (shown in Algorithm 3.1) that eliminates the variables node by node along a pre-defined elimination order. At the ith step of Algorithm 3.1, we multiply all the factors that include x_i (called the "bucket" $\mathbf{B}_i$), and eliminate x_i from the product (step (3.2)), that is,

$$\psi_i^{new}(x_{\pi_i}) = \sum_{x_i} \prod_{\psi \in \mathbf{B}_i} \psi(\boldsymbol{x}_{\text{scope}(\psi)}).$$

This results a new factor $\psi_i^{new}(x_{\pi_i})$ which replaces the factors in $\mathbf{B}_i$ that have been eliminated

away; the variable scope of ψ_i^{new} is $\pi_i := \text{scope}(\psi_i^{new}) = \cup_{\psi \in \mathbf{B}_i} \text{scope}(\psi) \setminus \{i\}$, and can be thought of as the "parent set"[1] (meaning the neighboring, un-eliminated nodes) of i. This elimination step is performed sequentially for all the nodes, so the newly created factor $\psi_i^{new}(x_{\pi_i})$ should be placed into the "bucket" of the first variable to be eliminated in π_i (i.e., the "first parent" of i). In this sense, ψ_i^{new} can also be treated as a "message" that is passed from node i to its first parent, which can motivate additional "message passing" variants of the algorithm such as we explain in Section 3.1.3.

Obviously, the general computational complexity of Algorithm 3.1 is dominated by the elimination step (3.2), which requires $O(nd^{\omega_o+1})$ with $\omega_o = \max\{|\pi_i| \colon i \in [n]\}$. Here ω_o is called the *induced width* (or *tree width*) of $p(\boldsymbol{x})$ along the elimination order $o = [x_1, \ldots, x_n]$. We discuss the induced width further in the sequel, but for now note that the induced width can be computed relatively efficiently for a given ordering by "simulating" Algorithm 3.1 — going through the procedure, but without actually running the elimination step (3.2). Different elimination orders o may lead to different induced widths ω_o; the minimum induced width $\omega = \min_o\{\omega_o\}$ among all possible orders is called the induced width of $p(\boldsymbol{x})$, and captures the best possible complexity of the variable elimination method. Unfortunately, it is in general an NP-complete task to find the best variable ordering. In practice, a number of greedy algorithms exist, including the min-fill and min-induced-width heuristic methods (see Dechter [2013, Section 3.4.2] for an introduction).

Backward Elimination for Computing Marginal Probabilities

Algorithm 3.1 only computes the partition function Z. If we are interested in calculating the marginal probabilities, say $p(x_i)$, a naïve method is to simply fix x_i and run Algorithm 3.1 for all the possible values of x_i. However, this approach is very inefficient if we want to calculate

[1]Note that the term "parent set" π_i in variable elimination should be distinguished from the parent set $\text{pa}(i)$ in the definition of Bayesian networks (Chapter 2).

Algorithm 3.1 Bucket (or variable) elimination for computing the partition function Z [Dechter, 1999].

Input: Factors of a graphical model $\mathbf{F} = \{\psi_\alpha(\boldsymbol{x}_\alpha)\}$, an elimination order $o = [x_1, x_2, \cdots, x_n]$.
Output: The partition function Z.
for $i \leftarrow 1$ to n **do**

1. Find the set (or "bucket") of factors involving variable x_i:

$$\mathbf{B}_i \leftarrow \{\psi \colon \psi \in \mathbf{F}, i \in \text{scope}(\psi)\}.$$

2. Eliminate variable x_i,

$$\psi_i^{new}(\boldsymbol{x}_{\pi_i}) = \sum_{x_i} \prod_{\psi \in \mathbf{B}_i} \psi(\boldsymbol{x}_{\text{scope}(\psi)}), \tag{3.2}$$

where π_i is the variable scope of ψ_i^{new}, that is, $\pi_i := \text{scope}(\psi_i^{new}) = \cup_{\psi \in \mathbf{B}_i} \text{scope}(\psi) \setminus \{i\}$.

3. Update the factor list: $\mathbf{F} \leftarrow (\mathbf{F} \setminus \mathbf{B}_i) \cup \{\psi_i^{new}\}$.

end for
Return: the partition function, $Z = \prod_{\psi \in \mathbf{F}} \psi$.

Note: after all eliminations, any remaining factors ψ are constants, i.e., $\text{scope}(\psi) = \emptyset$.

all the marginals $p(x_i)$ for all i simultaneously, since these computations would repeat many of the same calculations. We can instead derive much more efficient algorithms by sharing these repeated calculations. We do so by performing an additional backward elimination pass (compared to the forward elimination pass described in Algorithm 3.1) that constructs the marginal probabilities recursively using the chain rule of probability. Algorithm 3.2 shows an example of backward elimination that calculates the marginals $\{p(\boldsymbol{x}_{\pi_i \cup \{i\}}) \colon i \in [n]\}$ simultaneously, with a total computational complexity of only $O(n \exp(\omega_o + 1))$.

Triangulation and the Induced Graph

Although the complexity bound ω can be evaluated by "simulating" Algorithm 3.1 beforehand, it would be nice if there is a principled way to directly calculate it from the Markov graph G. In this section, we interpret the variable elimination process as a sequence of graph-

Algorithm 3.2 Bucket elimination with backward pass [Dechter, 1999]

Input: The set of factors of a graphical model $\mathbf{F} = \{\psi_\alpha(\boldsymbol{x}_\alpha)\}$, an elimination order $o = [x_1, x_2, \cdots, x_n]$.

Output: The partition function Z and the marginal distributions $\{p(\boldsymbol{x}_{\pi_i \cup \{i\}}) \colon i \in [n]\}$.

Forward Pass: Run Algorithm 3.1 to calculate Z, and get the buckets $\{\mathbf{B}_i\}$ and $\{\psi_i^{new} \colon i \in V\}$.

Backward Pass (*to compute the marginals*):

Initialization: $p(x_n) = \prod_{\psi \in \mathbf{B}_i} \psi(x_{\mathrm{scope}(\psi)})$

for $i \leftarrow n-1$ to 1 **do**

$$p(x_i|\boldsymbol{x}_{\pi_i}) = \frac{\prod_{\psi \in \mathbf{B}_i} \psi(\boldsymbol{x}_{\mathrm{scope}(\psi)})}{\psi_i^{new}(\boldsymbol{x}_{\pi_i})} \qquad p(\boldsymbol{x}_{c_i}) = \sum_{\boldsymbol{x}_{c_j \setminus c_i}} p(x_i|\boldsymbol{x}_{\pi_i}) p(\boldsymbol{x}_{c_j}), \tag{3.3}$$

where $c_i = \pi_i \cup \{i\}$ and j is the variable whose bucket $\mathbf{B}_j$ receives the factor ψ_i^{new} in Forward elimination.

end for

Return partition function Z and marginals $\{p(\boldsymbol{x}_{\pi_i \cup \{i\}}) \colon i \in [n]\}$.

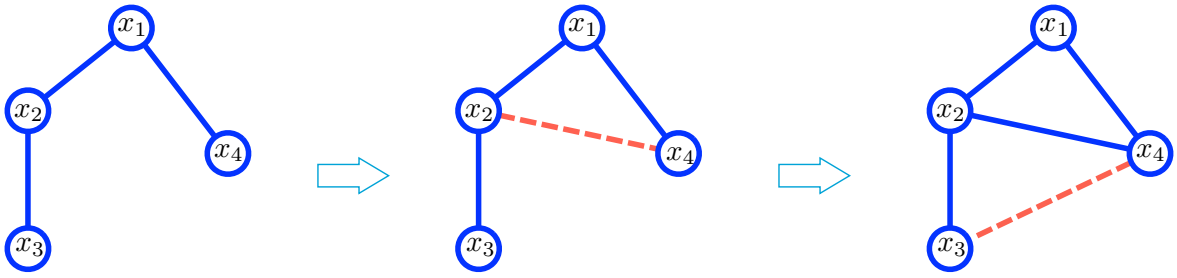
Note: marginals on smaller variable sets can be obtained by further marginalization, e.g., $p(x_i) = \sum_{x_{\pi_i}} p(\boldsymbol{x}_{c_i})$.

ical operations on the graph G; this results the notation of triangulation and of the induced graph, which play an important role in analyzing and deriving many inference algorithms (not limited to variable elimination).

It is useful to begin with some definitions.

Definition 3.1 (**Ordered Graph**). *An ordered graph* (G, o) *is a undirected graph* $G := (V, E)$ *equipped with a total ordering* o *over its node set* V*. For any node* $i \in V$*, the parent set* $\mathrm{pa}_G(i)$ *of node* i *is its connected, subsequent nodes, that is,* $\mathrm{pa}_G(i) = \{i' \in V \colon (i, i') \in E, i \prec_o i'\}$*, where* $i' \prec_o i$ *means that* i' *is ranked later than* i *in the ordering* o*. The width of a node is the size of its parent set, and the width of* (G, o) *is the maximal width of its nodes, i.e.,* $\max\{|\mathrm{pa}_G(i)| \colon i \in V\}$.

Definition 3.2 (**Induced Graph**). *Given an ordered graph* (G, o)*, and a node* $i \in V$*, triangulating node* i *means adding edges between all the parents of* i*. The induced (or trian-*



(a) Original graph

(b) Edges added between the parents of x_1

(c) Edges added between the parents of x_2

Figure 3.1: Triangulating the graph in (a) along ordering $[x_1, x_2, x_3, x_4]$; the induced width along this order is 2, but the induced width of the graph itself is 1, because it is a tree.

gulated) graph $(\tilde{G}, o)$ is obtained by triangulating each of the nodes in turn along the order o. The induced width of (G, o) is the width of its induced graph $(\tilde{G}, o)$. The induced width of the undirected graph G is the minimum induced width of all the ordered graphs generated by G. See Figure 3.1 for an illustrative example.

There is a clear and intuitive meaning for the induced graph: it is simply the Markov network of all the factors involved or generated during the variable elimination process. In particular, the triangulation step (connecting the parents) corresponds to the creation of the new factors ψ_i^{new} during the elimination step (3.2), and one can show that the variable scope π_i of ψ_i^{new} equals the parent set of node i in the resulting induced graph. Therefore, the induced width equals the maximum number of variables involved in the elimination operation (3.2), and hence characterizes the computational complexity.

The induced width of a tree is always one when eliminated from leaves to root; such orders are called tree orders, and defined as follows.

Definition 3.3 (Tree Order). *Given a tree $G = (V, E)$, a total ordering o on V is called a tree order if the induced width of (G, o) is one. In other words, each node $i \in V$ should have at most one parent along a tree order o. Nodes that have no parents are called roots; nodes that are not parents of any other nodes are called leaves.*

Therefore, the complexity of variable elimination along tree orders are $O(nd^2)$, which is very efficient.

Bucket Elimination for MAP and Marginal MAP

Although we have so far restricted our attention to sum-inference, the corresponding algorithms for max-inference (MAP) and mixed-inference (e.g., marginal MAP) can be derived in a straightforward way, by simply replacing the sum operators with max in the elimination steps whenever it is necessary. We omit the precise details of these different algorithm versions, and refer the readers to Dechter [1999, 2013] for more details.

Following similar procedures for analysis, one can show that the best possible complexity of variable elimination for MAP (along the best elimination order) is also $O(n \exp(\omega+1))$, again governed by the induced width $\omega := \min_o\{\omega_o\}$ of G. However, marginal MAP, which consists of a hybrid of both sum and max operators, can require significantly higher complexity due to the restriction on the choice of elimination order. To see this, note that the max and sum operators do not commute, that is,

$$\max_{x_2} \sum_{x_1} f(x_1, x_2) \neq \sum_{x_1} \max_{x_2} f(x_1, x_2).$$

Therefore, when solving the marginal MAP task, $\max_{x_B} \sum_{x_A} p(x_A, x_B)$, the elimination order must be chosen such that all the sum variables (x_A) are eliminated before any of the max variables x_B.

Definition 3.4. *For any node subsets $A \subset V$, $B \subset V$ with $A \cap B = \emptyset$, the A-B constrained induced width ω_{AB} of G is the minimum induced width of (G, o) along orders where x_A are eliminated before x_B, that is,*

$$\omega_{AB} = \min\{\omega_o \colon i \prec_o j, \textit{for } \forall i \in A, j \in B\}.$$

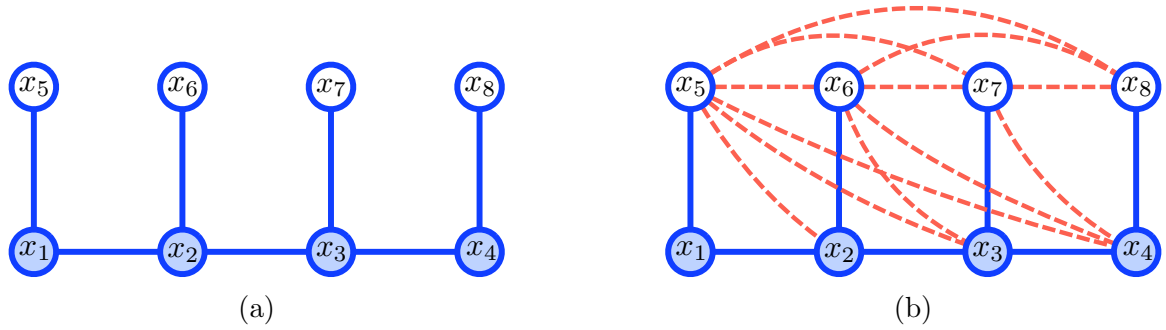


Figure 3.2: (a) A tree graph on which we define a marginal MAP problem $\max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} p(\boldsymbol{x})$ with $A = [1, 2, 3, 4]$ and $B = [5, 6, 7, 8]$; due to the non-exchangeability of max and sum, any valid elimination orders should eliminate A earlier than B. (b) The induced graph obtained using eliminating order $[x_1, \ldots, x_8]$. The A-B constrained induced width is $\omega_{AB} = 4$, while the unconstrained induced width is only $\omega = 1$.

Therefore, the best possible computational complexity of variable elimination for marginal MAP is at least $O(n \exp(\omega_{AB} + 1))$. Unfortunately, the constrained induced width ω_{AB} can be significantly larger than the regular induced width ω, depending on the choice of A and B. For example, Figure 3.2 shows an example of simple tree of size n with ω_{AB} as high as $n/2$ (while the induced width ω is always 1). In general, marginal MAP can be NP-hard even on trees, for which both pure sum- and max- inference can always be efficiently solved (linear complexity in n). Obviously, more general queries such as the sum-max-sum form of MEU (with perfect recall) may impose even more ordering restrictions, and cause even higher computational complexity. Therefore, developing efficient approximations for these hybrid tasks is of particular importance.

3.1.2 Mini-Bucket Elimination for Approximate Inference

The computational complexity of exact bucket elimination is exponential on the induced width, which is often too large in practice. Mini-bucket elimination (MBE) [Dechter and Rish, 2003] is an approximate version of bucket elimination that provides a flexible trade-off between complexity and accuracy. As with bucket elimination, we illustrate the general idea

of MBE using a toy example. Consider the summation task over x_1 in (3.1),

$$\psi_{23}^{new}(x_2, x_3) = \sum_{x_1} \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3). \tag{3.4}$$

This calculation requires summing over x_1 for each pair of values (x_2, x_3), and thus involves a total computational cost of $O(d^3)$, where d is the number of states of x_i. For the general case in (3.2), the exact calculation involves summing over x_i for all the values of $\boldsymbol{x}_{\pi_i}$, requiring $O(d^{|\pi_i|+1})$.

To decrease the complexity, Dechter and Rish [2003] developed an approximation that partitions the factors (or buckets) into several smaller groups (called "mini-buckets"), and eliminates over them separately. For the example in (3.4), Dechter and Rish [2003] provides the following upper and lower bounds as approximations:

$$\sum_{x_1} \psi_{12}(x_1, x_2) \cdot \max_{x_1} \psi_{13}(x_1, x_3) \;\geq\; \sum_{x_1} \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3), \tag{3.5}$$

$$\sum_{x_1} \psi_{12}(x_1, x_2) \cdot \min_{x_1} \psi_{13}(x_1, x_3) \;\leq\; \sum_{x_1} \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3). \tag{3.6}$$

Here, we eliminate x_1 in ψ_{12} and ψ_{13} separately, instead of within their product $\psi_{12}\psi_{13}$ jointly; since the separate elimination operates over smaller functions, it requires a complexity of only $O(2d^2)$, instead of $O(d^3)$.

More generally, we can apply a similar approximation within each elimination step (3.2), and this results a general mini-bucket elimination (MBE) algorithm given in Algorithm 3.3. MBE works in a style similar to exact bucket elimination, except that in each step the buckets are first split into smaller "mini-buckets", which are then eliminated separately before being passed into their own subsequent buckets.

Usually, the amount of splitting in MBE is controlled by a parameter called the *ibound*,

Algorithm 3.3 Mini-bucket elimination

Input: The factors of the graphical model $\mathbf{F} = \{\psi_\alpha(\boldsymbol{x}_\alpha)\}$, an elimination order $o = [x_1, \cdots, x_n]$, and an *ibound*.

Output: An upper (or lower) bound on the partition function Z.

for $i \leftarrow 1$ to n **do**

1. Find the set ("bucket") of factors involving variable x_i:

$$\mathbf{B}_i \leftarrow \{\psi \colon \psi \in \mathbf{F}, i \in \text{scope}(\psi)\}.$$

2. Partition $\mathbf{B}_i$ into R_i subgroups $\{\mathbf{B}_{ir}\}$, such that $\cup_{r=1}^{R_i} \mathbf{B}_{ir} = \mathbf{B}_i$ and

$$|\cup_{\psi \in \mathbf{B}_{ir}} \text{scope}(\psi)| \leq \mathit{ibound} + 1 \quad \text{for all} \quad r = 1, \ldots, R_i.$$

3. Eliminate variable x_i:

for $r \leftarrow 1 \cdots R_i$ **do**

$$\psi_{ir}^{new}(\boldsymbol{x}_{\pi_{ir}}) = \begin{cases} \sum\limits_{x_i} \prod\limits_{\psi \in \mathbf{B}_{ir}} \psi(\boldsymbol{x}_{\text{scope}(\psi)}), & \text{if } r = 1, \\ \max\limits_{x_i} \prod\limits_{\psi \in \mathbf{B}_{ir}} \psi(\boldsymbol{x}_{\text{scope}(\psi)}), & \text{if } r \neq 1, \end{cases} \tag{3.7}$$

where the variable scope of the result is $\pi_{ir} = \cup_{\psi \in \mathbf{B}_{ir}} \text{scope}(\psi) \setminus \{i\}$.

end for

4. $\mathbf{F} \leftarrow (\mathbf{F} \setminus \mathbf{B}_i) \cup \{\psi_{ir}^{new} \colon i = 1, \cdots, R_i\}$.

end for

The partition function bound is $\hat{Z} = \prod_{\psi \in \mathbf{F}} \psi$.

Note: after all eliminations, any remaining factors ψ are constants, i.e., $\text{scope}(\psi) = \emptyset$.

Note: replace the max operator with min in (3.7) *to obtain a lower bound.*

which represents the maximum number of variables allowed in the result of each mini-bucket. Clearly, by controlling the size of the factor scopes, the complexity of MBE is reduced to $O(\exp(\mathit{ibound}))$ instead of $O(\exp(\mathit{inducedwidth}))$ for exact bucket elimination. Note that the *ibound* should be specified by the user to trade-off the complexity and accuracy: smaller *ibound*s result in less computational cost, but are typically less accurate; higher *ibound*s give more accurate results, but are more expensive to compute. If the *ibound* is greater than the induced width, no bucket splitting is required, and the algorithm reduces to exact bucket elimination.

Mini-bucket is simple and fast. Unfortunately, however, the bounds defined in (3.6)-(3.5) are relatively loose, and as a result we need to use a fairly high *ibound* (in practice, often 10 – 20) to achieve good performance. In Chapter 5, we will introduce a more general bound based on Hölder's inequality, and propose a more efficient weighted mini-bucket algorithm that can obtain tighter bounds while using much smaller *ibound*s.

Remark. Note that Algorithm 3.3 only returns an approximation to the partition function. It is also possible to develop a backward elimination similar to Algorithm 3.2 to obtain approximations of the marginal probabilities [see Mateescu et al., 2010], but with a less obvious interpretation as to how the approximation is defined. In Chapter 5, we develop a more general, "weighted" mini-bucket approach, and discuss its resulting version of backward elimination in more detail.

Factor Reallocation Between Mini-buckets

The mini-bucket elimination algorithm in Algorithm 3.3 assumes that each factor ψ_α is assigned to exactly one of the mini-buckets. However, one could instead replace the factor ψ_α with two factors whose product is ψ_α, e.g., $\psi_\alpha = \psi_\alpha^{1/2} \cdot \psi_\alpha^{1/2}$, and assign these two factors into two different mini-buckets. More generally, we can define a slightly more general version of MBE that incorporates this freedom: we first split each original factor ψ_α into the product of multiple factors $\{\psi_{\alpha^s}\}$, which together satisfy $\prod_s \psi_{\alpha^s} = \psi_\alpha$, and then assign the various ψ_{α^s} into different mini-buckets. Note that this is effectively applying MBE on a *reparameterization* $p(\boldsymbol{x}) \propto \prod_\alpha \prod_s \psi_{\alpha^s}$ that is equivalent to the original distribution $p(\boldsymbol{x}) \propto \prod_\alpha \psi_\alpha$.

Another equivalent, but practically useful view is to think of the reparameterization as "reallocating" (or "shifting") the factors between mini-buckets while keeping the overall

product of the factors unchanged. As an example, the bound in (3.5) can be extended to

$$\big[\sum_{x_1} \psi_{12}(x_1, x_2)\varphi(x_1)\big] \cdot \big[\max_{x_1} \psi_{13}(x_1, x_3)\varphi(x_1)^{-1}\big] \;\geq\; \sum_{x_1} \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3), \qquad (3.8)$$

where φ is an arbitrary factor, which provides an additional degree of freedom within the algorithm that can be tuned, or adjusted, to give a tighter bound. More generally, for any two mini-buckets $\mathbf{B}_{ir_1}$ and $\mathbf{B}_{ir_2}$ that shares a variable clique α (that is, $\text{scope}(\mathbf{B}_{ir_1}) \cup \text{scope}(\mathbf{B}_{ir_2}) = \alpha$, we have the freedom to "reallocate" (or "shift") a factor $\varphi(\boldsymbol{x}_\alpha)$ between them, in order to improve the bound.

Roughly speaking, it turns out that an optimal "reallocation" φ should be chosen in such a way that the two mini-buckets' functions are properly matched – often called a *moment matching* condition – in a sense that is made concrete in Ihler et al. [2012] and Liu and Ihler [2011], and is discussed in depth in Section 5.3.3 of the thesis. Often, the reallocation process can be applied iteratively, until the moment matching condition is satisfied or the computational resources are used up.

Mini-bucket as Sum-Max-Sum Inference on Augmented Model

The mini-bucket procedure given in Algorithm 3.3 works in a recursive fashion, interleaving the bucket splitting (approximation) and variable elimination (inference) steps. An alternate but equivalent formulation that will be important for later analysis involves predefining the collection of splitting operations, and then performing all the elimination steps afterwards. From this viewpoint, the mini-bucket bound can be written as a form of hybrid, sum-max-sum (or sum-min-sum) inference on an augmented model constructed by splitting the variables into multiple copies.

To be specific, we can view the partitioning of bucket $\mathbf{B}_i$ into mini-buckets $\{\mathbf{B}_{ir}\}$ as splitting

the variable x_i into multiple copies (or replicates) $\bar{\boldsymbol{x}}_i = [\bar{x}_{i^1}, \ldots, \bar{x}_{i^{R_i}}]$, one for each bucket; correspondingly, we let each factor $\psi \in \mathbf{B}_{i^r}$ be a function of the replicate $\bar{x}_{i^r}$, rather than the original x_i. This reformulation involves increasing the number of variables in the model (to include each bucket's replicates); to this end, we define an augmented (or replicated) vector of variables $\bar{\boldsymbol{x}} = [\bar{x}_{1^1}, \ldots, \bar{x}_{1^{R_1}}, \ldots, \bar{x}_{n^{R_n}}]$, and a set of factors $\{\bar{\psi}_{\bar{\alpha}}(\bar{\boldsymbol{x}}_{\bar{\alpha}})\colon \bar{\alpha} \in \bar{\mathcal{I}}\}$ that define an new graphical model on $\bar{\boldsymbol{x}}$,

$$\bar{p}(\bar{\boldsymbol{x}}) \propto \prod_{\bar{\alpha}\in\bar{\mathcal{I}}} \bar{\psi}(\bar{\boldsymbol{x}}_{\bar{\alpha}}).$$

At a high level, we can describe the construction of the new model $\bar{p}(\bar{\boldsymbol{x}})$ as consisting of two steps, detailed in Algorithm 3.4 and Algorithm 3.5 and corresponding, respectively, to the determination of the structure and factor reparameterization:

1. We first generate the factor cliques $\bar{\mathcal{I}}$ of $\bar{p}(\bar{\boldsymbol{x}})$ in Algorithm 3.4 by going through the mini-bucket splitting process; this decides the Markov network structure of $\bar{p}(\bar{\boldsymbol{x}})$, which for convenience we refer to as a "covering graph" of the original Markov graph G (a term taken from [Yarkony et al., 2010]). As illustrated in Figure 3.3, the partitioning process can be viewed as splitting the nodes to break cycles in the original graph G, so that the resulting covering graph will have lower induced width (controlled by the *ibound*), and be computationally easier to process.
2. With fixed structure (cliques) $\bar{\mathcal{I}}$, we then decide the actual values of the augmented factors $\{\bar{\psi}_{\bar{\alpha}}(\bar{\boldsymbol{x}}_{\bar{\alpha}})\colon \bar{\alpha} \in \bar{\mathcal{I}}\}$, such that the overall model will be equivalent to the original model $p(\boldsymbol{x})$ when all replicates of each variable are equal, that is,

$$\prod_{\bar{\alpha}\in\bar{\mathcal{I}}} \bar{\psi}(\bar{\boldsymbol{x}}_{\bar{\alpha}}) = \prod_{\alpha\in\mathcal{I}} \psi(\boldsymbol{x}_\alpha), \qquad \text{when } \ \bar{x}_{i^r} = x_i, \ \ \forall i \in [n], r \in [R_i]. \tag{3.9}$$

   In Algorithm 3.5, the augmented factors are determined by "splitting" each original factor $\psi_\alpha(\boldsymbol{x}_\alpha)$, $\alpha \in \mathcal{I}$ into a product of multiple factors $\{\varphi_r\}$, such that $\prod_r \varphi_r = \psi_\alpha$, and then assigning each φ_r to the rth-replicate of $\boldsymbol{x}_\alpha$ (i.e., the r-th augmented clique

which includes replicates from all original x_i in α). This splitting operation corresponds to the reparameterization perspective we discussed in the previous section, and gives a slightly more general algorithm general than "standard" mini-bucket elimination.

With the augmented model constructed, we can then perform the elimination procedure as defined in Algorithm 3.3; putting all the elimination steps together, it is straightforward to see that the result of MBE can be written as a mixed inference computation that interleaves sum and max (or min) operators,

$$\hat{Z} = \max_{\bar{x}_{n^2} \cdots \bar{x}_{nR_n}} \sum_{\bar{x}_{n^1}} \cdots \max_{\bar{x}_{1^2} \cdots \bar{x}_{1R_1}} \sum_{\bar{x}_{1^1}} \prod_{\bar{\alpha} \in \bar{\mathcal{I}}} \bar{\psi}(\bar{\boldsymbol{x}}_{\bar{\alpha}}). \tag{3.10}$$

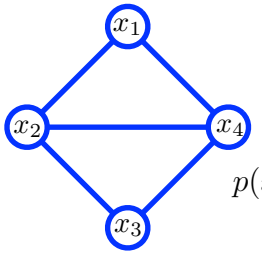
Specifically, each bucket's elimination operators are performed together, with the first mini-bucket replicate (i^1) eliminated using sum and the other replicates ($i^2 \cdots i^{R_i}$) eliminated with max. Because the induced width of the augmented model $\bar{p}(\bar{\boldsymbol{x}})$ is not greater than the *ibound* by construction, the calculation can be performed using exact bucket elimination on $\bar{p}(\bar{\boldsymbol{x}})$. Thus, the complexity of this sum-max-sum inference is only exponential in the *ibound*, which is controlled by the user. This closed form representation provides a useful tool for analyzing and improving the quality of mini-bucket approximations, and plays a crucial role in the development of our iterative, weighted mini bucket algorithm in Chapter 5.

Example 3.1. *Consider the toy model illustrated in Figure 3.3,*

$$p(\boldsymbol{x}) \propto \psi_{12}(x_1, x_2)\psi_{14}(x_1, x_4)\psi_{23}(x_2, x_3)\psi_{24}(x_2, x_4)\psi_{34}(x_3, x_4)\psi_1(x_1)\psi_2(x_2).$$

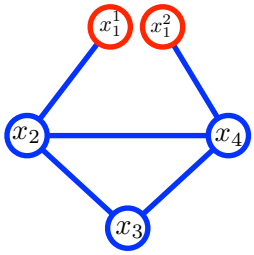
The mini-bucket procedure as described in Figure 3.3 defines a replicated variable $\bar{\boldsymbol{x}} = [\bar{x}_{1^1}, \bar{x}_{1^2}, \bar{x}_{2^1}, \bar{x}_{2^2}, \bar{x}_{3^1}, \bar{x}_{4^1}]$*, and an augmented model over* $\bar{\boldsymbol{x}}$*,*

$$\bar{q}(\bar{\boldsymbol{x}}) = \psi_{12}(\bar{x}_{1^1}, \bar{x}_{2^1})\psi_{14}(\bar{x}_{1^2}, \bar{x}_{4^1})\psi_{23}(\bar{x}_{2^1}, \bar{x}_{3^1})\psi_{24}(\bar{x}_{2^2}, \bar{x}_{4^1})\psi_{34}(\bar{x}_{3^1}, \bar{x}_{4^1})\psi_1(\bar{x}_{1^1})\psi_2(\bar{x}_{2^2}).$$



Original Graph:

$$p(\boldsymbol{x}) = \psi_{12}(x_1, x_2)\psi_{14}(x_1, x_4)\psi_{23}(x_2, x_3)\psi_{24}(x_2, x_4)\psi_{34}(x_3, x_4)\psi_1(x_2)\psi_2(x_2)$$

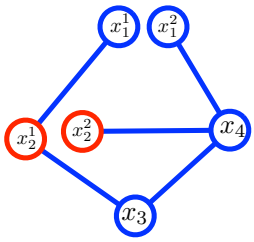


Eliminating x_1:

$$\mathbf{B}_1 = \{\psi_1, \psi_{12}, \psi_{14}\} \quad \overset{split}{\Longrightarrow} \quad \mathbf{B}_{1^1} = \{\psi_1, \psi_{12}\}, \mathbf{B}_{1^2} = \{\psi_{14}\}$$

$$\psi_{1^1}^{new}(x_2) = \sum_{x_1} \psi_1(x_1)\psi_{12}(x_1, x_2)$$

$$\psi_{1^2}^{new}(x_4) = \max_{x_1} \psi_{14}(x_1, x_4)$$

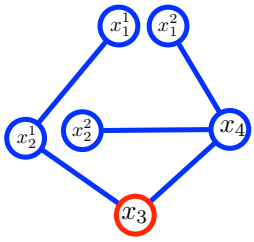


Eliminating x_2:

$$\mathbf{B}_2 = \{\psi_{1^1}^{new}, \psi_2, \psi_{23}, \psi_{24}\} \quad \overset{split}{\Longrightarrow} \quad \mathbf{B}_{2^1} = \{\psi_{1^1}^{new}, \psi_{23}\}, \mathbf{B}_{2^2} = \{\psi_2, \psi_{24}\}$$

$$\psi_{2^1}^{new}(x_3) = \sum_{x_2} \psi_{1^1}^{new}(x_2)\psi_{23}(x_2, x_3)$$

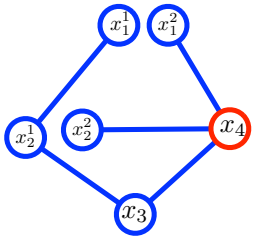
$$\psi_{2^2}^{new}(x_4) = \max_{x_2} \psi_2(x_2)\psi_{24}(x_2, x_4)$$



Eliminating x_3:

$$\mathbf{B}_3 = \{\psi_{2^1}^{new}, \psi_{34}\},$$

$$\psi_3^{new}(x_4) = \sum_{x_3} \psi_{2^1}^{new}(x_3)\psi_{34}(x_3, x_4)$$



Eliminating x_4:

$$\mathbf{B}_4 = \{\psi_{1^2}^{new}, \psi_{2^2}^{new}, \psi_3^{new}\}$$

$$\hat{Z} = \sum_{x_4} \psi_{1^2}^{new}(x_4)\psi_{2^2}^{new}(x_4)\psi_3^{new}(x_4)$$

Figure 3.3: Visualizing mini-bucket elimination. The mini-bucket partitioning process can be viewed as splitting the nodes to break cycles in the original Markov graph G (top), resulting a covering graph (bottom) – formally the Markov graph of some augmented model in which variables have been replicated – that has much lower induced width (a tree in this case), and is thus computationally easier to process.

Algorithm 3.4 Constructing augmented factor cliques for mini-bucket elimination

Input: The factor cliques $\mathcal{I}$ of the original graphical model. An elimination order $o = [x_1, \cdots, x_n]$, and an *ibound*. A bucket (clique) splitting strategy.
Output: The augmented factor cliques $\bar{\mathcal{I}}$ over the replicated variables $\bar{\boldsymbol{x}} = [\bar{x}_{1^1}, \ldots, \bar{x}_{1_1^R}, \ldots, \bar{x}_{n^{R_n}}]$.

Initialize $\bar{\mathcal{I}} = \mathcal{I}$.
for $i \leftarrow 1$ to n **do**

1. Find the factor cliques involving variable x_i: $\mathcal{B}_i \leftarrow \{\alpha \colon \alpha \in \bar{\mathcal{I}}, i \in \alpha\}$.

2. Partition $\mathcal{B}_i$ into R_i subgroups $\{\mathcal{B}_{i^r}\}$ according to the partition strategy, such that $\cup_{r=1}^{R_i} \mathcal{B}_{i^r} = \mathcal{B}_i$ and $|\cup_{\alpha \in \mathcal{B}_{i^r}} \alpha| \leq ibound + 1$ for all $r = 1, \ldots, R_i$. For each $\mathcal{B}_{i^r}$, construct $\bar{\mathcal{B}}_{i^r}$ by replacing x_i with $\bar{x}_{i^r}$ in all $\alpha \in \mathcal{B}_{i^r}$. Update $\bar{\mathcal{I}} = (\bar{\mathcal{I}} \setminus \mathcal{B}_i) \cup \{\bar{\mathcal{B}}_{i^r} \colon r = 1, \ldots, R_i\}$.

3. For $r = 1, \ldots, R_i$, let $\pi_{i^r} = \cup_{\bar{\alpha} \in \bar{\mathcal{B}}_{i^r}} \bar{\alpha} \setminus \{i^r\}$, and add π_{i^r} into $\bar{\mathcal{I}}$.

end for

Algorithm 3.5 Constructing augmented model for mini-bucket elimination

Input: The original factor graph $p(\boldsymbol{x}) \propto \prod_\alpha \psi_\alpha(\boldsymbol{x}_\alpha)$, and the augmented factor cliques $\mathcal{I}$ as output from Algorithm 3.4. A factor splitting strategy.
Output: An augmented model $\bar{p}(\bar{\boldsymbol{x}}) \propto \prod_{\bar{\alpha} \in \bar{\mathcal{I}}} \bar{\psi}_{\bar{\alpha}}(\bar{\boldsymbol{x}}_{\bar{\alpha}})$ over the replicated variables $\bar{\boldsymbol{x}}$.

Initialize to all-one factors: let $\bar{\psi}_{\bar{\alpha}}(\bar{\boldsymbol{x}}_\alpha) = 1$, for $\forall \bar{\alpha} \in \bar{\mathcal{I}}$ and $\bar{\boldsymbol{x}}$.
for $\alpha \in \mathcal{I}$ **do**

1. Let $\{\bar{\alpha}_r \colon r = 1, \ldots, R_\alpha\}$ be the set of augmented factors which include replicates of all the variables in α.

2. Split factor ψ_α into a product of φ_r such that $\prod_{r=1}^{R_\alpha} \varphi_r = \psi_\alpha$ (e.g., by uniform splitting: $\varphi_r = (\psi_\alpha)^{1/R_\alpha}$).

3. Update: $\bar{\psi}_{\bar{\alpha}_r}(\bar{\boldsymbol{x}}_{\bar{\alpha}}) \leftarrow \bar{\psi}_{\bar{\alpha}_r}(\bar{\boldsymbol{x}}_{\bar{\alpha}}) \cdot \varphi_r(\bar{\boldsymbol{x}}_{\bar{\alpha}_r})$ for all $r = 1, \ldots R_\alpha$.

end for

Combining all the elimination steps in Figure 3.3, one can show that the resulting upper bound $\hat{Z}$ equals

$$\hat{Z} = \sum_{\bar{x}_{4^1}} \sum_{\bar{x}_{3^1}} \max_{\bar{x}_{2^2}} \sum_{\bar{x}_{2^1}} \max_{\bar{x}_{1^2}} \sum_{\bar{x}_{1^1}} \bar{q}(\bar{\boldsymbol{x}}),$$

which can be solved using exact variable elimination; the complexity of this procedure is $O(d^2)$, since the covering graph is a tree (induced width of 1).

To summarize, mini-bucket elimination is reframed into a three-step procedure:

1. Generate a set of factor cliques $\bar{\mathcal{I}}$ over the replicated variables (i.e., decide on the covering graph structure) using Algorithm 3.4.
2. Determine the actual values of the augmented factors $\{\bar{\psi}_{\bar{\alpha}}(\bar{\boldsymbol{x}}_{\bar{\alpha}})\colon \bar{\alpha} \in \bar{\mathcal{I}}\}$ by Algorithm 3.5.
3. Perform sum-max-sum inference (3.10) on the augmented model via exact bucket elimination, to get an upper bound of the original partition function. (A lower bound can be obtained by using min instead.)

Note that we still require three components in order to fully specify the mini-bucket algorithm: (1) a *bucket (or clique) splitting strategy* as used in Algorithm 3.4 for partitioning the cliques; (2) an *factor splitting or reallocation strategy* to specify or adjust the augmented factors $\{\bar{\psi}_{\bar{\alpha}}(\bar{\boldsymbol{x}}_{\bar{\alpha}})\colon \bar{\alpha} \in \bar{\mathcal{I}}\}$ given the clique structure $\bar{\mathcal{I}}$ – the uniform splitting suggested in Algorithm 3.5 provides a simple heuristic or initialization, but can be further improved; and (3) a strategy to decide which replicate should be (uniquely) assigned the sum operator when performing the elimination steps – we arbitrarily assume this is the first replicate in (3.10).

All three of these components can greatly influence the results, and ideally should be optimized to obtain the best performance. Interestingly, these choices appear to have very different degrees of difficulty: selecting the best clique splitting strategy (for deciding the structure) involves a difficult combinatorial optimization, and currently only greedy heuristics exist [see e.g., Dechter and Rish, 2003, Rollon and Dechter, 2010]. On the other hand,

with the structure fixed, it is in fact possible to optimize (at least partly) the other two components in a principled way; this is discussed in Chapter 5 in the context of the more general, weighted mini-bucket procedure that we propose.

3.1.3 Elimination as Message Passing

In this section, we consider another view of the exact bucket elimination process in Algorithm 3.2 that treats it as passing "messages" (or propagating beliefs) between nodes and their parents along a tree structure; this reformulates the exact bucket elimination algorithm as *forward-backward belief propagation* (BP) on trees. Belief propagation can be further extended to more general graph structures that include cycles, or loops, yielding *loopy belief propagation*, which is no longer exact but often provides efficient and accurate approximations in practice.

At a high level, each newly generated factor ψ_i^{new} in the bucket elimination step (3.2) can be viewed as a "message" that is passed forward from node i (or the bucket $\mathbf{B}_i$) to its parents π_i. Correspondingly, we can treat the marginals $p(\boldsymbol{x}_{\pi_i})$ in the backward elimination (3.3) as messages that are passed backward from node i to its children. In the case that the graph G is a tree, the forward and backward elimination processes can be directly viewed as passing messages between the nodes, first from leaves to roots, and then backwards from the roots to the leaves. However, for more general models, especially those with higher order cliques, it is more notationally convenient to view the elimination as passing messages over a tree structure, known as a *junction tree*, formed by subsets of nodes in the original graph.

We start with the simple case of tree-structured graphs, and formulate a forward-backward belief propagation algorithm that passes messages between the variable nodes on the tree. We then introduce the definition of a junction graph and junction tree, and formulate bucket elimination as a forward-backward belief propagation on a junction tree. Finally, we introduce loopy belief propagation as an approximation algorithm, extending forward-backward

belief propagation to general loopy graphs.

Forward-Backward BP on Trees

We begin by setting up some notation. Assume $G = (V, E)$ is a tree, and o is its tree-order. Let $\mathrm{pa}_G(i)$ be the *parent* of node i along order o, and call i a *child* of j if $j = \mathrm{pa}_G(i)$. We denote by $\partial_G(i)$ the neighborhood of i, that is, $\partial_G(i) = \{i' \in V \colon (i'i) \in E\}$. The operator "\" denotes set-theoretic difference; for example, $\partial_G(i) \setminus \{j\} := \{i' \in \partial_G(i) \colon i' \neq j\}$ denotes neighboring nodes of i except j. Any model $p(\boldsymbol{x})$ on tree G must be a pairwise model that includes only factors of single or pairs of variables, and can be written as

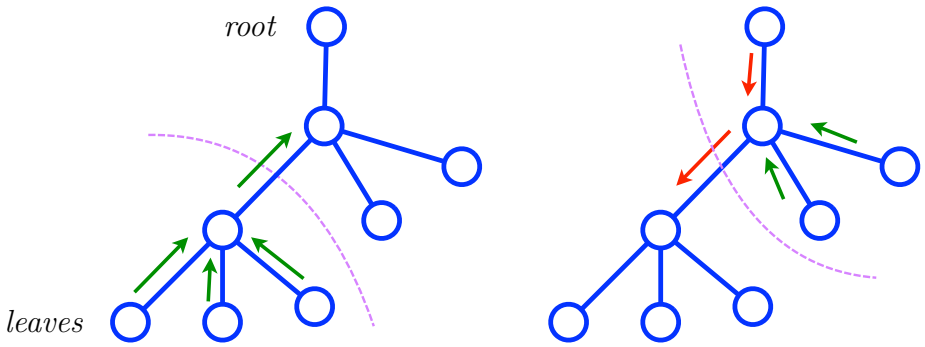
$$p(\boldsymbol{x}) \propto \prod_{\in V} \psi_i(x_i) \prod_{(ij)\in E} \psi_{ij}(x_i, x_j).$$

If we perform bucket elimination on G along order o, then π_i equals $\mathrm{pa}_G(i)$, the unique parent of node i on G along order o. If we consider the new factor $\psi^{new}(x_{\pi_i})$ as a "message", denoted as $m_{i\to\pi_i}(x_{\pi_i})$, passed from node i to its parent π_i, then the forward elimination step (3.2) can be rewritten as an update equation on the messages,

$$m_{i\to\pi_i}(x_{\pi_i}) \leftarrow \sum_{x_i} \psi_{i\pi_i}(x_i, x_{\pi_i})\psi_i(x_i) \prod_{i'\in\partial_G(i)\setminus\{\pi_i\}} m_{i'\to i}(x_i), \tag{3.11}$$

where we take the product of all the factors involving i and all the messages sent forward into i from its children i' (since any node $i' \in \partial_G(i) \setminus \{\pi_i\}$ is a child of i; see Figure 3.4a), and then eliminate x_i from the product.

Similarly, the backward elimination can be treated as passing messages from each parent π_i



(a) Forward phase (b) Backward phase

Figure 3.4: Illustration of forward and backward message passing in a tree, in which messages are computed (a) forward from leaves to root, then (b) backward from root to leaves. (a) Forward messages (green) only depend on earlier forward messages, while (b) backward messages (red) depend on both earlier backward messages as well as some forward messages.

backwards to node i, via a somewhat more complex definition of *backward message* $m_{\pi_i \to i}(x_i)$,

$$m_{\pi_i \to i}(x_i) = \sum_{x_{\pi_i}} \frac{\psi_{i\pi_i}(x_i, x_{\pi_i}) p(x_{\pi_i})}{m_{i \to \pi_i}(x_{\pi_i})}. \tag{3.12}$$

Then the backward elimination in (3.3) is equivalent to a similar message update rule,

$$m_{\pi_i \to i}(x_{\pi_i}) \leftarrow \sum_{x_{\pi_i}} \psi_{i\pi_i}(x_i, x_{\pi_i}) \psi_{\pi_i}(x_{\pi_i}) \prod_{i' \in \partial_G(\pi_i) \setminus \{i\}} m_{i' \to \pi_i}(x_{\pi_i}), \tag{3.13}$$

which is identical to the forward update (3.11), except that the incoming messages (from any node $i' \in \partial_G(\pi_i) \setminus \{i\}$) can now be interpreted as including both the earlier backward message sent from the parent of π_i and the forward messages sent from children of π_i except i (that is, the *siblings* of i). See Figure 3.4b for an illustration.

In addition, the marginal distribution of single variable $p(x_i)$ and adjacent pairs of variables

Algorithm 3.6 Forward-backward belief propagation on trees

Input: A pairwise graphical model $p(\boldsymbol{x}) \propto \prod_{\in V} \psi_i(x_i) \prod_{(ij)\in E} \psi_{ij}(x_i, x_j)$ on a tree G; an tree order $o = [1, \dots n]$ of G.
Forward: For $i = 1$ to n, calculate $m_{i\to\pi_i}$ by (3.11).
Backward: For $i = n$ to 1, calculate $m_{\pi_i\to i}$ by (3.13).
Return: Calculate marginals $\{p(x_i)\colon i \in V\}$ and $\{p(x_i, x_j)\colon (ij) \in E\}$ by (3.14)-(3.15).

$p(x_i, x_j)$, $\forall (ij) \in E$, can be calculated in terms of the messages via,

$$p(x_i) \propto \psi_i(x_i) \prod_{i'\in\partial_G(i)} m_{i'\to i}(x_i), \tag{3.14}$$

$$p(x_i, x_j) \propto \psi_i(x_i, x_j)\psi_i(x_i)\psi_j(x_j) \prod_{i'\in\partial_G(i)\setminus\{j\}} m_{i'\to i}(x_i) \prod_{j'\in\partial_G(j)\setminus\{i\}} m_{j'\to j}(x_j), \tag{3.15}$$

where $p(x_i)$ equals the product of the factor ψ_i and all the messages sent into node i from its neighboring nodes, and similarly $p(x_i, x_j)$ equals the product of all the factors that only involve i and/or j, and the messages sent into i or j from the neighboring nodes outside of the (ij) clique. Note that (3.15) is correct only when (ij) is an edge of tree G.

Overall, we can rewrite the bucket elimination of Algorithm 3.2 as a forward-backward message passing algorithm known as *belief propagation*, shown in Algorithm 3.6, in which we first pass messages in the forward direction, beginning at the leaves and passing from each node to its parent, and then work backwards from the root (or roots) passing from nodes to their children; finally, we calculate the marginal probabilities based on the messages by (3.14)-(3.15).

Junction Tree and Junction Tree BP

When the graph is not a tree, or when the model includes high order factors, it is no longer convenient (although still possible) to represent bucket elimination as passing messages between the variable nodes. It turns out to be more natural to consider passing messages

over a *junction tree* formed by subsets of variables (called clusters). In this section, we introduce the notion of a junction tree and rewrite bucket elimination as a junction tree BP that passes messages between clusters in the junction tree.

We start by introducing the general concepts of a cluster graph and a junction graph.

Definition 3.5 (**Cluster Graph**). *Given a graphical model* $p(\boldsymbol{x}) = \prod_{\alpha\in\mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha)$*, a cluster graph is a graph of subsets of the variables. Formally, a cluster graph is a triple* $(\mathcal{G}, \mathcal{C}, \mathcal{S})$*, where* $\mathcal{C}$ *and* $\mathcal{S}$ *are families of subsets of variables, and* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *is an undirected graph. Each node* $k \in \mathcal{V}$ *is associated with a variable subset (called a cluster)* $c_k \in \mathcal{C}$*, and each edge* $(kl) \in \mathcal{E}$ *is also associated with a subset* $s_{kl} \in \mathcal{S}$ *(called a* separator*), such that* $s_{kl} = c_k \cap c_l$*. In addition,* $\mathcal{C}$ *should subsume* $\mathcal{I}$ *in the following sense:*

(Subsumption Property). *For any* $\alpha \in \mathcal{I}$*, there exists at least one element in* $\mathcal{C}$*, denoted by* $c[\alpha]$*, such that* $\alpha \in c[\alpha]$*.*

The subsumption condition above ensures that the original distribution $p(\boldsymbol{x}) = \prod_{\alpha\in\mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha)$ can be rewritten (or reparameterized) into the form $p(\boldsymbol{x}) = \prod_{\beta\in\mathcal{C}} \psi_\beta(\boldsymbol{x}_\beta)$ by transforming $\psi_\beta \leftarrow \prod_{\alpha\in\mathcal{I}:\, c[\alpha]=\beta} \psi_\alpha$. Thus, we will always rewrite the distribution $p(\boldsymbol{x})$ into the latter form whenever we use a junction graph based on clusters $\mathcal{C}$.

Definition 3.6 (**Junction Graph**). *A cluster graph is called a junction graph if it satisfies the* **running intersection property***: for each variable* $i \in V$*, the sub-graph consisting of the clusters and separators that include* i *is a connected tree. A junction graph* G *is a junction tree if it is tree structured. A junction tree of* G *is also called a tree decomposition of* G*. The width of a junction graph is its maximum cluster size minus, that is,* $\max\{|\alpha| - 1 \colon \alpha \in \mathcal{C}\}$*.*

Constructing Junction Tree via Triangulation. It is perhaps not immediately clear how to construct a junction graph (or tree). Fortunately, we can conveniently construct a junction tree $\mathcal{G}$ based on the induced graph $(\tilde{G}, o)$ obtained by triangulation along ordering o: define

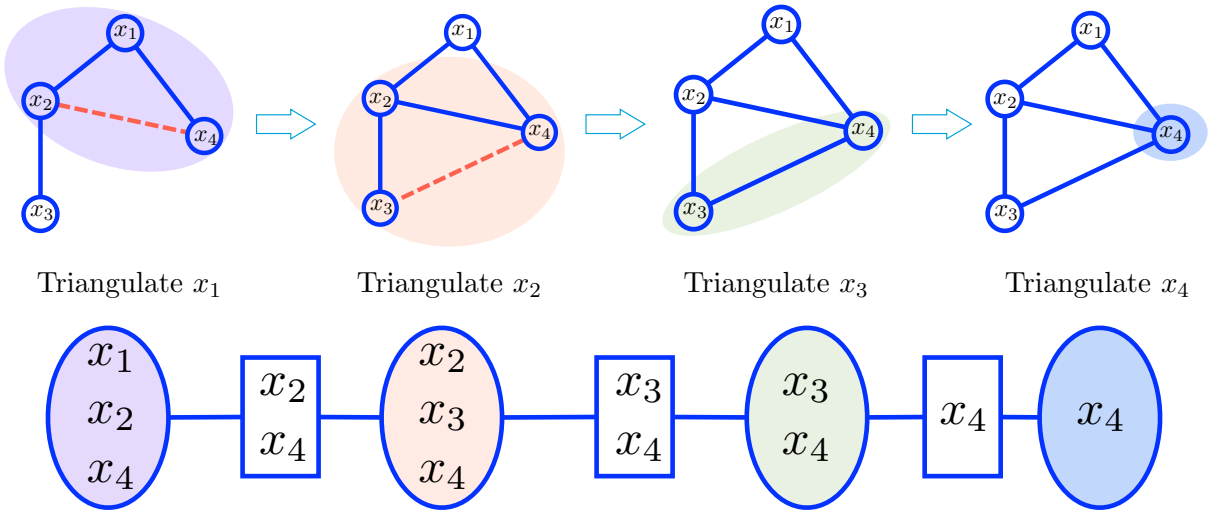


Figure 3.5: Constructing a junction tree via triangulation.

the clusters by $\mathcal{C} = \{c_i := \text{pa}_{\tilde{G}}(i) \cup \{i\} : i \in V\}$, and add an edge between two clusters c_i and c_j if node j is the first parent of node i in the induced graph $(\tilde{G}, o)$, that is, $j = \min\{j' : j' \in \text{pa}_{\tilde{G}}(i)\}$. To connect this induced graph construction to the bucket elimination process, the junction tree simply records the variable scopes of the buckets $\{\mathbf{B}_i\}$ and their message trajectories (since j is the first parent of i iff the factor ψ_i^{new} created when eliminating x_i falls into the bucket $\mathbf{B}_j$ of variable x_j). This process is visualized for a small graphical model in Figure 3.5.

We can then readily see the following properties:

1. $\mathcal{G}$ is a junction tree, and the elimination order o also induces a tree order on the clusters $\mathcal{C}$, since any cluster can have at most one parent in $\mathcal{G}$ (that is, any ψ_i^{new} can fall into only one subsequent bucket). In other words, each cluster c_i in junction tree $\mathcal{G}$ corresponds to node i in the induced graph $(\tilde{G}, o)$, and the (unique) parent $\text{pa}_{\mathcal{G}}(i)$ of c_i in $\mathcal{G}$ corresponds the first parent of node i in $(\tilde{G}, o)$.
2. $\mathcal{G}$ satisfies the running intersection property: the subgraph including variable x_i records the message passing trajectory of all the factors associated with x_i, and this trajectory can only be a tree (again, because any factor can only assigned to or passed into a

single bucket).

3. The separator s_{ij} between cluster c_i and its parent c_j, $j = \mathrm{pa}_{\mathcal{G}}(i)$, equals π_i, the parent set of node i in the induced graph $(\tilde{G}, o)$, and equivalently, π_i, the variable scope of ψ_i^{new} in the bucket elimination step (3.2).
4. We can rewrite the original graphical model $p(\boldsymbol{x}) = \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha)$ in a form defined on the clusters of the junction tree, $p(\boldsymbol{x}) = \prod_{c_i \in \mathcal{C}} \psi_{c_i}(\boldsymbol{x}_{c_i})$, by defining $\psi_{c_i} \leftarrow \prod\{\psi_\alpha \in \mathbf{B}_i : \alpha \in \mathcal{I}\}$.

With the junction tree as constructed above, we can now rewrite the bucket elimination algorithm as passing messages on the junction tree. Specifically, we define the forward and backward messages between cluster c_i and its parent $c_{\mathrm{pa}_{\mathcal{G}}(i)}$ in junction tree $\mathcal{G}$, in terms of the quantities in bucket elimination,

$$\begin{aligned}
&\text{Forward message:} && m_{i \to \mathrm{pa}_{\mathcal{G}}(i)}(\boldsymbol{x}_{\pi_i}) = \psi_i^{new}(\boldsymbol{x}_{\pi_i}), \\
&\text{Backward message:} && m_{\mathrm{pa}_{\mathcal{G}}(i) \to i}(\boldsymbol{x}_{s_{i\mathrm{pa}_{\mathcal{G}}(i)}}) = \frac{1}{m_{i \to \mathrm{pa}_{\mathcal{G}}(i)}(\boldsymbol{x}_{\pi_i})} \sum_{\boldsymbol{x}_{\neg \pi_i}} p(\boldsymbol{x}_{c_{\mathrm{pa}_{\mathcal{G}}(i)}}),
\end{aligned}$$

where $m_{i \to \mathrm{pa}_{\mathcal{G}}(i)}$ and $m_{\mathrm{pa}_{\mathcal{G}}(i) \to i}$ are the forward and backward messages between cluster c_i and its parent $c_{\mathrm{pa}_{\mathcal{G}}(i)}$, respectively. Both the messages are functions of $\boldsymbol{x}_{\pi_i}$, where π_i is the separator between cluster c_i and its parent c_j, $j = \mathrm{pa}_{\mathcal{G}}(i)$. Note that the message definitions differ slightly from the tree forms (3.11)–(3.12); roughly, these differences arise because the messages on junction trees correspond to messages between *cliques*, rather than between variables. Comparing the backward messages, for example, the junction tree message is defined on the separator set $s_{i\mathrm{pa}_{\mathcal{G}}(i)}$, rather than the neighboring variable x_i, and no factor ψ appears explicitly in the equation; any factors are included in the marginal $p(\boldsymbol{x}_{c_{\mathrm{pa}_{\mathcal{G}}(i)}})$.

By re-arranging, one can then show that both the forward elimination (3.2) and backward

Algorithm 3.7 Junction tree belief propagation

Input: A graphical model $p(\boldsymbol{x}) = \prod_{\alpha\in\mathcal{C}} \psi_\alpha(\boldsymbol{x}_\alpha)$, and a junction tree $(\mathcal{G}, \mathcal{C}, \mathcal{E})$ (e.g., constructed via triangulation), an tree order $o = [1, \ldots, n]$ of$\mathcal{G}$.
Output: The marginals $\{p(\boldsymbol{x}_{c_i})\colon c_i \in \mathcal{C}\}$.
Forward: For $i = 1$ to n, calculate $m_{i\to \mathrm{pa}_\mathcal{G}(i)}$ by (3.16).
Backward: For $i = n$ to 1, calculate $m_{\mathrm{pa}_\mathcal{G}(i)\to i}$ by (3.16).
Return: Calculate the marginals $\{p(\boldsymbol{x}_{c_i})\}$ by (3.17).

elimination (3.3) can be rewritten into the following message passing form,

$$m_{i\to j}(\boldsymbol{x}_{s_{ij}}) \propto \sum_{\boldsymbol{x}_{c_i\setminus c_j}} \psi_{c_i}(\boldsymbol{x}_{c_i}) \prod_{i'\in\partial_\mathcal{G}(i)\setminus\{j\}} m_{i'\to i}(\boldsymbol{x}_{s_{i'i}}), \tag{3.16}$$

where $\partial_\mathcal{G}(j)$ is the neighborhood of the jth cluster in $\mathcal{G}$, that is, $\partial_\mathcal{G}(j) = \{j'\colon (j'j) \in \mathcal{E}\}$. In addition, the marginal probabilities can be calculated from the messages via

$$p(\boldsymbol{x}_{c_i}) \propto \psi_{c_i}(\boldsymbol{x}_{c_i}) \prod_{i'\in\partial_\mathcal{G}(i)} m_{i'\to i}(\boldsymbol{x}_{s_{i'i}}). \tag{3.17}$$

Similar to BP on trees, we obtain the junction tree belief propagation algorithm in Algorithm 3.7, which is equivalent to the bucket elimination algorithm in Algorithm 3.2, by passing the messages first in the forward direction, beginning at the leaves and passing from each node to its parent, and then working backwards from the roots passing from nodes to their children.

Remarks. (1). Although we mainly discuss the specific junction tree constructed via the variable elimination process here, Algorithm 3.7 works for arbitrary junction trees.

(2). On the other hand, note that message passing on an arbitrary junction tree along its tree order is equivalent to a variable elimination process with any variable elimination order consistent with the junction tree-order in the following sense: if node i ranks earlier than node j in the variable elimination order, then there exists no cluster in the junction tree that

contains (ij), while its parent (along the junction tree-order) contains i but not j.

(3). The complexity of the message passing in Algorithm 3.7 is exponential in the width (i.e., the maximum cluster size minus one) of the junction tree $\mathcal{G}$, which is equal to the induced width of the ordered graph (G, o) if $\mathcal{G}$ is constructed via elimination along ordering o. Additionally, the minimum possible width of any junction tree of G equals the induced width (or tree width) of G. In fact, the complexity of exact algorithms are in general fundamentally restricted by the induced width, regardless of the particular form of the algorithm used. Therefore, for graphs with high induced width (which are common in practice), it is critical to define efficient approximation algorithms to provide a trade-off between computational complexity and accuracy.

Loopy Belief Propagation for Approximate Inference

Preceding sections presented the forward-backward belief propagation on trees and junction trees, both giving the exact variable elimination results. However, they have obvious limitations when applied to practical graphs with high induced widths: the pairwise BP in Algorithm 3.6 works only on trees by definition (i.e., induced width of one), while the minimum width of any junction tree equals the induced width, making the complexity of junction tree BP exponential in the induced width.

An important observation was made by Pearl [1988], that the message updates in belief propagation ((3.11)-(3.13) and (3.16)) involve only the messages and factors associated with the local neighborhood of a node, and therefore can be applied even when the graph includes cycles. This gives a set of *loopy belief propagation* algorithms, which do not in general provide the correct marginals (as with BP on tree structures), are widely used for efficient, approximate inference.

Algorithm 3.8 shows loopy BP for pairwise models, which extends BP on trees from Al-

Algorithm 3.8 Loopy belief propagation for pairwise models

Input: A pairwise model $p(\boldsymbol{x}) \propto \prod_{\in V} \psi_i(x_i) \prod_{(ij)\in E} \psi_{ij}(x_i, x_j)$ on a graph $G = (V, E)$.
Initialize all the messages $\{m_{i\to j}(x_j) \colon \forall (kl) \in E\}$.
Repeat until convergence:

For all the edges $(ij) \in E$ (in some order as defined by a scheduling scheme), update the messages via

$$m_{i\to j} \propto \sum_{x_i} \psi_{ij}(x_i, x_j)\psi_i(x_i) \prod_{i'\in\partial_G(i)\setminus\{j\}} m_{i'\to i}(x_i), \tag{3.18}$$

End
Return: Calculate the approximate marginals $\{p(x_i) \colon i \in V\}$ and $\{p(x_i, x_j) \colon (ij) \in E\}$ by (3.14)-(3.15).

Note: "$\propto$" denotes equality up to a multiplicative constant; for numerical stability, the messages $m_{i\to j}(x_j)$ are usually normalized (e.g., to sum to one) each time they are updated.

gorithm 3.6; it uses the same message update as Algorithm 3.6, but has modifications for initialization and message scheduling:

1. Because the graph may not contain any leaves, and the messages may depend on each other in a cyclic way, one needs to initialize all the messages at the beginning and then iteratively update the messages until they converge to some equilibrium, or meet some stopping criterion.
2. In addition, at each iteration, there are no longer any constraints on the ordering of the message updates. We can use an *asynchronous scheduling*, for example one that obeys a forward-backward-like ordering when updating the messages (w.r.t. to some order o), or proceed in a more distributed fashion by using a *synchronous scheduling* where all nodes receive messages in parallel and then send out updated messages in parallel. It is easy to show that in tree structured graphs, and with any schedule that properly covers all the edges, the messages in loopy BP will converge to the messages of exact, forward-backward BP on trees within a finite number of iterations, related to the diameter of the tree.

Algorithm 3.9 Loopy junction graph belief propagation

Input: A graphical model $p(\boldsymbol{x}) = \prod_{c_k \in \mathcal{C}} \psi_{c_k}(\boldsymbol{x}_{c_k})$, and a junction graph $(\mathcal{G}, \mathcal{C}, \mathcal{E})$, where $\mathcal{C} = \{c_k\}$ and $\mathcal{S} = \{s_{kl}\}$ are the clusters and separaters, respectively.
Initialize all the messages $\{m_{k \to l}(\boldsymbol{x}_{s_{kl}}) \colon \forall s_{kl} \in \mathcal{S}\}$.
Until convergence, iteratively update messages on all the edges via

$$m_{k \to l}(\boldsymbol{x}_{s_{kl}}) \propto \sum_{\boldsymbol{x}_{c_k \setminus s_{kl}}} \psi_{c_k}(\boldsymbol{x}_{c_k}) \prod_{k' \in \partial_{\mathcal{G}}(k) \setminus \{k\}} m_{k' \to k}(\boldsymbol{x}_{s_{k'k}}).$$

Return: Calculate the approximate marginals $\{p(\boldsymbol{x}_{c_i})\}$ by

$$p(\boldsymbol{x}_{c_k}) \propto \psi_{c_k}(\boldsymbol{x}_{c_k}) \prod_{k' \in \partial_{\mathcal{G}}(k)} m_{k' \to k}(\boldsymbol{x}_{c_k}).$$

Note: as in Algorithm 3.8, it is usual to normalize each message when updated.

Convergence Issues. Unfortunately, loopy BP does not necessarily convergence on general loopy graphs. In particular, synchronous scheduling often results in poorer convergence behavior than asynchronous scheduling. There have been many theoretical works on the convergence properties of loopy BP; see for example Ihler et al. [2005], Wainwright and Jordan [2008] and references therein.

In practice, various methods for improving the convergence of BP have been proposed. One simple way is to use *damping*. That is, instead of using the message $m_{i \to j}^{new}$ as calculated from the update rule (3.18), we use a damped message $m_{i \to j}^{damp}$ by e.g., geometrically averaging the old and newly computed message:

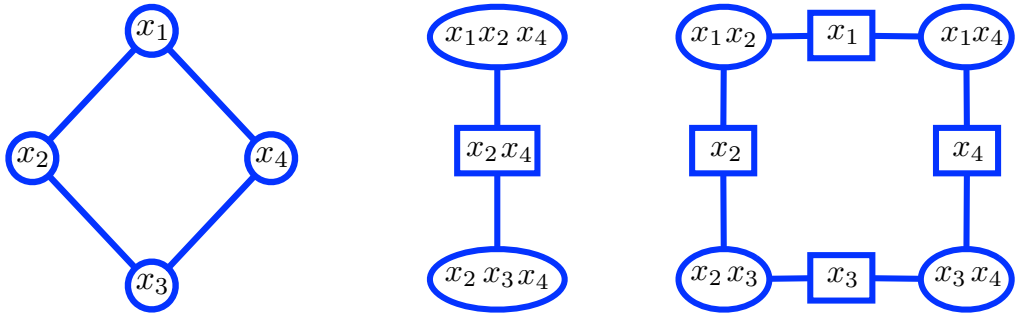
$$\log m_{i \to j}^{damp} \leftarrow \beta \log m_{i \to j}^{new} + (1 - \beta) \log m_{i \to j}^{old}, \tag{3.19}$$

where $m_{i \to j}^{old}$ is the message at the previous iteration, and $0 < \beta \leq 1$ is a damping coefficient. Obviously, if $\beta = 1$, this reduces to standard loopy BP without damping. Damping can help loopy BP converge more often, but may cause the algorithm to converge more slowly to a fixed point. Therefore, for the experiments in this thesis, we apply damping only we when find that loopy BP fails to converge after some number (e.g., 100) of iterations.

In addition to the simple damping method, many other, more advanced methods have been investigated to improve convergence. Residual BP [Elidan et al., 2006] works by using an adaptive scheduling that prioritizes updating messages that differ the most from their previous values, and has been found to converge faster and more often than synchronous or many other asynchronous schedules. Another set of work relates to deriving *double loop algorithms* that are guaranteed to converge to a fixed point of loopy BP [see e.g., Welling and Teh, 2001, Yuille, 2002]; these are explained in more detail in Section 3.2 when we discuss the variational interpretation of loopy BP.

Loopy BP on Junction Graphs. Similar to pairwise loopy BP, we can derive a loopy BP algorithm for general junction graphs, shown in Algorithm 3.9; this generalizes exact inference via BP on junction trees in Algorithm 3.7. Compared to pairwise loopy BP, the advantage of loopy BP on junction graphs lies in the flexibility to define arbitrary loopy junction graphs with various widths. Although the width of any junction tree is constrained by the induced width, there are no substantial constraints on the width of general junction graphs with circles, except that it can not be smaller than the maximum clique size minus one (i.e., $\max_{\alpha\in\mathcal{I}} |\alpha| - 1$) of the model $p(\boldsymbol{x}) \propto \prod_{\alpha\in\mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha)$, due to the subsumption requirement in Definition 3.5. In particular, for any pairwise Markov random field, we can always construct a junction graph with width 1 by selecting its edges as the clusters; see Figure 3.6 for an example.

The width of the loopy junction graph provides a cost-accuracy trade-off similar to the *ibound* in mini-bucket elimination: junction graphs with higher width may need to introduce fewer loops, leading to more accurate solutions, while junction graphs with lower width are only possible by introducing many loops, usually causing loopy BP to be less accurate or less likely to converge.



(a) (b) (c)

Figure 3.6: Junction tree vs. junction graph. (a) A simple pairwise Markov random field, with (b) a junction tree, and (c) a loopy junction graph. The width of the junction graph is 1, while that of the junction tree, and the induced with of the Markov random field, is 2.

3.2 Dual View – Variational Methods

In contrast to variable elimination methods, which directly eliminate the variables in some sequence, variational optimization methods transform the inference problem into a functional optimization problem over the space of distributions, minimizing a divergence to the target distribution. Variational methods can be elegantly viewed as applying the theory of convex conjugate duality to the inference task. Defining various approximations to the dual optimization problem then leads to many efficient, approximate inference algorithms, including another interpretation of loopy belief propagation [Pearl, 1988], as well as mean field [Saul and Jordan, 1995] and many of their variants (see Wainwright and Jordan [2008] and references therein).

This section reviews the variational inference framework and several important algorithms. Section 3.2.1 introduces the exponential family form for probablistic graphical models, and the variational form for the log-partition function (its convex conjugate function). The subsequent sections then introduce several categories of variational inference methods based on approximating the variational form, including loopy belief propagation (BP) and its extensions on junction and factor graphs in Section 3.2.3, tree reweighted BP and convex conditional entropy decompositiong in Section 3.2.4, and mean field and related methods in

Section 3.2.5. Finally, although our presentation mostly focuses on sum-inference, we give a brief discussion on max-inference in Section 3.2.6, where a linear programming relaxation plays an important role similar to the dual representation of the log-partition function in sum-inference, and enables a spectrum of efficient max-inference algorithms.

3.2.1 Exponential Family and Conjugate Duality

We start by rewriting the factorized distribution $p(\boldsymbol{x}) \propto \prod_{\alpha\in\mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha)$ into an overcomplete exponential family form [Wainwright and Jordan, 2008],

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \exp\big(\theta(\boldsymbol{x}) - \Phi(\boldsymbol{\theta})\big), \qquad \theta(\boldsymbol{x}) = \sum_{\alpha\in\mathcal{I}} \theta_\alpha(\boldsymbol{x}_\alpha), \tag{3.20}$$

where $\theta_\alpha(\boldsymbol{x}_\alpha) = \log(\psi_\alpha(\boldsymbol{x}_\alpha))$ are the log of our usual factors ψ_α. We denote by $\boldsymbol{\theta} = \{\theta_\alpha(\boldsymbol{x}_\alpha)\colon \alpha \in \mathcal{I}, \boldsymbol{x}_\alpha \in \mathcal{X}_\alpha\}$ the vector formed by elements (values) of the θ_α; these are called the *natural parameters* of the exponential family distribution. The log-partition function $\Phi(\boldsymbol{\theta})$, treated as a function of $\boldsymbol{\theta}$, normalizes the distribution:

$$\Phi(\boldsymbol{\theta}) = \log \sum_{\boldsymbol{x}} \exp(\theta(\boldsymbol{x})).$$

This exponential family representation has several important properties that are central to both inference and learning [e.g., Wainwright and Jordan, 2008].

Proposition 3.1. *(1).* $\Phi(\boldsymbol{\theta})$ *can be represented using the variational form,*

$$\Phi(\boldsymbol{\theta}) = \max_{q\in\mathcal{P}_{\boldsymbol{x}}} \{\mathbb{E}_q(\theta(\boldsymbol{x})) + H(\boldsymbol{x}; q)\}, \tag{3.21}$$

where $\mathcal{P}_{\boldsymbol{x}}$ *is the set of all possible distributions defined on* $\boldsymbol{x}$*, that is,*

$$\mathcal{P}_{\boldsymbol{x}} = \Big\{q(\boldsymbol{x})\colon \sum_{\boldsymbol{x}} q(\boldsymbol{x}) = 1, \quad \textit{and} \;\; q(\boldsymbol{x}) \geq 0, \forall \boldsymbol{x}\Big\},$$

and $H(\boldsymbol{x}; q)$ is the entropy of distribution q, that is, $H(\boldsymbol{x}; q) = -\mathbb{E}_q(\log q(\boldsymbol{x}))$. In addition, the maximum q^* is obtained when $q^*(\boldsymbol{x}) = p(\boldsymbol{x}) = \exp(\theta(\boldsymbol{x}) - \Phi(\boldsymbol{\theta}))$.

(2). $\Phi(\boldsymbol{\theta})$ is a convex function of $\boldsymbol{\theta}$.

(3). The derivatives of $\Phi(\boldsymbol{\theta})$ equal the marginal distributions of $p(\boldsymbol{x})$, that is,

$$\frac{\partial \Phi(\boldsymbol{\theta})}{\partial \theta_\alpha(\boldsymbol{x}_\alpha)} = p(\boldsymbol{x}_\alpha), \quad \forall \alpha \in \mathcal{I}, \quad \forall \boldsymbol{x}_\alpha \in \mathcal{X}_\alpha. \tag{3.22}$$

Proof. (1). The proof of (3.21) stems from the non-negativity of KL-divergence, i.e.,

$$\min_{q \in \mathcal{P}_{\boldsymbol{x}}} \mathrm{KL}(q(\boldsymbol{x}) || p(\boldsymbol{x})) = 0, \tag{3.23}$$

where the minimum is obtained when $q(\boldsymbol{x}) = p(\boldsymbol{x})$. Plugging the exponential form $p(\boldsymbol{x}) = \exp(\theta(\boldsymbol{x}) - \Phi(\boldsymbol{\theta}))$ into the definition of the KL divergence,

$$\begin{aligned} \mathrm{KL}(q||p) &= \mathbb{E}_q \log[q(\boldsymbol{x})/p(\boldsymbol{x})] \\ &= -H(\boldsymbol{x}; q) - \mathbb{E}_q[\theta(\boldsymbol{x}) - \Phi(\boldsymbol{\theta})] \\ &= -H(\boldsymbol{x}; q) - \mathbb{E}_q[\theta(\boldsymbol{x})] + \Phi(\boldsymbol{\theta}) \end{aligned}$$

Combining this with (3.23) gives (3.21).

(2). One can prove the convexity of $\Phi(\boldsymbol{\theta})$ by directly calculating and checking the semi-definiteness of its Hessian matrix. However, a much simpler proof is based on the form (3.21), by noting that supremums over sets of linear functions are always convex [Boyd and

Vandenberghe, 2009]. In detail, for any $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$ and $\alpha \in [0,1]$, we use the form (3.21),

$$\begin{aligned}
&\Phi(\alpha\boldsymbol{\theta}_1 + (1-\alpha)\boldsymbol{\theta}_2) \\
&\quad = \max_{q\in\mathcal{P}_{\boldsymbol{x}}}\{\mathbb{E}_q(\alpha\theta_1(\boldsymbol{x}) + (1-\alpha)\theta_2(\boldsymbol{x})) + H(\boldsymbol{x};q)\} \\
&\quad \leq \alpha \max_{q\in\mathcal{P}_{\boldsymbol{x}}}\{\mathbb{E}_q(\theta_1(\boldsymbol{x})) + H(\boldsymbol{x};q)\} + (1-\alpha)\max_{q\in\mathcal{P}_{\boldsymbol{x}}}\{\mathbb{E}_q(\theta_2(\boldsymbol{x})) + H(\boldsymbol{x};q)\} \\
&\quad = \alpha\Phi(\boldsymbol{\theta}_1) + (1-\alpha)\Phi(\boldsymbol{\theta}_2).
\end{aligned}$$

(3). The derivatives are also implied immediately from (3.21):

$$\frac{\partial\Phi(\boldsymbol{\theta})}{\partial\theta_\alpha(\boldsymbol{x}_\alpha)} = q^*(\boldsymbol{x}_\alpha) = p(\boldsymbol{x}_\alpha).$$

This completes the proof. □

The variational form (3.21) requires optimizing over the set of all distributions $\mathcal{P}_x$; this is prohibitively high dimensional in practice, since specifying an arbitrary $q(\boldsymbol{x})$ requires $\prod_i |\mathcal{X}_i| - 1 = O(\exp(n))$ values. However, note that the expectation term $\mathbb{E}_q(\theta(\boldsymbol{x}))$ can be decomposed into terms defined only over each variable scope α:

$$\mathbb{E}_q(\theta(\boldsymbol{x})) = \sum_{\alpha\in\mathcal{I}} \mathbb{E}_q(\theta_\alpha(\boldsymbol{x}_\alpha)) = \sum_{\alpha\in\mathcal{I}}\sum_{\boldsymbol{x}_\alpha} q(\boldsymbol{x}_\alpha)\theta_\alpha(\boldsymbol{x}_\alpha),$$

which can be calculated if we know all the marginal distributions $\{q(\boldsymbol{x}_\alpha)\colon \alpha \in \mathcal{I}\}$. This motivates the definition of the marginal polytope.

Definition 3.7. *A marginal polytope $\mathbb{M}(\mathcal{I})$ is the set of all possible marginal distributions $\boldsymbol{\tau} := \{\tau_\alpha(\boldsymbol{x}_\alpha)\colon \alpha \in \mathcal{I}, \boldsymbol{x}_\alpha \in \mathcal{X}_\alpha\}$ that are consistent with (e.g., can be extended to) a valid*

joint distribution on $\boldsymbol{x}$*, that is,*

$$\mathbb{M}(\mathcal{I}) = \big\{\boldsymbol{\tau} : \exists \text{ distribution } q(\boldsymbol{x}) \in \mathcal{P}_{\boldsymbol{x}}, \text{ such that } \tau_\alpha(\boldsymbol{x}_\alpha) = \sum_{\boldsymbol{x}_{\neg\alpha}} q(\boldsymbol{x}) \text{ for } \forall \alpha \in \mathcal{I}, \boldsymbol{x}_\alpha \in \mathcal{X}_\alpha\big\}.$$

Proposition 3.2 (Maximum Entropy Principle)**.** *For each* $\boldsymbol{\tau} \in \mathbb{M}(\mathcal{I})$*, denote by* $\mathcal{P}[\boldsymbol{\tau}]$ *the set of distributions whose marginals are* $\boldsymbol{\tau}$*. Note that* $\mathcal{P}[\boldsymbol{\tau}]$ *is non-empty by definition. Then there exists an unique distribution with the maximum entropy in* $\mathcal{P}[\boldsymbol{\tau}]$*, that is* $q^* = \arg\max_{q \in \mathcal{P}[\boldsymbol{\tau}]} H(\boldsymbol{x}; q)$*, which has the exponential family form of* (3.20)*,*

$$q^*(\boldsymbol{x}) = \exp(\theta(\boldsymbol{x}) - \Phi(\boldsymbol{\theta})),$$

where $\boldsymbol{\theta}$ *is chosen such that* $q^*(\boldsymbol{x}) \in P[\boldsymbol{\tau}]$*.*

Proof. See [Jaynes, 1957]. □

This proposition builds a one-to-one correspondence between valid marginals $\boldsymbol{\tau} \in \mathbb{M}(\mathcal{I})$ and joint distributions with the exponential family form defined by the sets $\mathcal{I}$. With a slight abuse of notation, we use $\tau(\boldsymbol{x})$ to refer to the maximum entropy distribution in $\mathcal{P}[\boldsymbol{\tau}]$, and correspondingly, we denote by $H(\boldsymbol{x}; \boldsymbol{\tau})$ the entropy of the distribution $\tau(\boldsymbol{x})$. We can now immediately obtain an important result:

Proposition 3.3. $\Phi(\boldsymbol{\theta})$ *can be represented using the variational form,*

$$\Phi(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau} \in \mathbb{M}(\mathcal{I})} \{\langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + H(\boldsymbol{x}; \boldsymbol{\tau})\}, \tag{3.24}$$

where $\mathbb{M}(\mathcal{I})$ *is the marginal polytope, and* $\langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle$ *is the inner product,*

$$\langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle = \sum_{\alpha \in \mathcal{I}} \sum_{\boldsymbol{x}_\alpha} \tau_\alpha(\boldsymbol{x}_\alpha) \theta_\alpha(\boldsymbol{x}_\alpha).$$

In addition, the maximum is attained when $\boldsymbol{\tau}$ *is equal to the marginals of* $p(\boldsymbol{x}) = \exp(\theta(\boldsymbol{x}) - \Phi(\boldsymbol{\theta}))$*, that is,* $\boldsymbol{\tau}_\alpha(\boldsymbol{x}_\alpha) = p(\boldsymbol{x}_\alpha)$*,* $\forall \alpha \in \mathcal{I}, \forall \boldsymbol{x}_\alpha \in \mathcal{X}_\alpha$*.*

Proof. This is a direct result of Proposition 3.2 and 3.1. □

The form (3.24) is of fundamental importance for the development of variational inference methods: it transforms the sum inference task, calculating the log-partition function $\Phi(\theta)$ and the marginals $\{p(\boldsymbol{x}_\alpha) \colon \forall \alpha \in \mathcal{I}\}$ into a continuous optimization problem. However, it should be noted that the form itself does not actually decrease the complexity of the marginalization task, since (1) the marginal polytope $\mathbb{M}$ is difficult to characterize exactly (and may require an exponential number of linear constraints), and (2) the entropy $H(\boldsymbol{x}; \boldsymbol{\tau})$ is usually intractable to calculate exactly from the marginals $\boldsymbol{\tau}$.

Although exact calculation remains difficult, the form (3.24) provides a flexible framework for deriving a spectrum of powerful approximate inference algorithms (known as *variational inference*) by approximating the marginal polytope $\mathbb{M}(\mathcal{I})$ and the entropy term $H(\boldsymbol{x}; \boldsymbol{\tau})$ using various techinques. In the sequel, we first introduce the local consistency polytope for approximating the marginal polytope, and then various entropy approximation techniques that yield different variational message passing algorithms, including the loopy BP algorithm introduced in Section 3.1.3 via a Bethe entropy approximation, and its variant, tree reweighted BP, via convex entropy approximations.

3.2.2 Local Consistency Polytope

It is intractable to specify the marginal polytope $\mathbb{M}(G)$ on general graphs; the local consistency polytope provides a convenient approximation, and forms an important component for most belief-propagation-type methods. For notational simplicity, we first restrict our discussion on pairwise models, then discuss the more general case in Section 3.2.3.

A pairwise graphical model on an undirected graph $G = (V, E)$ can be written, in exponential family form, as

$$p(\boldsymbol{x}) \propto \exp(\sum_{i \in V} \theta_i(x_i) + \sum_{(ij) \in E} \theta_{ij}(x_i, x_j) - \Phi(\boldsymbol{\theta})).$$

We have $\mathcal{I} = V \cup E$ in this case, and the marginal polytope is defined in a vector space over only the singleton and pairwise marginals, that is,

$$\mathbb{M}(G) = \big\{\{\tau_i, \tau_{ij}\} \colon \exists\ q \in \mathcal{P}_{\boldsymbol{x}}, \text{ s.t. } \tau_i(x_i) = q(x_i),\ \tau_{ij}(x_i, x_j) = q(x_i, x_j),\ \forall i \in V, (ij) \in E\big\}.$$

Note that for any $\boldsymbol{\tau} \in \mathbb{M}(G)$, it is obvious that $\sum_{x_j} \tau_{ij}(x_i, x_j) = \tau_i(x_i)$, that is, the pairwise and singleton marginals will be consistent with each other. This motivates the definition of the local consistency polytope,

$$\mathbb{L}(G) = \big\{\{\tau_i, \tau_{ij}\} \colon \sum_{x_j} \tau_{ij}(x_i, x_j) = \tau_i(x_i),\ \ \sum_{x_i} \tau_i(x_i) = 1,\ \tau_i(x_i, x_j) \geq 0,\ \forall i \in V, (ij) \in E\big\}.$$

Obviously, we have $\mathbb{M}(G) \subseteq \mathbb{L}(G)$, that is, the marginal polytope is a subset of the local consistency polytope. In addition, $\mathbb{M}(G)$ and $\mathbb{L}(G)$ are equal when G is a tree.

Proposition 3.4. *If G is a tree structured graph, we have $\mathbb{M}(G) = \mathbb{L}(G)$.*

Proof. We already know that $\mathbb{M}(G) \subseteq \mathbb{L}(G)$, and just need to prove that $\mathbb{L}(G) \subseteq \mathbb{M}(G)$. To this end, we assign a tree order on G, such that the (unique) parent of node i is $\mathrm{pa}(i)$. Then for any $\{\tau_i, \tau_{ij}\} \in \mathbb{L}(G)$, we can construct a valid joint distribution by the chain rule, with

$$q(\boldsymbol{x}) \stackrel{def}{=} \prod_{i \in V} q(x_i | x_{\mathrm{pa}(i)}), \qquad \text{where} \quad q(x_i | x_{\mathrm{pa}(i)}) = \frac{\tau_{ij}(x_i, x_{\mathrm{pa}(i)})}{\tau(x_{\mathrm{pa}(i)})},$$

where for the root node i_0 with empty parents ($\mathrm{pa}(i_0) = \emptyset$), we set $q(x_{i_0} | x_{\mathrm{pa}(i_0)}) = q(x_{i_0}) = \tau_{i_0}(x_{i_0})$. Then it is straightforward to verify that $\{\tau_i, \tau_{ij}\}$ are the marginals of $q(\boldsymbol{x})$. □

In contrast, when G has cycles, there can be locally consistent marginals in $\mathbb{L}(G)$ for which no consistent joint distribution exists; this is not the case for $\mathbb{M}(G)$. See the following example:

Example 3.2. *Consider the following marginals* $\boldsymbol{\tau}$ *on* $[x_1, x_2, x_3] \in \{0,1\}^3$*,*

$$\tau_1 = \tau_2 = \tau_3 = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} \qquad \tau_{12} = \tau_{23} = \tau_{13} = \begin{bmatrix} 0 & 1/2 \\ 1/2 & 0 \end{bmatrix}.$$

Obviously the marginals are locally consistent with each other (i.e., $\boldsymbol{\tau} \in \mathbb{L}(G)$*). However, there exists no joint distribution* $[x_1, x_2, x_3]$ *whose marginals are* $\{\tau_i, \tau_{ij}\}$ *(i.e.,* $\boldsymbol{\tau} \notin \mathbb{M}(G)$*). To see this, note that each pairwise marginal* τ_{ij} *implies that the configurations with non-zero probability should satisfy* $x_i \neq x_j$*; however, their exists no configuration in* $\{0,1\}^3$ *that satisfies* $x_1 \neq x_2$*,* $x_2 \neq x_3$ *and* $x_1 \neq x_3$ *simultaneously.*

These marginals correspond to a frustrated cycle, in which the pairs (x_i, x_j) *are all anti-correlated (in fact, constrained to be different); looking at each pair locally without considering the value of the third variable, it appears to be possible to satisfy the constraints, but satisfying all three is impossible. Such frustrated cycles are often difficult for loopy BP and its local polytope approximation* $\mathbb{L}(G)$*, and illustrates one way in which the approximation can fail.*

3.2.3 Loopy Belief Propagation

An efficient approximation of the variational form (3.24) requires three components: (1) approximating the marginal polytope; (2) approximating the entropy $H(\boldsymbol{x}; \boldsymbol{\tau})$; and (3) solving the continuous optimization with those approximations. We introduced the local consistency polytope to approximate the marginal polytope; next, we introduce a Bethe entropy approximation, which together with a Lagrange multiplier optimization method, derive the loopy belief propagation algorithm introduced in Section 3.1.3. We first restrict discussion to pairwise models, then give extensions to more general factor and junction graphs.

Bethe Entropy Approximation

The exact joint entropy $H(\boldsymbol{x};\boldsymbol{\tau})$ is in general intractable to calculate directly; the Bethe entropy approximates $H(\boldsymbol{x};\boldsymbol{\tau})$ using a linear combination of entropies on the singleton and pairwise marginals. To motivate it, we begin by discussing the case when G is a tree, on which the joint entropy exactly decomposes to a combination of singleton and pairwise entropies.

Proposition 3.5. *For any distribution $\tau(\boldsymbol{x})$ on tree-structured graph G, whose marginals on single and pairs of variables are $\{\tau_i, \tau_{ij}\}$, we have*

$$H(\boldsymbol{x};\tau) = \sum_{i\in V} H_i(\tau_i) - \sum_{(ij)\in E} I_{ij}(\tau_{ij}), \tag{3.25}$$

where H_i and I_{ij} are the entropy and multual information on the singleton and pairwise marginals, respectively, i.e.,

$$H_i(\tau_i) = -\sum_{x_i} \tau_i(x_i)\log\tau_i(x_i), \qquad I_{ij}(\tau_{ij}) = \sum_{x_i,x_j} \tau_{ij}(x_i,x_j)\log\frac{\tau_{ij}(x_i,x_j)}{\tau_i(x_i)\tau_j(x_j)}.$$

Proof. Just note that when G is a tree, $\tau(\boldsymbol{x})$ can be written as

$$\tau(\boldsymbol{x}) = \prod_{i\in V}\tau_i(x_i)\prod_{(ij)\in E}\frac{\tau_{ij}(x_i,x_j)}{\tau_i(x_i)\tau_j(x_j)}.$$

Substituting this into the entropy, we get

$$\begin{aligned} H(\boldsymbol{x};\tau) = -\mathbb{E}_\tau\left[\log\tau(\boldsymbol{x})\right] &= -\mathbb{E}_\tau\left[\sum_{i\in V}\log\tau_i(x_i) + \sum_{(ij)\in E}\log\frac{\tau_{ij}(x_i,x_j)}{\tau_i(x_i)\tau_j(x_j)}\right] \\ &= \sum_{i\in V} H_i(\tau_i) - \sum_{(ij)\in E} I_{ij}(\tau_{ij}). \end{aligned}$$

This finishes the proof. □

The pairwise decomposition (3.25) no longer holds exactly in general, e.g., when the graph G is loopy. However, it can be used to define a convenient approximation on loopy graphs,

$$H(\boldsymbol{x};\boldsymbol{\tau}) \approx H^{bethe}(\boldsymbol{\tau}) \stackrel{def}{=} \sum_{i\in V} H_i(\tau_i) - \sum_{(ij)\in E} I_{ij}(\tau_{ij}),$$

which is known as the *Bethe entropy approximation*.

Remarks. (1). Although the exact entropy $-H(\boldsymbol{x};\boldsymbol{\tau})$ is a convex function of $\boldsymbol{\tau}$, the Bethe approximation $-H^{bethe}(\boldsymbol{\tau})$ is generally not guaranteed to be convex; convexified variants are discussed later in Section 3.2.4.

(2). Note that $H^{bethe}(\boldsymbol{x})$ is well-defined on any marginals $\boldsymbol{\tau}$ in $\mathbb{L}(G)$, even when $\boldsymbol{\tau} \notin \mathbb{M}(G)$ and an exact joint entropy is not defined.

Example 3.3. *For the marginals $\boldsymbol{\tau} = \{\tau_i, \tau_{ij}\}$ defined in Example 3.2, we have*

$$H_i(\tau_i) = \log 2, \qquad H_{ij}(\tau_{ij}) = \log 2, \qquad I_{ij}(\tau_{ij}) = H_i(\tau_i) + H_j(\tau_j) - H_{ij}(\tau_{ij}) = \log 2.$$

We can calculate the Bethe entropy,

$$H^{bethe}(\boldsymbol{\tau}) = H_1(\tau_1) + H_2(\tau_2) + H_3(\tau_3) - I_{12}(\tau_{12}) - I_{23}(\tau_{23}) - I_{13}(\tau_{13}) = 0.$$

On the other hand, the joint entropy itself is not defined in this case, since $\boldsymbol{\tau} \notin \mathbb{M}(G)$.

Algorithm 3.10 Loopy belief propagation on pairwise models

Input: Pairwise graphical model $p(\boldsymbol{x}) \propto \prod_{i \in V} \psi_i(x_i) \prod_{(ij) \in E} \psi_{ij}(x_i, x_j)$ on undirected graph $G = (V, E)$. Let $\partial(i)$ be the neighborhood of $i \in V$.

Output: Approximate marginals $\{\tau_i, \tau_{ij} \colon i \in V, (ij) \in E\}$.

1. Pass messages between the nodes until convergence:

$$m_{i \to j}(x_j) \propto \sum_{x_i} \psi_{ij}(x_i, x_j)\ \psi(x_i) \prod_{i' \in \partial(i) \setminus \{j\}} m_{i' \to i}(x_i). \tag{3.27}$$

2. Calculate the approximate marginals,

$$\tau_i(x_i) \propto \psi_i(x_i) \prod_{i' \in \partial(i)} m_{i' \to i}(x_i), \tag{3.28}$$

$$\tau_{ij}(x_i, x_j) \propto \psi_{ij}(x_i, x_j) \psi_i(x_i) \psi_j(x_j) \prod_{i' \in \partial(i) \setminus \{j\}} m_{i' \to i}(x_i) \prod_{j' \in \partial(j) \setminus \{i\}} m_{j' \to j}(x_j), \tag{3.29}$$

and the approximate log-partition function, $\Phi(\theta) \approx \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + \sum_{i \in V} H_i(\tau_i) - \sum_{(ij) \in E} I_{ij}(\tau_{ij})$.

Loopy BP via Lagrangian Multiplier Method

Using the local consistency polytope and the Bethe entropy approximation, the variational form (3.24) can be approximated by

$$\max_{\boldsymbol{\tau} \in \mathbb{L}(G)} \big\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + \sum_{i \in V} H_i(\tau_i) - \sum_{(ij) \in E} I_{ij}(\tau_{ij}) \big\}. \tag{3.26}$$

This optimization is not convex in general. However, it is possible to find its local optima via fixed point updates. The following result by Yedidia et al. [2005] shows that the stationary points of (3.26) are fixed points of the loopy belief propagation algorithm introduced in Section 3.1.3, which we repeat here in Algorithm (3.10), with $\psi_i = \exp(\theta_i)$ and $\psi_{ij} = \exp(\theta_{ij})$.

Theorem 3.1. *A set of strictly positive marginals $\{\tau_i, \tau_{ij}\}$ is a stationary point of (3.26) if and only if there exists a set of positive functions (called "messages") $\{m_{i \to j}(\boldsymbol{x}_i) \colon (ij) \in E\}$, which satisfy (3.28)-(3.29) and is a fixed point of (3.27).*

Proof. We start by writing the Lagrangian of (3.26) w.r.t. the local consistency constraint $\sum_{x_i} \tau_{ij}(x_i, x_j) = \tau_j(x_j)$, and the normalization constraints $\sum_{x_i} \tau_i(x_i) = 1$,

$$\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{i \in V} [H_i(\tau_i) + \lambda_i^0 \sum_{x_i} \tau_i(x_i)] - \sum_{(ij) \in E} [I_{ij}(\tau_{ij}) + \sum_{x_j} \lambda_{i \to j}(x_j) \sum_{x_i} (\tau_{ij}(x_i, x_j) - \tau_j(x_j))],$$

where $\{\lambda_{j \to i}(x_i) \colon (ij) \in E, x_i \in \mathcal{X}_i\}$ and $\{\lambda_i^0 \colon i \in V\}$ are the Lagrange multipliers; because we assume $\{\tau_i, \tau_{ij}\}$ are strictly positive, the inequality constraints $\tau_{ij}(x_i, x_j) \geq 0$ are inactive and not included (see Yedidia et al. [2005] for more discussion). Recall:

$$\begin{aligned}
\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle &= \sum_{i \in V} \theta_i(x_i) \tau_i(x_i) + \sum_{(ij) \in E} \theta_{ij}(x_i, x_j) \tau_{ij}(x_i, x_j), \\
H_i(\tau_i) &= -\sum_{x_i} \tau_i(x_i) \log \tau_i(x_i), \\
I_{ij}(\tau_{ij}) &= \sum_{x_i, x_j} \tau_{ij}(x_i, x_j) \log \frac{\tau_{ij}(x_i, x_j)}{\sum_{x_i} \tau_{ij}(x_i, x_j) \sum_{x_j} \tau_{ij}(x_i, x_j)}.
\end{aligned}$$

Taking the derivative of the Lagrangian w.r.t. $\tau_i(x_i)$ and $\tau_{ij}(x_i, x_j)$ gives

$$\theta_i(x_i) - \log \tau_i(x_i) + \sum_{j \in \partial(i)} \lambda_{j \to i}(x_i) = const, \tag{3.30}$$

$$\theta_{ij}(x_i, x_j) - \log \frac{\tau_{ij}(x_i, x_j)}{\tau_i(x_i) \tau_j(x_j)} - \lambda_{i \to j}(x_j) - \lambda_{j \to i}(x_i) = const, \tag{3.31}$$

where we used the local consistency condition that $\sum_{x_j} \tau_{ij}(x_i, x_j) = \tau_i(x_i)$ in (3.31).

Let us define $m_{i \to j}(x_j) = \exp(\lambda_{i \to j}(x_j))$; then (3.30) and (3.31) can be rewritten,

$$\tau_i(x_i) \propto \psi_i(x_i) \prod_{i' \in \partial(i)} m_{i' \to i}(x_i), \tag{3.32}$$

$$\tau_{ij}(x_i, x_j) \propto \frac{\psi_{ij}(x_i, x_j)}{m_{i \to j} m_{j \to i}} \tau_i(x_i) \tau_j(x_j). \tag{3.33}$$

This proves (3.28). Substituting (3.32) and its equivalent on $\tau_j(x_j)$ into (3.33) gives (3.29).

To derive the message update (3.27), we plug (3.32)-(3.33) into the consistency constraint $\tau_j(x_j) = \sum_{x_i} \tau_{ij}(x_i, x_j)$; this gives

$$\tau_j(x_j) \propto \sum_{x_i} \frac{\psi_{ij}(x_i, x_j)}{m_{i\to j} m_{j\to i}} \tau_i(x_i) \tau_j(x_j).$$

Canceling $\tau_j(x_j)$ on both sides (assuming it is non-zero), and moving $m_{i\to j}$ to the left side, we get the message update (3.27):

$$m_{i\to j} \quad \propto \quad \sum_{x_i} \psi_{ij}(x_i, x_j) \frac{\tau_i(x_i)}{m_{j\to i}} \quad \propto \quad \sum_{x_i} \psi_{ij}(x_i, x_j)\ \psi(x_i) \prod_{i' \in \partial(i) \setminus \{j\}} m_{i'\to i}(x_i),$$

where we used (3.32). This proves that any stationary point of (3.26) is a fixed point of Algorithm 3.10. Similarly, it is easy to show that any fixed point of Algorithm 3.10 should satisfy the stationary conditions in (3.30)-(3.31). This completes the proof. □

Therefore, we can solve the optimization (3.26) by performing the message update (3.27) iteratively on the graph, and then decoding the marginals via (3.28)-(3.29) when the messages converge; this gives the loopy belief propagation in Algorithm 3.10. In addition, we can also estimate the log-partition function based on the decoded marginals via

$$\Phi(\theta) \approx \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + \sum_{i \in V} H_i(\tau_i) - \sum_{(ij) \in E} I_{ij}(\tau_{ij}).$$

Note that Algorithm 3.10 reduces to an exact variable elimination algorithm (and terminates within finite steps) when applied on trees. In more general, loopy graphs, one is required to iteratively update the messages until convergence (e.g., until some stopping criterion). However, loopy BP is generally not guaranteed to converge; see Ihler et al. [2005] for discussions on convergence conditions.

Loopy BP was originally proposed (and is easily seen as) a heuristic that simply transforms

Algorithm 3.11 Sum-product factor graph belief propagation

Input: Factor graphical model $p(\boldsymbol{x}) \propto \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha)$. Let $\partial(i)$ be the set of factors that include i.

Output: Approximate marginals $\{\tau_i, \tau_\alpha\}$.

1. Pass messages between the factors and variables until convergence:

$$\text{Variables to Factors:} \quad m_{i \to \alpha}(x_i) \propto \prod_{\alpha' \in \partial(i) \setminus \{\alpha\}} m_{\alpha' \to i}(x_i),$$

$$\text{Factors to Variables:} \quad m_{\alpha \to i}(x_i) \propto \sum_{\boldsymbol{x}_{\alpha \setminus \{i\}}} \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{i' \in \alpha \setminus \{i\}} m_{i' \to \alpha}(x_{i'}).$$

2. Calculate the approximate marginals:

$$\tau_i(x_i) \propto \prod_{\alpha \in \partial(i)} m_{\alpha \to i}(x_i), \qquad \tau_\alpha(\boldsymbol{x}_\alpha) \propto \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{i \in \alpha} m_{i \to \alpha}(x_i).$$

the steps of variable elimination into *iterative* updates applicable to *loopy* graphs [Pearl, 1988]. However, the variational form discovered by Yedidia et al. [2005] provides a more principled interpretation, and more importantly, motivates many new methods that can improve on loopy BP. For example, beyond the basic fixed point algorithm corresponding to loopy BP, many potentially more efficient optimization algorithms, particularly ones with convergence guarantees, can be used to directly solve the variational optimization (3.26) [see e.g., Welling and Teh, 2001, Yuille, 2002]. It is also possible to modify or extend the variational objective, either by exploiting higher order cliques (such as generalized BP [Yedidia et al., 2005]), or by using different entropy approximations to ensure that the variational optimization is convex (see Wainwright and Jordan [2008] and reference therein).

Loopy BP on Factor Graphs and Junction Graphs

In the preceding sections, we restricted the discussion to pairwise models and pairwise entropy approximations. There are many methods to extend loopy BP to handle and exploit

higher order factors, including junction graph BP [Mateescu et al., 2010], factor graph BP [Kschischang et al., 2001], and generalized BP [Yedidia et al., 2005]. In this section, we briefly introduce junction graph BP and factor graph BP, and their associated Bethe-like entropy approximations.

Factor Graph BP. Consider a factorized distribution $p(\boldsymbol{x}) \propto \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha)$. Its factor graph is a bipartite graph $(V, \mathcal{I}, E)$ consisting of variable nodes $V = \{i\}$, factor nodes $\mathcal{I} = \{\alpha\}$, and edges between factors and their associated variables $E = \{(i, \alpha) \in V \times \mathcal{I} \colon i \in \alpha\}$. We denote by $\partial(i)$ the set of factors that includes node i, that is, $\partial(i) = \{\alpha \in \mathcal{I} \colon i \in \alpha\}$.

To approximate the variational form using a factor graph, we first approximate the marginal polytope with a local consistency polytope on $\boldsymbol{\tau} = \{\tau_i, \tau_\alpha \colon i \in V, \alpha \in \mathcal{I}\}$,

$$\mathbb{L}(V \cup \mathcal{I}) = \{\boldsymbol{\tau} \colon \sum_{\boldsymbol{x}_{\alpha \setminus \{i\}}} \tau_\alpha(\boldsymbol{x}_\alpha) = \tau_i(x_i), \quad \sum_{x_i} \tau_i(x_i) = 1, \;\; \tau_\alpha(\boldsymbol{x}_\alpha) \geq 0, \;\; \forall i \in V, \; \alpha \in \mathcal{I}, \; \boldsymbol{x} \in \mathcal{X}\},$$

and then approximate the entropy via

$$H(\boldsymbol{x}; \boldsymbol{\tau}) \;\approx\; \sum_{\alpha \in \mathcal{I}} H(\boldsymbol{x}_\alpha; \tau_\alpha) - \sum_{i \in V} (|\partial(i)| - 1) H(x_i; \tau_i).$$

For pairwise factors, this entropy form is equivalent to the Bethe entropy defined in the previous section. Using this polytope and entropy approximation, we can derive factor graph BP, shown in Algorithm 3.11, using a Lagrange multiplier method similar to that used in Theorem 3.1 [Yedidia et al., 2005]. Note that the *factor-to-variable* and *variable-to-factor* messages are given different update rules in factor graph BP.

Junction Graph BP. Assume $(\mathcal{G}, \mathcal{C}, \mathcal{S})$ is a junction graph with clusters $\mathcal{C}$ and separators $\mathcal{S}$. To approximate the variational form (3.24), we first replace the marginal polytope with a higher order local consistency polytope $\mathbb{L}(\mathcal{C})$, which is the set of local marginals

Algorithm 3.12 Sum-product junction graph belief propagation

Input: Graphical model $p(\boldsymbol{x}) \propto \prod_{c_k \in \mathcal{C}} \psi_{c_k}(\boldsymbol{x}_{c_k})$ and its junction graph $(\mathcal{G}, \mathcal{C}, \mathcal{E})$ with clusters $\mathcal{C}$ and separators $\mathcal{S}$.

Output: Approximate marginals $\{\tau_{c_k}, \tau_{s_{kl}} : c_k \in \mathcal{C}, s_{kl} \in \mathcal{S}\}$.

1. Pass messages between clusters on the junction graph until convergence:

$$m_{k \to l}(\boldsymbol{x}_{s_{kl}}) \propto \sum_{\boldsymbol{x}_{c_k \setminus s_{kl}}} \psi_{c_k}(\boldsymbol{x}_{c_k}) \prod_{k' \in \partial(k) \setminus \{l\}} m_{k' \to k}(\boldsymbol{x}_{s_{k'k}}), \tag{3.34}$$

where $\partial(k)$ is the neighborhoood of cluster c_k.

2. Calculate the approximate marginal distributions:

$$\tau_{c_k}(\boldsymbol{x}_{c_k}) \propto \psi_{c_k}(\boldsymbol{x}_{c_k}) \prod_{k' \in \partial(k)} m_{k' \to k}(\boldsymbol{x}_{s_{k'k}}), \qquad \text{for all } c_k \in \mathcal{C}, \tag{3.35}$$

$$\tau_{s_{kl}}(\boldsymbol{x}_{s_{kl}}) \propto m_{k \to l}(\boldsymbol{x}_{s_{kl}}) m_{l \to k}(\boldsymbol{x}_{s_{kl}}), \qquad \text{for all } s_{kl} \in \mathcal{S}. \tag{3.36}$$

$\boldsymbol{\tau} = \{\tau_{c_k}, \tau_{s_{kl}} : c_k \in \mathcal{C}, s_{kl} \in \mathcal{S}\}$ that are consistent on the intersections of the clusters and separators, that is,

$$\mathbb{L}(\mathcal{C}) = \Big\{ \boldsymbol{\tau} : \sum_{x_{c_k \setminus s_{kl}}} \tau_{c_k}(x_{c_k}) = \tau(x_{s_{kl}}), \tau_{c_k}(x_{c_k}) \geq 0, \text{for } \forall\ c_k \in \mathcal{C}, s_{kl} \in \mathcal{E} \Big\}.$$

We then approximate the variational form as

$$\max_{\boldsymbol{\tau} \in \mathbb{L}(\mathcal{C})} \Big\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{k \in \mathcal{V}} H(\boldsymbol{x}_{c_k}; \boldsymbol{\tau}_{c_k}) - \sum_{(kl) \in \mathcal{E}} H(\boldsymbol{x}_{s_{kl}}; \boldsymbol{\tau}_{s_{kl}}) \Big\}, \tag{3.37}$$

where the joint entropy is approximated by a linear combination of the entropies of local marginals on the clusters $H(\boldsymbol{x}_{c_k}; \boldsymbol{\tau}_{c_k})$ and separators $H(\boldsymbol{x}_{s_{kl}}; \boldsymbol{\tau}_{s_{kl}})$. Again, it is straightforward to show that this form is equivalent to the Bethe approximations on pairwise and factor graphs, for an appropriately chosen junction graph. The approximate objective can again maximized using the method of Lagrange multipliers [Yedidia et al., 2005], leading to the sum-product belief propagation (BP) algorithm shown in Algorithm 3.12, which iteratively updates a set of "messages" $\{m_{k \to l}(\boldsymbol{x}_{s_{kl}})\}$ between the neighboring clusters.

3.2.4 Convex Variational Methods

The Bethe entropy approximation is a non-concave function of $\boldsymbol{\tau}$, causing (3.26) to be a non-convex optimization. A large spectrum of methods have been proposed to define concave entropy approximations, mostly by using more general counting numbers, or coefficients in the linear combination of local entropies. In the pairwise case, these approximations have the following form:

$$H^{conv}(\boldsymbol{\tau}) = \sum_{i \in V} H_i(\tau_i) - \sum_{(ij) \in E} \rho_{ij} I_{ij}(\tau_{ij}), \tag{3.38}$$

where $\{\rho_{ij} \colon (ij) \in E\}$ is a set of properly chosen weights that make the objective a concave function. Heskes [2006] proposed a set of sufficient conditions on $\{\rho_{ij}\}$ to ensure concavity. The idea is to make sure that (3.38) can be rewritten into a form,

$$\sum_{i \in V} \kappa_i H(x_i) + \sum_{(ij) \in E} \kappa_{i \to j} H(x_i | x_j) + \kappa_{ij} H(x_i, x_j), \tag{3.39}$$

where κ_{ij}, $\kappa_{i \to j}$, κ_i are all non-negative numbers and $H(x_i|x_j) := H(x_i, x_j) - H(x_j)$ is the conditional entropy. Equation (3.39) is guaranteed to be concave, because both the marginal and conditional entropies are all concave functions. This motivates the following definition [Heskes, 2006, Weiss et al., 2007],

Definition 3.8. *A set of weights* $\{\rho_{ij}\}$ *in* (3.38) *is said to be* provably convex*, if there exist a set of non-negative numbers* $\kappa_{i \to j}$ *such that* $\sum_{i' \in \partial(i)} \kappa_{i' \to i} \geq (|\partial(i)| - 1)$ *and* $\kappa_{i \to j} + \kappa_{j \to i} \leq \rho_{ij}$.

Obviously, a set of provably convex weights $\{\rho_{ij}\}$ gives rise to a provably concave entropy approximation.

Tree Reweighted Belief Propagation

Wainwright et al. [2005] introduced an important subclass of provably concave entropy approximations that have the additional property of giving upper bounds for the exact entropy[2]. Their idea is based on breaking the loopy graph into combinations of spanning trees. In detail, let $\mathcal{T}$ be the set of spanning trees of G, and $\{w^T : T \in T\}$ a set of non-negative weights assigned on $\mathcal{T}$, such that $\sum_{T\in\mathcal{T}} w^T = 1$. We define ρ_{ij} to be the sum of weights of the spanning trees that include (ij) as an edge, that is,

$$\rho_{ij} = \sum_{T:\ (ij)\in E^T} w^T.$$

The quantities ρ_{ij} are called edge appearance probabilities. Then (3.38) can be rewritten into a convex combination of the Bethe entropies on the trees,

$$H^{trw}(\boldsymbol{\tau}) = \sum_{T\in\mathcal{T}} w^T H^{bethe}(\boldsymbol{\tau}; T), \qquad H^{bethe}(\boldsymbol{\tau}; T) = \sum_{i\in V} H_i(\tau_i) - \sum_{(ij)\in T} I_{ij}(\tau_{ij}),$$

where each $H^{bethe}(\boldsymbol{\tau}; T)$ is the Bethe entropy restricted to the spanning tree T. Wainwright and Jordan [2008] showed the following properties of $H^{bethe}(\boldsymbol{\tau}; T)$:

1. $H^{bethe}(\boldsymbol{\tau}; T)$ is a concave function of $\boldsymbol{\tau} \in \mathbb{L}(G)$ (by rewriting it into (3.39) via the chain rule).
2. If $\boldsymbol{\tau} \in \mathbb{M}(G)$, then $H^{bethe}(\boldsymbol{\tau}; T) \geq H(\boldsymbol{x}; \boldsymbol{\tau})$.

Therefore, $H^{trw}(\boldsymbol{\tau})$ is also a concave function and an upper bound of the exact joint entropy. It is then natural to approximate the variational optimization via

$$\Phi_{trw}(\{\rho_{ij}\}) = \max_{\boldsymbol{\tau}\in\mathbb{L}(G)} \big\{\langle \boldsymbol{\theta}, \boldsymbol{\tau}\rangle + \sum_{i\in V} H_i(\tau_i) - \sum_{(ij)\in E} \rho_{ij} I_{ij}(\tau_{ij})\big\}. \tag{3.40}$$

[2]There exist other provably convex weights that are not TRW weights, as well; for example, the Bethe weights on a single cycle are probably convex but are not valid TRW weights; see Weiss et al. [2007] for a discussion

This gives an upper bound on the log-partition function, that is, $\Phi_{trw}(\{\rho_{ij}\}) \geq \Phi(\boldsymbol{\theta})$, because it maximizes an upper bound on a larger set compared to the exact variational optimization (3.24). Since (3.40) is a convex optimization over $\boldsymbol{\tau}$, it can be solved efficiently using many optimization methods, including tree reweighted BP, proposed in Wainwright et al. [2005], which iteratively passes messages that are similar to loopy BP, but uses powered sums in the message updates.

In addition, one can search for the optimal $\{\rho_{ij}\}$ to minimize $\Phi_{trw}(\{\rho_{ij}\})$ and obtain the tightest possible upper bound. Wainwright et al. [2005] performs this minimization using a double-loop conditional gradient descent, with an inner loop that optimizes $\boldsymbol{\tau}$ to solve (3.40) with fixed $\{\rho_{ij}\}$ via tree reweighted BP, and an outer loop that takes conditional gradient descent updates on $\{\rho_{ij}\}$ (which, very elegantly, corresponds to solving a maximum spanning tree problem in this case). Practical experiments have shown that optimized weights $\{\rho_{ij}\}$ can significantly outperform naïve weight choices, such as using uniform or randomly chosen weights. Unfortunately, the conditional gradient descent procedure is very slow due to the expensive inner loops. In Chapter 5, we introduce a new class of related upper bounds on which we can update the weights more efficiently with single-loop algorithms.

Tree Reweighted BP via Jensen's Inequality

In addition to its interpretation as a convex entropy approximation, tree reweighted BP can also be nicely interpreted as directly building an upper bound for the log-partition function $\Phi(\boldsymbol{\theta})$ via Jensen's inequality.

In detail, let us denote by $\{\boldsymbol{\theta}^T : T \in \mathcal{T}\}$ a set of natural parameters such that (1) the Markov network of $p(\boldsymbol{x}|\boldsymbol{\theta}^T) := \exp(\theta^T(\boldsymbol{x}) - \Phi(\boldsymbol{\theta}^T))$ is the spanning tree T, and (2) $\sum_{\in T \in \mathcal{T}} \boldsymbol{\theta}^T = \boldsymbol{\theta}$. In this way, the original $\boldsymbol{\theta}$ is decomposed into a combination of sub-models defined on the

spanning trees. Because $\Phi(\boldsymbol{\theta})$ is a convex funciton of $\boldsymbol{\theta}$, by Jensen's inequality, we have

$$\Phi(\boldsymbol{\theta}) \;=\; \Phi\Big(\sum_{T\in\mathcal{T}} w^T \frac{\boldsymbol{\theta}^T}{w^T}\Big) \;\leq\; \sum_{T\in\mathcal{T}} w^T \Phi\Big(\frac{\boldsymbol{\theta}^T}{w^T}\Big) \;\stackrel{def}{=}\; \Psi(\{\boldsymbol{\theta}^T\},\{w^T\}).$$

where $\Psi(\{\boldsymbol{\theta}^T\},\{w^T\})$ is both convex and an upper bound of $\Phi(\boldsymbol{\theta})$. The choice of the spanning tree parameters $\{\boldsymbol{\theta}^T\}$ and their weights $\{w^T\}$ influences the tightness of the bound, and one can find the optimal ($\{\boldsymbol{\theta}^T\}$ and $\{w^T\}$ to give the tightest upper bound. Wainwright et al. [2005] showed that the TRW variational optimization in (3.40) is the dual problem of optimizing $\{\theta^T\}$ with fixed weights $\{w^T\}$, that is,

$$\min_{\{\boldsymbol{\theta}^T\}} \Psi(\{\boldsymbol{\theta}^T\},\{w^T\}) = \max_{\boldsymbol{\tau}\in\mathbb{L}(G)} \big\{\langle\boldsymbol{\theta},\boldsymbol{\tau}\rangle + \sum_{i\in V} H_i(\tau_i) - \sum_{(ij)\in E} \rho_{ij} I_{ij}(\tau_{ij})\big\}.$$

These two forms of the TRW bound have their own pros and cons: although the variational form on the right side (which we call the dual form) is guaranteed to be an upper bound only when τ is fully optimized, the primal bound $\Psi(\{\boldsymbol{\theta}^T\},\{w^T\})$ on the left side provides a provable bound for any $\{\theta^T\}$ and $\{w^T\}$. On the other hand, it is computationally intractable to fully optimize the primal bound $\Psi(\{\boldsymbol{\theta}^T\},\{w^T\})$ directly, since the number of spanning trees may be extremely large. For efficiency, some approaches restrict to a small subset of trees [e.g., Jancsary and Matz, 2011], but if too few trees are included, the quality of the bound will suffer. Ideally, we would like to combine the advantages of both the primal and variational bounds; this is possible under a new method we propose in Chapter 5.

Conditional Entropy Decomposition

The standard form of TRBP is defined on pairwise models; generalized TRBP [Wainwright et al., 2005, Wiegerinck, 2005] extents TRBP to use combinations of junction trees and works for models with higher order cliques. Additionally, conditional entropy decomposition (CED) [Globerson and Jaakkola, 2007a] uses combinations of even more general (possibly

un-triangulated) ordered graphs. This section gives a brief introduction to CED.

Consider a set of ordered graphs $\mathcal{G}_{\text{ced}} = \{(G^r, o^r) \colon r = 1, \dots, R\}$, where $G^r = (V, E^r)$ is a sub-graph of the triangulation of G along order o^r, and $o^r = [o_1^r, \dots, o_n^r]$ is an total ordering over nodes V. We assign a set of weights $\boldsymbol{w} = \{w^1, \dots, w^R\}$ on $\mathcal{G}_{\text{ced}}$ such that $\sum_{r=1}^R w^r = 1$, and $w^r \geq 0$. Let $\text{pa}_r(i)$ be the parent set of i on G^r under ordering o^r. It is noted in Globerson and Jaakkola [2007a] that

$$H(\boldsymbol{x}; \boldsymbol{\tau}) = \sum_{i=1}^N H(x_{o_i^r} | \boldsymbol{x}_{o_{i+1:n}^r}; \boldsymbol{\tau}) \leq \sum_{i=1}^N H(x_{o_i^r} | \boldsymbol{x}_{\text{pa}_r(o_i^r)}; \boldsymbol{\tau}), \tag{3.41}$$

where the first equality holds by the entropic chain rule, and the second inequality holds because $o_{i+1:n}^r \subseteq \text{pa}_r(o_i^r)$, and the entropy can only increase when conditioning on fewer variables; see, e.g., Cover and Thomas [2006].

We can also construct a local consistency polytope based on $\mathcal{G}_{\text{ced}}$. Let $\mathcal{I}_{\text{ced}} = \{\text{pa}_r(o_i^r) \cup \{o_i^r\} \colon o_i^r \in V, r = 1, \dots, R\}$ be the cliques defined by the graphs G^r and orders o^r, and define a local consistency polytope on the marginals $\boldsymbol{\tau} = \{\tau_c \colon c \in \mathcal{I}(\mathcal{G}_{\text{ced}})\}$,

$$\mathbb{L}(\mathcal{G}_{\text{ced}}) = \big\{\boldsymbol{\tau} \geq 0 \colon \sum_{x_c} \tau_c(x_c) = 1, \sum_{x_{c \cap t}} \tau_c(x_c) = \sum_{x_{c \cap t}} \tau_t(x_t), \, \forall c, t \in \mathcal{I}(\mathcal{G}_{\text{ced}})\big\}.$$

Clearly, $\mathbb{M} \subseteq \mathbb{L}(\mathcal{G}_{\text{ced}})$. This gives an variational upper bound for the log-partition function,

$$\Phi(\boldsymbol{\theta}) \leq \max_{\tau \in \mathbb{L}(\mathcal{G}_{\text{ced}})} \big\{\langle \theta, \tau \rangle + \sum_{r=1}^R w^r \sum_{i=1}^n H(x_{o_i^r} | x_{\text{pa}_r(o_i^r)})\big\}, \tag{3.42}$$

where the inequality holds because the objective function is upper bounded and the optimization's constraint set is relaxed.

It is easy to see that the CED bound (3.42) generalizes that of TRBP and generalized TRBP: when G_r in $\mathcal{G}_{\text{ced}}$ are all trees and o^r are the tree-orders, CED reduces to TRBP; when G_r

in $\mathcal{G}_{\text{ced}}$ are all triangulated along order o_r, corresponding to junction trees, CED reduces to generalized TRBP. Unfortunately, this generality does not directly imply better practical performance, because it is intractable to choose the optimal weights $\{w^r\}$ over the set of all possible ordered subgraphs. This problem is partly addressed in Chapter 5 where we propose a novel weighted mini-bucket method, for which we show it is possible to optimize the weights efficiently on subsets of ordered graphs that share a common total ordering o.

3.2.5 Mean Field and Lower bounds

Mean field methods are another class of approximate inference algorithms, which work by restricting $\mathbb{M}$ to a set of tractable distributions on which both the marginal polytope and joint entropy are tractable. More precisely, let $\mathbb{M}_{mf}$ be a subset of $\mathbb{M}$ that corresponds to a set of tractable distributions; for example, *naïve mean field* selects fully factored distributions,

$$\mathbb{M}_{mf} = \big\{\boldsymbol{\tau} \in \mathbb{M} \colon \tau(\boldsymbol{x}) = \prod_{i \in V} \tau_i(x_i)\big\}.$$

Note that the joint entropy $H(\boldsymbol{x}; \boldsymbol{\tau})$ for any $\boldsymbol{\tau} \in \mathbb{M}_{mf}$ decomposes to the sum of singleton entropies $H_i(\tau_i)$ of the marginal distributions $\tau_i(x_i)$, that is,

$$H(\boldsymbol{x}; \boldsymbol{\tau}) = \sum_i H_i(\tau_i).$$

Naïve mean field then approximates the log-partition function by

$$\max_{\boldsymbol{\tau} \in \mathbb{M}_{mf}} \big\{\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{i \in V} H_i(\tau_i)\big\}, \tag{3.43}$$

which is guaranteed to give a lower bound of the log-partition function, because $\mathbb{M}_{mf} \subseteq \mathbb{M}$. Unfortunately, mean field methods usually lead to non-convex optimization problems, because $\mathbb{M}_{mf}$ is usually a non-convex set. In practice, block coordinate descent methods

can be adopted to find the local optima of (3.43). Naïve mean field can be extended to structured mean field [Xing et al., 2002] by using sets of distributions on tree graphs.

Another, related lower bounding algorithm is our negative tree reweigthed BP [Liu and Ihler, 2010], which derives a lower bound using a reverse Jensen's inequality; negative TRBP has the same form as TRBP, but uses properly chosen negative weights to obtain lower bounds. Negative TRBP reduces to naïve or structured mean field bounds when the weights are taken to approach negative infinity in carefully chosen ways; see Liu and Ihler [2010] for details.

3.2.6 Variational Methods for Max-Inference

We have discussed various variational methods for sum-inference, all of which are based on the variational (dual) form (3.24) for the log-partition function. A similar representation exists for max-inference tasks, and also allows us to develop efficient algorithms in similar ways. Specifically, we have,

$$\max_{\boldsymbol{x}} \theta(\boldsymbol{x}) = \max_{\boldsymbol{\tau} \in \mathbb{M}} \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle. \tag{3.44}$$

The right-hand side of (3.44) is a linear program on $\boldsymbol{\tau}$; when the marginal polytope constraint set $\mathbb{M}$ is relaxed to a more manageable constraint set, such as the local consistency polytope $\mathbb{L}(\mathcal{G})$, the resulting approximation is known as the linear programing relaxation for MAP. Notice that (3.44) differs from the log-partition dual form (3.24) only in that it is missing the entropy term. Many efficient algorithms have been developed to solve the linear program (3.44), including max-product linear programming [Globerson and Jaakkola, 2007b], dual decomposition [e.g., Sontag et al., 2011, Komodakis et al., 2011], and more recently augmented Lagrangian methods [e.g., Meshi and Globerson, 2011, Aguiar et al., 2011, Forouzan and Ihler, 2013].

In addition to approaches that directly solve the linear programming objective (3.44), various

methods can also be derived by solving a related, "entropy regularized" version that connects closely to the earlier sum-inference algorithms,

$$\max_{\boldsymbol{\tau} \in \mathbb{M}} \left\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \epsilon \hat{H}(\boldsymbol{x}; \boldsymbol{\tau}) \right\},$$

where ϵ is a small positive number (known as the "temperature" or "annealing parameter"), and $\hat{H}(\boldsymbol{x}; \boldsymbol{\tau})$ is any entropy approximation used for sum-inference. The sum-inference algorithms introduced earlier can be extended for solving this ϵ-annealed version; we can then take $\epsilon \to 0^+$ to obtain corresponding algorithms for max-inference. Roughly speaking, doing so leads to algorithms that replace the sum operators with max operators in the message updates. For example, the max-product BP algorithm for max-inference can be interpreted as a zero-temperature limit of loopy sum-product BP. Weiss et al. [2007] showed that the resulting zero temperature algorithms are guaranteed to return solutions for the linear programming relaxation when the entropy approximation $\hat{H}(\boldsymbol{x}; \boldsymbol{\tau})$ is provably concave. Otherwise, a zero temperature algorithm – such as max-product BP derived via a non-convex Bethe entropy – may still give reasonable approximations on the MAP problem even if it does not solve the linear programming relaxation.

One of the contributions of this thesis is to provide similar extensions for the more general MAP and maximum expected utility (MEU) tasks. Doing so requires that we also develop more general variational representations for these mixed inference problems, which will be discussed in Chapter 4.

Chapter 4

Unifying Variational Representations

In Section 3.2, we introduced the classical variational forms for both max-inference (combinatorial optimization problems) and sum-inference (weighted counting and marginalization problems), given by

$$\log \max_{\boldsymbol{x}} \exp(\theta(\boldsymbol{x})) = \max_{\tau \in \mathbb{M}} \ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle, \tag{4.1}$$

$$\log \sum_{\boldsymbol{x}} \exp(\theta(\boldsymbol{x})) = \max_{\tau \in \mathbb{M}} \left\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + H(\boldsymbol{x}; \boldsymbol{\tau}) \right\}, \tag{4.2}$$

and showed how these forms provide a powerful toolkit for developing various efficient approximate message passing algorithms. However, many of the inference tasks presented in Chapter 2, including marginal MAP and maximum expected utility queries, involve *mixed* inference tasks, corresponding to a combination of both max and sum elimination operators. In this section, we generalize many of these tasks to a unified framework of *weighted* inference problems, and develop a variational representation for exact inference in this weighted framework. This representation forms the foundation of many of the results developed in subsequent chapters, such as the design and analysis of efficient approximate algorithms for the various inference tasks.

We begin in Section 4.1 by simply presenting a number of the key results of our work, specifically the variational forms of various types of mixed inference problems, while highlighting

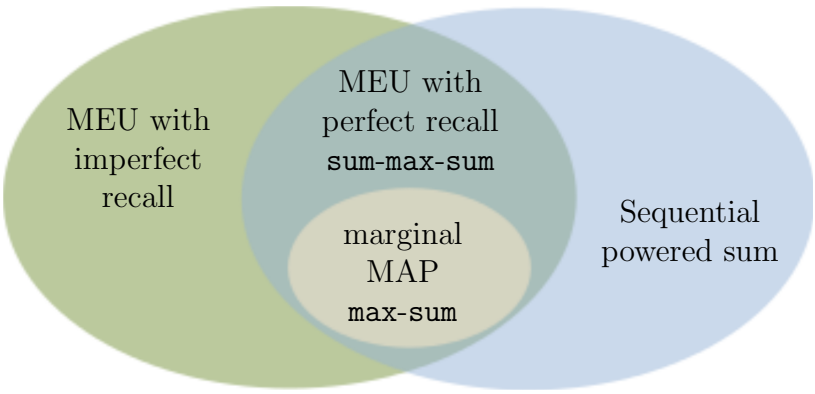


Figure 4.1: Venn diagram of various inference tasks. MEU for decision networks and sequential powered sum represent two types of general inference tasks, both of which include the mixed-elimination forms for marginal MAP and MEU with perfect recall as special cases.

some of the useful intuitions that underlie the results. The subsequent sections then go on to prove these results in two different ways, by studying two generalizations of mixed inference: sequential power sum tasks, and generalized maximum expected utility (MEU) tasks. Specifically, Section 4.2 defines the weighted or powered sum elimination operator, which generalizes both the max and sum elimination operators, and presents a variational representation for a sequential powered sum inference task. This result is sufficient to provide most of the key results stated in Section 4.1, including marginal MAP and decision networks with perfect recall. Section 4.3 then gives an alternate generalization of these inference problems, providing a unified variational form for maximum expected utility inference in decision networks (including those without perfect recall). Both perspectives are useful in understanding certain properties of the inference problems themselves, and in developing new approximations and algorithms in later chapters. An illustration of the relationship among these inference tasks is shown in Figure 4.1.

4.1 Overview and Intuitions

To start, note that the variational forms of max-inference (4.1) and sum-inference (4.2) differ only in the presence or absence of an entropy term. We can understand intuitively what is the effect of this additional term: for maximization, we expect the optimal distribution's moments τ to correspond to a deterministic assignment, with all probability mass assigned to the most likely configuration. For summation, in contrast, the entropy term causes the optimal distribution to spread its probability over many high-probability assignments to increase its entropy – specifically, matching the probability of those assignments. Our main observation is that this intuition applies even to the more general case, such as when max and sum are applied sequentially. Taking marginal MAP as example, our results in this chapter (Section 4.2) show that

$$\log \max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} \exp(\theta(\boldsymbol{x}_{A\cup B})) = \max_{\boldsymbol{\tau}\in\mathbb{M}} \big\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau}) \big\} \tag{4.3}$$

$$= \max_{\boldsymbol{\tau}\in\mathbb{M}} \big\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}_{A\cup B}; \boldsymbol{\tau}) - H(\boldsymbol{x}_B; \boldsymbol{\tau}) \big\}, \tag{4.4}$$

where $H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau}) := -\sum_{\boldsymbol{x}} \tau(\boldsymbol{x}) \log \tau(\boldsymbol{x}_A|\boldsymbol{x}_B)$ is the conditional entropy of the sum variables $\boldsymbol{x}_A$ conditioned on the max variables $\boldsymbol{x}_B$. The equivalent form in (4.4) follows from the entropic chain rule, that the conditional entropy equals the joint entropy minus the entropy of the max variables ($H(\boldsymbol{x}_{A\cup B}; \boldsymbol{\tau}) - H(\boldsymbol{x}_B; \boldsymbol{\tau})$). Intuitively, this objective corresponds to spreading the probability mass of τ over the $\boldsymbol{x}_A$ part (as with sum-inference), while encouraging a deterministic assignment on the $\boldsymbol{x}_B$ part (as with max-inference). Since, like the joint entropy, the conditional entropy is a concave function of $\boldsymbol{\tau}$, this variational form is also a concave optimization over $\boldsymbol{\tau}$. Obviously, the variational forms of both sum-inference in (4.2) and max-inference in (4.1) are special cases of (4.4), corresponding to when the max set B is empty or consists of all nodes, respectively.

Taking this intuition still further, we find that it also applies to more general hybrid sequences, including sum-max-sum inference (corresponding to MEU queries with perfect recall), where sum and max are interleaved in an arbitrary order:

$$\log \max_{x_{D_m}} \sum_{x_{R_m}} \cdots \max_{x_{D_1}} \sum_{x_{R_1}} \exp(\theta(\boldsymbol{x}))$$

$$= \max_{\boldsymbol{\tau} \in \mathbb{M}} \big\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + \sum_{k=1}^{m} H(x_{R_k} | \boldsymbol{x}_{\mathrm{pa}(R_k)} \; ; \; \boldsymbol{\tau}) \big\}, \tag{4.5}$$

$$= \max_{\boldsymbol{\tau} \in \mathbb{M}} \big\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + H(\boldsymbol{x}; \boldsymbol{\tau}) - \sum_{k=1}^{m} H(x_{D_k} | \boldsymbol{x}_{\mathrm{pa}(D_k)} \; ; \; \boldsymbol{\tau}) \big\}, \tag{4.6}$$

where $\boldsymbol{x} = [\boldsymbol{x}_{R_1}, \boldsymbol{x}_{D_1}, \ldots, \boldsymbol{x}_{R_m}, \boldsymbol{x}_{D_m}]$, and $\mathrm{pa}(R_k) := \cup_{k'=k+1}^{m} (R_{k'} \cup D_{k'}) \cup D_k$ are the variables that are not eliminated yet while processing $\boldsymbol{x}_{R_k}$, and similarly $\mathrm{pa}(D_k) := \cup_{k'=k+1}^{m} (R_{k'} \cup D_{k'})$. Here, again, the entropic term can be written either as the sum of conditional entropies of the random (sum) variables $\boldsymbol{x}_{R_m}$ in (4.5), or as the joint entropy minus the conditional entropies of the decision (max) variables $\boldsymbol{x}_{D_m}$ in (4.6). Once again, this variational form is a concave optimization over $\boldsymbol{\tau}$.

Interestingly, we show in Section 4.3 that the "entropy truncation" form in (4.6) remains correct even for more general MEU inference problems in models without perfect recall; in this case, the parent set $\mathrm{pa}(D_k)$ of the decision variables $\boldsymbol{x}_{D_k}$ is specified by the information arcs, and is only a subset of $\cup_{k'=k+1}^{m} (R_{k'} \cup D_{k'})$.

However, in this case the entropic chain rule does not apply, and the "entropy truncation" form (4.6) is no longer equal to the "conditional entropy" form (4.5). Importantly, without perfect recall the objective (4.6) is no longer guaranteed to be concave, because when $\mathrm{pa}(D_k)$ is too small, the negative conditional entropy terms in (4.6) can be "too convex", making the overall entropic term not provably concave, in that it can not be written as a positive combination of conditional entropies. For example, if $\mathrm{pa}(D_k)$ are all empty sets (that is, all the decisions are made independently and simultaneously, without any observed information),

then the entropy term in (4.6) becomes $H(\boldsymbol{x};\boldsymbol{\tau}) - \sum_k^m H(x_{D_k})$, which resembles a (negative) mutual information, and is generally non-concave. See Theorem 4.3 for a full statement of the results.

One of our unifying perspectives is that we can treat both `max` and `sum` as special cases of a `weighted` or `powered sum`, defined as

$$\sum_x^{\overset{w}{+}} f(x) = \Big[\sum_x f(x)^{1/w}\Big]^w,$$

where w can be viewed as a "temperature" or annealing parameter; note that the power sum is equivalent to a p-norm with $p = 1/w$. The power sum elimination reduces to either `sum` or `max` for the specific choices of weights $w = 1$ and $w \to 0^+$ (i.e., $p \to \infty$), respectively. In this view, it is perhaps not surprising to obtain the following result, that interpolates between (4.1) and (4.2):

$$\log \sum_{\boldsymbol{x}}^{\overset{w}{+}} \exp(\theta(\boldsymbol{x})) = \max_{\boldsymbol{\tau}\in\mathbb{M}} \left\{\langle \boldsymbol{\tau}, \boldsymbol{\theta}\rangle + wH(\boldsymbol{x};\boldsymbol{\tau})\right\},$$

for any positive number w.

More generally, we can extend this weighted elimination result to a sequence of eliminations using power sums, in which each variable is given a different weight, that is,

$$\log \sum_{x_n}^{\overset{w_n}{+}} \cdots \sum_{x_1}^{\overset{w_1}{+}} \exp(\theta(\boldsymbol{x})) = \max_{\boldsymbol{\tau}\in\mathbb{M}} \big\{\langle \boldsymbol{\tau}, \boldsymbol{\theta}\rangle + \sum_i w_i H(x_i|\boldsymbol{x}_{i+1:n}\ ;\ \boldsymbol{\tau})\big\},$$

where the w_i are arbitrary positive weights. This form includes (4.1)–(4.6) as special cases, when the various w_i are each chosen to be either 1 or 0^+.

Some important comments are in order:

(1). It is important to note that the ordering of the max and sum elimination operators on the left-hand sides are encoded in the conditioning sets of the entropies (e.g., $\mathrm{pa}(R_k)$) of the

variational forms on the right-hand sides. Altering the elimination order (among operators that do not commute, e.g., have different weights w_i) thus changes the entropy term in the variational form. For example, it is not difficult to show from (4.12) that the "reversed marginal MAP" $\sum_{\boldsymbol{x}_A} \max_{\boldsymbol{x}_B} \exp(\theta(\boldsymbol{x}))$ problem has a form given by

$$\begin{aligned}\log \sum_{\boldsymbol{x}_A} \max_{\boldsymbol{x}_B} \exp(\theta(\boldsymbol{x}_{A\cup B})) &= \max_{\boldsymbol{\tau}\in\mathbb{M}} \big\{\langle \boldsymbol{\theta}, \boldsymbol{\tau}\rangle + H(\boldsymbol{x}_A; \boldsymbol{\tau})\big\} \\ &= \max_{\boldsymbol{\tau}\in\mathbb{M}} \big\{\langle \boldsymbol{\theta}, \boldsymbol{\tau}\rangle + H(\boldsymbol{x}_{A\cup B}; \boldsymbol{\tau}) - H(\boldsymbol{x}_B|\boldsymbol{x}_A; \boldsymbol{\tau})\big\},\end{aligned}$$

which is distinct from the marginal MAP task form in (4.3)–(4.4). In this case, marginal MAP requires us to compute the conditional entropy $H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau})$, while the reversed marginal MAP requires the marginal entropy $H(\boldsymbol{x}_A; \boldsymbol{\tau})$.

(2). The difficulty caused by the non-exchangeability of the max and sum operations is also encoded in the entropy terms, in that these conditional and marginal entropies can be more difficult to calculate than the joint entropy. For example, the variational form (4.4) of marginal MAP requires us to calculate the marginal entropy $H(\boldsymbol{x}_B; \boldsymbol{\tau})$, which may be intractable even when the joint distribution $\tau(\boldsymbol{x}_{A\cup B})$ is a tree; this is because the marginal distribution $\tau(\boldsymbol{x}_B) = \sum_{\boldsymbol{x}_A} \tau(\boldsymbol{x}_{A\cup B})$ does not necessarily inherit the factorization structure of the joint distribution $\tau(\boldsymbol{x}_{A\cup B})$. Therefore, the dual optimization in (4.4) may be intractable even on a tree, reflecting the intrinsic difficulty of marginal MAP compared to joint MAP or marginalization.

(3). One important feature of our variational representations for these mixed inference tasks is that they naturally integrate the max and sum sub-problems into one joint optimization problem, which serves to provide a novel and potentially more efficient framework for mixed inference problems such as marginal MAP, compared to traditional approaches that treat the marginalization sub-problem as a sub-routine of the maximization problem. As we show in Chapter 6 and 7, this viewpoint enables us to derive efficient "mixed-product" message pass-

ing algorithms that simultaneously take marginalization and maximization steps, avoiding expensive and possibly wasteful inner loop steps in the marginalization sub-routine.

4.2 Variational Form for Sequential Powered Sum

In this section, we study the properties of and derive a dual representation for sequential powered sums.

We start by establishing the zero-temperature limits, showing that max and min are special cases of powered sum with weights $w \to 0^+$ and $w \to 0^-$, respectively.

Lemma 4.1. *For any positive function $f(x)$, we have*

$$\lim_{w \to 0^+} [\sum_x f(x)^{1/w}]^w = \max_x f(x), \tag{4.7}$$

$$\lim_{w \to 0^-} [\sum_x f(x)^{1/w}]^w = \min_x f(x). \tag{4.8}$$

Proof. Without loss of generality, we can assume $\max_x f(x) = 1$. Then it is easy to see that

$$\lim_{w \to 0^+} f(x)^{1/w} = \mathbf{1}[x \in \arg\max_x f(x)], \quad \forall x.$$

Summing over x on both sides, we get

$$\lim_{w \to 0^+} \sum_x f(x)^{1/w} = |\arg\max_x f(x)| \stackrel{def}{=} C,$$

where C is the number of global maxima of $f(x)$. Since we always have $C \geq 1 > 0$, we get

$$\lim_{w \to 0^+} [\sum_x f(x)^{1/w}]^w = \lim_{w \to 0^+} C^w = 1.$$

This proves (4.7). The result for $w \to 0^-$ follows similarly. □

We then prove a simple dual form for a single powered sum.

Proposition 4.1. *For any scalar weight w, and the powered sum defined as*

$$\sum_{\boldsymbol{x}}^{w} \exp(\theta(\boldsymbol{x})) = \Big[\sum_{\boldsymbol{x}} \exp\big(\frac{1}{w}\, \theta(\boldsymbol{x}) \big) \Big]^{w}, \tag{4.9}$$

we have:

$$\textit{If } w > 0, \qquad \log \sum_{\boldsymbol{x}}^{w} \exp(\theta(\boldsymbol{x})) = \max_{\boldsymbol{\tau} \in \mathbb{M}} \left\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + w H(\boldsymbol{x}; \boldsymbol{\tau}) \right\}. \tag{4.10}$$

$$\textit{If } w < 0, \qquad \log \sum_{\boldsymbol{x}}^{w} \exp(\theta(\boldsymbol{x})) = \min_{\boldsymbol{\tau} \in \mathbb{M}} \left\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + w H(\boldsymbol{x}; \boldsymbol{\tau}) \right\}. \tag{4.11}$$

Clearly, (4.10) *reduces to the standard sum-inference dual* (4.2) *(with the entropy term) when $w = 1$, and to the max-inference dual* (4.1) *(which is missing the entropy term) when $w \to 0^+$ (e.g., in the limit from the positive side), while* (4.11) *reduces to a "min-inference" dual form when $w \to 0^-$.*

Proof. Simply applying (4.2) on $\exp\big(\frac{1}{w}\theta(\boldsymbol{x})\big)$, we get

$$w \log \sum_{\boldsymbol{x}} \exp(\frac{1}{w}\theta(\boldsymbol{x})) = w \max_{\boldsymbol{\tau} \in \mathbb{M}} \big\{ \langle \boldsymbol{\tau}, \frac{1}{w}\boldsymbol{\theta} \rangle + H(\boldsymbol{x}; \boldsymbol{\tau}) \big\} = \begin{cases} \max\limits_{\boldsymbol{\tau} \in \mathbb{M}} \big\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + w H(\boldsymbol{x}; \boldsymbol{\tau}) \big\}, & w > 0 \\ \min\limits_{\boldsymbol{\tau} \in \mathbb{M}} \big\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + w H(\boldsymbol{x}; \boldsymbol{\tau}) \big\}, & w < 0 \end{cases}$$

where in the last equality, moving w into the max operator yields either (4.10) or (4.11) depending on the sign of w. □

The preceding result applies only to powered sums with a scalar weight. In general, we can sequentially apply powered sums with different weights on different variables. For example, in a simple two variable case, the sequential powered sum is

$$\sum_{x_2}^{w_2} \sum_{x_1}^{w_1} f(x_1, x_2) = (\sum_{x_2} (\sum_{x_1} f(x_1, x_2)^{1/w_1})^{w_1/w_2})^{w_2},$$

which is equivalent to the mixed norm when treating $[f(k,l)]_{kl}$ as a matrix. Note that powered sums with different weights are not commutative; in fact, we have

$$\sum_{x_2}^{w_2}\sum_{x_1}^{w_1} f(x_1,x_2) < \sum_{x_1}^{w_1}\sum_{x_2}^{w_2} f(x_1,x_2), \qquad \text{when } \; w_1 < w_2,$$

unless $f(x_1,x_2)$ does not depend on x_1 or x_2 (e.g., when $f(x_1,x_2)=f(x_1)$). This property is a generalization of the non-commutativity that arises between the max and sum operators.

The following theorem provides the dual form for general, sequential powered sums.

Theorem 4.1. *Consider a weight vector* $\boldsymbol{w}=[w_1,\dots,w_n]$ *associated with variables* $\boldsymbol{x}=[x_1,\dots,x_n]$. *Define the* $\boldsymbol{w}$-weighted log-partition function *of* $p(\boldsymbol{x})\propto\exp(\theta(\boldsymbol{x}))$ *as*

$$\Phi_w(\boldsymbol{\theta},\boldsymbol{w}) = \log \sum_{x_n}^{w_n}\cdots\sum_{x_1}^{w_1}\exp(\theta(\boldsymbol{x})),$$

where $\sum$ *is the powered sum as defined in* (4.9). *Then*

(1). If $w_i>0$, *we have*

$$\log \sum_{x_n}^{w_n}\cdots\sum_{x_1}^{w_1}\exp(\theta(\boldsymbol{x})) = \max_{\boldsymbol{\tau}\in\mathbb{M}}\big\{\langle\boldsymbol{\tau},\boldsymbol{\theta}\rangle + \sum_{i=1}^{n} w_i H(x_i|\boldsymbol{x}_{i+1:n}\;;\;\boldsymbol{\tau})\big\}, \tag{4.12}$$

where $H(x_i|\boldsymbol{x}_{i+1:n}\;;\;\boldsymbol{\tau})$ *is the conditional entropy of* $\tau(\boldsymbol{x})$ *defined by*

$$H(x_i|\boldsymbol{x}_{i+1:n}\;;\;\boldsymbol{\tau}) = -\sum_{\boldsymbol{x}}\tau(\boldsymbol{x})\log\tau(x_i|\boldsymbol{x}_{i+1:n}).$$

As a direct result of (4.12), $\Phi_w(\boldsymbol{\theta},\boldsymbol{w})$ *is a jointly convex function of* $(\boldsymbol{\theta},\boldsymbol{w})$ *when* $w_i>0,\forall i\in[n]$, *because the supremum over a set of linear functions is convex.*

(2). The maximum of (4.12) *is obtained by* $\tau^*(\boldsymbol{x})$ *defined recursively as follows,*

$$\tau^*(\boldsymbol{x}) = \prod_i \tau^*(x_i|\boldsymbol{x}_{i+1:n}), \qquad \tau^*(x_i|\boldsymbol{x}_{i+1:n}) = \frac{\big(Z_{i-1}(\boldsymbol{x}_{i:n}\ ;\ \boldsymbol{\theta}, \boldsymbol{w})\big)^{1/w_i}}{\big(Z_i(\boldsymbol{x}_{i+1:n}\ ;\ \boldsymbol{\theta}, \boldsymbol{w})\big)^{1/w_i}}, \tag{4.13}$$

where Z_i *is the partial powered sum upto* $\boldsymbol{x}_{1:i}$*, that is,*

$$Z_0(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \exp(\theta(\boldsymbol{x})), \qquad Z_i(\boldsymbol{x}_{i+1:n}\ ;\ \boldsymbol{\theta}, \boldsymbol{w}) = \sum_{x_i}^{w_i} \cdots \sum_{x_1}^{w_1} \exp(\theta(\boldsymbol{x})).$$

Note that we have $\sum_{x_i} Z_{i-1}^{1/w_i} = Z_i^{1/w_i}$ *by recursion; this guarantees the conditional distribution* $\tau^*(x_i|\boldsymbol{x}_{i+1:n})$ *to be properly normalized, that is,* $\sum_{x_i} \tau^*(x_i|\boldsymbol{x}_{i+1:n}) = 1$.

(3). The form (4.12) *makes it easy to calculate the derivatives of* $\Phi_w(\boldsymbol{\theta}, \boldsymbol{w})$*, so that*

$$\frac{\partial \Phi_w(\boldsymbol{\theta}, \boldsymbol{w})}{\partial \theta_\alpha(\boldsymbol{x}_\alpha)} = \tau^*(\boldsymbol{x}_\alpha), \qquad \frac{\partial \Phi_w(\boldsymbol{\theta}, \boldsymbol{w})}{\partial w_i} = H(x_i|\boldsymbol{x}_{i+1:n}\ ;\ \boldsymbol{\tau}^*),$$

where $\boldsymbol{\tau}^*$ *is defined in* (4.13)*.*

Proof. The proof is straightforward based on the following more general result that we will prove in Lemma 4.2:

$$\Phi_w(\boldsymbol{\theta}, \boldsymbol{w}) - \sum_i w_i \,\mathrm{KL}(\tau(x_i|\boldsymbol{x}_{i+1:n}) \,||\, \tau^*(x_i|\boldsymbol{x}_{i+1:n})) \ = \ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + \sum_i w_i H(x_i|\boldsymbol{x}_{i+1:n}\ ;\ \boldsymbol{\tau})$$

where the last terms on the left are conditional KL divergences, and $\boldsymbol{\tau}^*$ is as defined in (4.13). Because the minima of the KL divergence terms equal zero and are achieved when $\tau^*(\boldsymbol{x}) = \tau(\boldsymbol{x})$, if all $w_i > 0$ we obtain the dual form (4.12) immediately by simply maximizing over $\boldsymbol{\tau}$ on both sides. □

Lemma 4.2. *For any values of* $(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ *and* $\boldsymbol{\tau} \in \mathbb{M}$*, we have*

$$\Phi_w(\boldsymbol{\theta}, \boldsymbol{w}) - \sum_i w_i \,\mathrm{KL}(\tau(x_i|\boldsymbol{x}_{i+1:n}) \;||\; \tau^*(x_i|\boldsymbol{x}_{i+1:n})) = \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + \sum_i w_i H(x_i|\boldsymbol{x}_{i+1:n} \;;\; \boldsymbol{\tau}) \tag{4.14}$$

where $\tau^*(x_i|\boldsymbol{x}_{i+1:n})$ *are as defined in* (4.13)*, and the last terms on the left are conditional KL divergences (see e.g., Cover and Thomas [2006]), defined by*

$$\mathrm{KL}(\tau(x_i|\boldsymbol{x}_{i+1:n}) \;||\; \tau^*(x_i|\boldsymbol{x}_{i+1:n})) = \mathbb{E}_\tau \log \Big[\frac{\tau(x_i|\boldsymbol{x}_{i+1:n})}{\tau^*(x_i|\boldsymbol{x}_{i+1:n})} \Big],$$

where $\mathbb{E}_\tau[\cdot]$ *is the expectation under distribution* $\tau(\boldsymbol{x})$*.*

Proof. By the definition in Theorem 4.1(2), we have

$$\log Z_0(x_{1:n} \;;\; \boldsymbol{\theta}, \boldsymbol{w}) = \theta(\boldsymbol{x}), \qquad\qquad \log Z_n(\emptyset \;;\; \boldsymbol{\theta}, \boldsymbol{w}) = \Phi_w(\boldsymbol{\theta}, \boldsymbol{w}).$$

Then, we can write the LHS of (4.14) as

$$\begin{aligned} &\Phi_w(\boldsymbol{\theta}, \boldsymbol{w}) - \sum_i w_i \mathrm{KL}(\tau(x_i|\boldsymbol{x}_{i+1:n}) \;||\; \tau^*(x_i|\boldsymbol{x}_{i+1:n})) \\ &\qquad = \log Z_n - \sum_i w_i \mathbb{E}_\tau \big[-\log \tau^*(x_i|\boldsymbol{x}_{i+1:n}) + \log \tau(x_i|\boldsymbol{x}_{i+1:n}) \big] \end{aligned}$$

Applying the definition of τ^* in (4.13) and of the conditional entropy gives,

$$\begin{aligned} &= \log Z_n + \sum_i \mathbb{E}_\tau \big[\log Z_{i-1} - \log Z_i \big] + \sum_i w_i H(x_i|\boldsymbol{x}_{i+1:n} \;;\; \tau) \\ &= \log Z_n + \mathbb{E}_\tau \big[\log Z_0 \big] - \log Z_n + \sum_i w_i H(x_i|\boldsymbol{x}_{i+1:n} \;;\; \tau) \\ &= \mathbb{E}_\tau[\theta(\boldsymbol{x})] + \sum_i w_i H(x_i|\boldsymbol{x}_{i+1:n} \;;\; \tau). \end{aligned}$$

which demonstrates (4.14) and completes the proof. □

Remarks. (1). The results in Theorem 4.1 only apply for positive weights $w_i > 0$; later, we will provide more general results for both positive and negative weights in Theorem 4.2 (proof also based on Lemma 4.2, which establishes for general weights). In particular, the same derivative forms directly extend to general weights.

(2). The result in Theorem 4.1 works for any potential function $\theta(\boldsymbol{x})$. When $\theta(\boldsymbol{x})$ has a factorization structure that is Markov with respect to a graph G, (4.12) can be simplified to

$$\log \sum_{x_n}^{w_n} \cdots \sum_{x_1}^{w_1} \exp(\theta(\boldsymbol{x})) = \max_{\boldsymbol{\tau} \in \mathbb{M}} \big\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + \sum_i w_i H(x_i | \boldsymbol{x}_{\mathrm{pa}_{\tilde{G}}(i)} \; ; \; \boldsymbol{\tau}) \big\},$$

where the conditioning sets on the right side are replaced by $\mathrm{pa}_{\tilde{G}}(i)$, the parent set of i in the induced graph $\tilde{G}$ obtained when triangulating along order $[x_1, \ldots, x_n]$. This form is more computationally efficient, and should be used in practice.

By taking the weights to be either 1 or 0^+, we immediately obtain general variational representations for marginal MAP and interleaved, sum-max-sum inference as follows:

Corollary 4.1. *(1). For marginal MAP, we have*

$$\log \max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} \exp(\theta(\boldsymbol{x})) = \max_{\boldsymbol{\tau} \in \mathbb{M}} \big\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + H(\boldsymbol{x}_A | \boldsymbol{x}_B \; ; \; \boldsymbol{\tau}) \big\}, \tag{4.15}$$

whose maximum is obtained by $\tau^*(\boldsymbol{x}) = p(\boldsymbol{x}_A | \boldsymbol{x}_B) \cdot \mathbf{1}[\boldsymbol{x}_B \in \arg\max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} \exp(\boldsymbol{x})]$*, where* $p(\boldsymbol{x}_A | \boldsymbol{x}_B) = \exp(\theta(\boldsymbol{x})) / \sum_{\boldsymbol{x}_A} \exp(\boldsymbol{x})$*, and* $\mathbf{1}[\cdot]$ *is the indicator function.*

(2). For random variables $\boldsymbol{x} = [x_1, \ldots, x_n]$*, assume there is a partition* $\{D_i, R_i \colon i \in [m]\}$*, satisfying* $R_1 \cup D_1 \cup \cdots R_m \cup D_m = [n]$*. Then, we have*

$$\log \max_{x_{D_m}} \sum_{x_{R_m}} \cdots \max_{x_{D_1}} \sum_{x_{R_1}} \exp(\theta(\boldsymbol{x})) = \max_{\boldsymbol{\tau} \in \mathbb{M}} \big\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + \sum_{k=1}^{m} H(x_{R_k} | \boldsymbol{x}_{\mathrm{pa}(R_k)} \; ; \; \boldsymbol{\tau}) \big\}, \tag{4.16}$$

where $\mathrm{pa}(R_k) = \cup_{k'=k+1}^{m} (R_{k'} \cup D_{k'}) \cup D_k$.

Proof. Both (1) and (2) are direct results of (4.12) by taking weights to be either 1 or 0^+. Here, we give another simpler, more direct proof of (1).

For any $\boldsymbol{\tau} \in \mathbb{M}$ and its corresponding joint distribution $\tau(\boldsymbol{x})$, consider the conditional KL divergence between $\tau(\boldsymbol{x}_A|\boldsymbol{x}_B)$ and $p(x_A|\boldsymbol{x}_B;\boldsymbol{\theta}) := \exp(\theta(\boldsymbol{x}))/\sum_{\boldsymbol{x}_A}\exp(\theta(\boldsymbol{x}))$,

$$\begin{aligned}
\mathrm{KL}(\boldsymbol{\tau}(\boldsymbol{x}_A|\boldsymbol{x}_B) \,||\, p(\boldsymbol{x}_A|\boldsymbol{x}_B)) &= \sum_{\boldsymbol{x}} \tau(\boldsymbol{x}) \log \frac{\tau(\boldsymbol{x}_A|\boldsymbol{x}_B)}{p(\boldsymbol{x}_A|\boldsymbol{x}_B \;;\; \boldsymbol{\theta})} \\
&= -H(\boldsymbol{x}_A|\boldsymbol{x}_B \;;\; \boldsymbol{\tau}) - \mathbb{E}_\tau[\log p(\boldsymbol{x}_A|\boldsymbol{x}_B \;;\; \boldsymbol{\theta})] \\
&= -H(\boldsymbol{x}_A|\boldsymbol{x}_B \;;\; \boldsymbol{\tau}) - \mathbb{E}_\tau[\theta(\boldsymbol{x})] + \mathbb{E}_\tau[\log \sum_{\boldsymbol{x}_A} \exp(\theta(\boldsymbol{x}))] \quad \geq \quad 0,
\end{aligned}$$

where the last inequality follows from the non-negativity of KL divergence, and is tight if and only if $\tau(\boldsymbol{x}_A|\boldsymbol{x}_B) = p(\boldsymbol{x}_A|\boldsymbol{x}_B \;;\; \boldsymbol{\theta})$. Therefore, we have for any $\tau(\boldsymbol{x})$,

$$\log \max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} \exp(\theta(\boldsymbol{x})) \geq \mathbb{E}_{\boldsymbol{\tau}}[\sum_{\boldsymbol{x}_A} \exp(\theta(\boldsymbol{x}))] \geq \mathbb{E}_\tau[\theta(\boldsymbol{x})] + H(\boldsymbol{x}_A|\boldsymbol{x}_B \;;\; \boldsymbol{\tau}).$$

It is easy to show that the two inequality signs are tight if and only if $\tau(x)$ equals $\tau^*(\boldsymbol{x})$ defined above. Substituting $\mathbb{E}_\tau[\theta(\boldsymbol{x})] = \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle$ completes the proof of (4.15). □

As we show in Chapters 6 and 7, these new variational representations play fundamental roles in deriving efficient message passing algorithms for the marginal MAP and sum-max-sum inference problems.

Negative Weights The result in (4.12) applies to positive weights, and allows us to deal with arbitrary combinations of max and sum. However, in some cases it may also be useful to use negative weights, or the min operator (which corresponds to $w \to 0^-$), together with positive weights and max and/or sum operations. As one example, consider two person, zero-sum games, such as captured under multi-agent influence diagram representations [Koller and Milch, 2003] that generalizes influence diagrams; these games involve both min and max,

corresponding to the two players' decisions, and sum operators for averaging over random variables. Related examples arise in adversarial learning, or robust optimization, in which we desire to maximize a worse-case objective against an adversary working to degrade (min) our results. In addition, the min operator and negative weights can be used for constructing and analyzing lower bounds for inference; examples include the mini-bucket lower bound that we introduced in Section 3.1.2 (e.g., see (3.6)), the more general negatively weighted mini-bucket that we introduce in Chapter 5, or our related work on negative tree reweighted BP [Liu and Ihler, 2010], which uses negative counting numbers in TRBP to obtain lower bounds on the log-partition function.

For these reasons, it is important to extend (4.12) to cases involving both positive and negative weights; then, we can take $w_i \to 0^-$ to get min and $w_i \to 0^+$ (or $w_i = 1$) to get max (or sum). The following theorem provides this extension, where the variational form involves finding a saddle point on $\boldsymbol{\tau}$ instead of an maximum.

Theorem 4.2. *(1). Consider a non-zero weight vector $\boldsymbol{w} = [w_1, \dots, w_n]$ associated with variables $\boldsymbol{x} = [x_1, \dots, x_n]$. Let V^+ (resp. V^-) be the subsets on which $w_i > 0$ (resp. $w_i < 0$). Let $\boldsymbol{\tau}_{V^+} = \{\tau(x_i|x_{i+1:n}) \colon i \in V^+\}$ and $\boldsymbol{\tau}_{V^-} = \{\tau(x_i|x_{i+1:n}) \colon i \in V^-\}$, that is, the set of conditional distributions with positive and negative weights, respectively. We have*

$$\log \sum_{x_n}^{w_n} \cdots \sum_{x_1}^{w_1} \exp(\theta(\boldsymbol{x})) = \min_{\boldsymbol{\tau}_{V^-}} \max_{\boldsymbol{\tau}_{V^+}} \big\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + \sum_{i=1}^{n} w_i H(x_i | \boldsymbol{x}_{\mathrm{pa}_{\tilde{G}}(i)} \; ; \; \boldsymbol{\tau}) \big\}, \tag{4.17}$$

for which the τ^* as defined in Theorem 4.1 is a minimax solution. By taking weights equal to $0^{\pm}$ or 1, we can derive variational forms for arbitrary combinations of min, max, and sum elimination operators.

(2). For any $\boldsymbol{\theta}$ and non-zero weights $\boldsymbol{w}$, the derivatives of $\Phi(\boldsymbol{\theta}, \boldsymbol{w})$ (the LHS of (4.17)) w.r.t. $(\boldsymbol{\theta}, \boldsymbol{w})$ are the same as that in Theorem 4.1(3).

Proof. Similarly to the proof of Theorem 4.1, we maximize $\boldsymbol{\tau}_{V^+}$ and minimize $\boldsymbol{\tau}_{V^-}$ on the both sides of (4.14) in Lemma 4.2, and (4.17) follows immediately by the non-negativity of the conditional KL divergence. Note that we can exchange the order of the min and max over $\boldsymbol{\tau}_{V^-}$ and $\boldsymbol{\tau}_{V^+}$ because of the minimax theorem.

The derivative results can be also conveniently calculated through (4.14). Note that $\langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle = \sum_\alpha \tau(\boldsymbol{x}_\alpha)\theta_\alpha(\boldsymbol{x}_\alpha)$. Taking the derivative w.r.t. $\theta_\alpha(\boldsymbol{x}_\alpha)$ on both sides of (4.14), we get

$$\frac{\partial \Phi_w(\boldsymbol{\theta}, \boldsymbol{w})}{\partial \theta_\alpha(\boldsymbol{x}_\alpha)} = \tau(\boldsymbol{x}_\alpha) + \sum_i w_i \frac{\partial \mathrm{KL}(\tau(x_i|\boldsymbol{x}_{i+1:n}) \;||\; \tau^*(x_i|\boldsymbol{x}_{i+1:n}))}{\partial \tau^*(x_i|\boldsymbol{x}_{i+1:n})} \times \frac{\partial \tau^*(x_i|\boldsymbol{x}_{i+1:n})}{\partial \theta_\alpha(\boldsymbol{x}_\alpha)},$$

where the terms in the sum on the right use the derivative chain rule and appear because $\tau^*(x_i|\boldsymbol{x}_{i+1:n})$, as defined in (4.13), is a function of $\boldsymbol{\theta}$. Now taking $\boldsymbol{\tau} = \boldsymbol{\tau}^*$ on the both sides, the terms in the sum become zero because the conditional divergences achieve their minima, and have zero gradient when $\boldsymbol{\tau} = \boldsymbol{\tau}^*$; this proves that

$$\frac{\partial \Phi_w(\boldsymbol{\theta}, \boldsymbol{w})}{\partial \theta_\alpha(\boldsymbol{x}_\alpha)} = \tau^*(\boldsymbol{x}_\alpha).$$

Similarly, taking the derivative w.r.t. w_i on the both sides of (4.14), we get

$$\begin{aligned}\frac{\partial \Phi_w(\boldsymbol{\theta}, \boldsymbol{w})}{\partial w_i} = H(x_i|\boldsymbol{x}_{i+1:n}\;;\;\boldsymbol{\tau}) \;+\; \mathrm{KL}(\tau(x_i|\boldsymbol{x}_{i+1:n}) \;||\; \tau^*(x_i|\boldsymbol{x}_{i+1:n})) \;+\; \cdots \\ \cdots \;+\; \frac{\partial \mathrm{KL}(\tau(x_i|\boldsymbol{x}_{i+1:n}) \;||\; \tau^*(x_i|\boldsymbol{x}_{i+1:n}))}{\partial \tau^*(x_i|\boldsymbol{x}_{i+1:n})} \times \frac{\partial \tau^*(x_i|\boldsymbol{x}_{i+1:n})}{\partial w_i}.\end{aligned}$$

Again, by taking $\boldsymbol{\tau} = \boldsymbol{\tau}^*$ on the both sides, and noting that both the conditional KL divergences and their derivatives equal zero when $\boldsymbol{\tau} = \boldsymbol{\tau}^*$, we get

$$\frac{\partial}{\partial w_i} \Phi_w(\boldsymbol{\theta}, \boldsymbol{w}) = H(x_i|\boldsymbol{x}_{i+1:n}\;;\;\boldsymbol{\tau}^*).$$

This completes the proof. □

4.3 Variational Form for Decision Making With Imperfect Recall

The preceding section provided variational representations for a general sequential powered sum inference task, which includes marginal MAP and decision making with perfect recall (sum-max-sum) as special cases, but does not encompass decision making without perfect recall. In this section, we give a similar variational representation for decision making without perfect recall, which demonstrates the results for marginal MAP and sum-max-sum (Corrolary 4.1) from another perspective.

We start by setting up the MEU task. Consider a decision network (or influence diagram) on $\boldsymbol{x} = [\boldsymbol{x}_R, \boldsymbol{x}_D]$ where $\boldsymbol{x}_R$ and $\boldsymbol{x}_D$ are random and decision variables, respectively. Let $\mathrm{pa}(i)$ be the parent set of decision variable x_i, $\forall i \in D$ as specified by the information arcs. Given the conditional probability $p(\boldsymbol{x}_R|\boldsymbol{x}_D)$ and a utility function $u(\boldsymbol{x}_R, \boldsymbol{x}_D)$, the maximum expected utility task (MEU) is to find an optimal decision strategy $\boldsymbol{\delta} = \{\delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})\colon i \in D\}$ to maximize the expectation of the utility function, that is,

$$\begin{aligned} \mathrm{MEU} &= \max_{\boldsymbol{\delta}\in\Delta} \mathbb{E}(u(\boldsymbol{x}_R, \boldsymbol{x}_D) \mid \boldsymbol{\delta}) \\ &= \max_{\boldsymbol{\delta}\in\Delta} \sum_{\boldsymbol{x}} p(\boldsymbol{x}_R|\boldsymbol{x}_D) u(\boldsymbol{x}_R, \boldsymbol{x}_D) \prod_{i\in D} \delta(x_i|x_{\mathrm{pa}(i)}), \end{aligned} \tag{4.18}$$

where

$$\Delta = \{\delta(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \quad : \quad \delta(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \geq 0, \quad \sum_{x_i} \delta(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) = 1 \quad \forall i \in D,\ \boldsymbol{x} \in \mathcal{X}\}.$$

Theorem 4.3. *(1). Let $\theta(\boldsymbol{x}) = \log(p(\boldsymbol{x}_R|\boldsymbol{x}_D)u(\boldsymbol{x}_R, \boldsymbol{x}_D))$. Then we have*

$$\log \mathrm{MEU}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau}\in\mathbb{M}} \{\langle \boldsymbol{\theta}, \boldsymbol{\tau}\rangle + H(\boldsymbol{x}; \boldsymbol{\tau}) - \sum_{i\in D} H(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}; \boldsymbol{\tau})\}. \tag{4.19}$$

Suppose $\boldsymbol{\tau}^*$ is a maximum of (4.19)*, then $\boldsymbol{\delta}^* = \{\boldsymbol{\tau}^*(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})\colon i \in D\}$ is an optimal strategy that solves* (4.18)*.*

(2). If the perfect recall condition is satisfied, then (4.19) *reduces to a convex optimization,*

$$\log \mathrm{MEU}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau} \in \mathbb{M}} \{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{o_i \in R} H(x_{o_i} | \boldsymbol{x}_{o_{i+1:n}}; \boldsymbol{\tau}) \}, \tag{4.20}$$

where $o = [r_0, d_1, r_1, \cdots, d_m, r_m]$ *is the reverse of a "temporal" ordering of the decision and chance nodes as implied by the perfect recall.*

Proof. (1) Let $\delta = \{\delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \colon i \in D\}$ be any randomized strategy, and let $\exp(\theta^\delta(\boldsymbol{x})) = \exp(\theta(\boldsymbol{x})) \prod_{i \in D} \delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$. We apply the standard duality result (4.2) of the log-partition function on $p^\delta(\boldsymbol{x}) \propto \exp(\theta^\delta(\boldsymbol{x}))$:

$$\begin{aligned}
\log \mathrm{MEU}(\boldsymbol{\theta}) &= \max_{\boldsymbol{\delta}} \log \sum_{\boldsymbol{x}} \prod_{i \in R} p(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \prod_{i \in D} \delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \\
&= \max_{\boldsymbol{\delta}} \log \sum_{\boldsymbol{x}} \exp(\theta^\delta(\boldsymbol{x})) \\
&= \max_{\boldsymbol{\delta}} \big\{ \max_{\boldsymbol{\tau} \in \mathbb{M}} \big[\langle \boldsymbol{\theta}^\delta, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}; \boldsymbol{\tau}) \big] \big\} \\
&= \max_{\boldsymbol{\tau} \in \mathbb{M}} \big\{ \max_{\boldsymbol{\delta}} \big[\langle \boldsymbol{\theta}^\delta, \boldsymbol{\tau} \rangle \big] + H(\boldsymbol{x}; \boldsymbol{\tau}) \big\}.
\end{aligned} \tag{4.21}$$

Then, note that

$$\begin{aligned}
\max_{\boldsymbol{\delta}} \big[\langle \boldsymbol{\theta}^\delta, \boldsymbol{\tau} \rangle \big] &= \max_{\boldsymbol{\delta}} \{ \langle \boldsymbol{\theta} + \sum_{i \in D} \log \delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}), \boldsymbol{\tau} \rangle \} \\
&= \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{i \in D} \max_{\delta_i} \{ \sum_x \tau(\boldsymbol{x}) \log \delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \} \\
&\overset{*}{=} \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{i \in D} \{ \sum_x \tau(\boldsymbol{x}) \log \tau(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \} \\
&= \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle - \sum_{i \in D} H(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}; \boldsymbol{\tau}),
\end{aligned} \tag{4.22}$$

where the equality "$\overset{*}{=}$" holds because the solution of $\max_{\delta_i} \{ \sum_x \tau(\boldsymbol{x}) \log \delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \}$ subject to the normalization constraint $\sum_{x_i} \delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) = 1$ is $\delta_i = \tau(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$. Finally, we obtain

(4.19) by plugging (4.22) into (4.21).

(2) Note that the perfect recall assumption by definition implies $\mathrm{pa}(o_i) = o_{i+1:n}$ for $o_i \in D$. By the entropic chain rule, we have

$$H(\boldsymbol{x};\boldsymbol{\tau}) = \sum_{o_k \in R \cup D} H(x_{o_i}|\boldsymbol{x}_{o_{i+1:n}}).$$

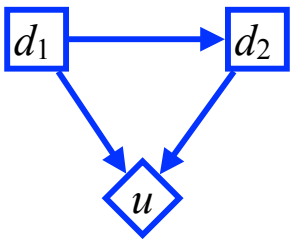
Subtracting the terms in (4.19) leaves exactly those terms in (4.20). □

Remarks. (1). Note that Theorem 4.3 applied to the case of perfect recall is equivalent to Corollary 4.1 for its sum-max-sum form, but provides a different proof.

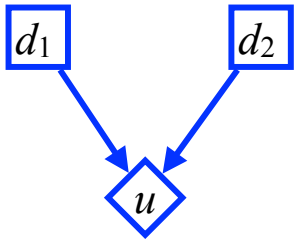
(2). It is interesting to see that, under the variational forms, the perfect recall and imperfect recall cases differ in whether the optimization is guaranteed to be convex. In the perfect recall case, (4.20) is always a convex optimization (if perhaps not strictly convex), due to the concavity of each conditional entropy function in the sum. In contrast, (4.19) may be non-convex – each subtracted entropy term is conditioned on fewer variables than in perfect recall, which increases their entropy, and may "overwhelm" the joint entropy causing their difference to be non-convex. This matches the intuition that incomplete information sharing gives rise to multiple, locally optimal strategies (corresponding to locally optimal points of the non-convex optimization), and greatly increases the computational difficulty for finding the global optimal solution.

Example 4.1. *Consider the two simple decision networks in Figure 4.2(a) and Figure 4.2(b), which have perfect recall and imperfect recall, respectively (see also Figure 2.5 for more details). Their variational representations are*

Figure 2.5(a), Perfect Recall:	$\max_{\boldsymbol{\tau}} \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle,$
Figure 2.5(b), Imperfect Recall:	$\max_{\boldsymbol{\tau}} \big\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + H(x_1, x_2) - H(x_1) - H(x_2) \big\}$



(a) Perfect Recall



(b) Imperfect Recall

d_1	d_2	$u(d_1, d_2)$
1	1	3
0	1	1
1	0	1
0	0	2

(c) Utility Function

Figure 4.2: Figures for example 4.1. (a)-(b) Two decision networks sharing a utility function in (c), with perfect recall (a), and imperfect recall (b), respectively.

where there is no entropic term for Figure 2.5(a) since no random variables exist; the entropic term for Figure 2.5(b) equals the negative mutual information, $-I(x_1, x_2) = H(x_1, x_2) - H(x_1) - H(x_2)$ *and is not a concave function of* $\boldsymbol{\tau}$.

Another corollary provides additional insights into the non-convexity issues raised by imperfect recall. Let us consider extending the parent sets of a LIMID to ensure that it has the perfect recall property. Given a LIMID, assume o is an "reverse temporal ordering" over the chance and decision nodes that is consistent with its information arcs — for any decision variable o_i, all of its parent set $\text{pa}(o_i)$ should rank later than o_i in o, that is, $\text{pa}(o_i) \subseteq o_{i+1:n}$ for $\forall o_i \in D$. We can then relax the LIMID to a new ID with perfect recall by augmenting the parent sets of each decision node to $\{\overline{\text{pa}}(o_i) = o_{i+1:n} \colon o_i \in D\}$, that is, the decision policies can depend on more information ($\overline{\text{pa}}(o_i)$) than before ($\text{pa}(o_i)$). The imperfect recall MEU then corresponds to an optimization in the perfect recall ID, subject to additional constraints on what policies (and thus marginals) are allowed:

Corollary 4.2. *For an ID with natural parameter* $\boldsymbol{\theta}$ *and an augmented parent set* $\{\overline{\text{pa}}(o_i) = o_{1:i-1} \colon o_i \in D\}$ *as defined above, we have*

$$\log \text{MEU}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau} \in \mathbb{I}} \{\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{o_i \in R} H(\boldsymbol{x}_{o_i} | \boldsymbol{x}_{o_{i+1:n}}; \boldsymbol{\tau})\}, \tag{4.23}$$

where $\mathbb{I} = \{\boldsymbol{\tau} \in \mathbb{M} : \ \boldsymbol{x}_{o_i} \perp \boldsymbol{x}_{\overline{\text{pa}}(o_i) \setminus \text{pa}(o_i)} | \boldsymbol{x}_{\text{pa}(o_i)}, \forall o_i \in D\}$, *corresponding to those distributions*

that respect the imperfect recall constraints; "$x \perp y \mid z$" denotes conditional independence of x and y given z, that is, $\tau(x, y|z) = \tau(x|z)\tau(y|z)$. Thus, $\mathbb{I}$ is the set of marginals whose policies leave the distribution of decision variable x_{o_i} independent of its ancestors, except for its original information arcs $\mathrm{pa}(o_i)$.

Proof. See Appendix. □

Thus Corollary 4.2 gives another intuitive interpretation of imperfect vs. perfect recall: MEU with imperfect recall can be viewed as optimizing the same objective function as the "relaxed" perfect recall ID, but over a subset of the marginal polytope that restricts the observation domains of the decision rules; this inner subset is non-convex and similar to the mean field approximation for partition functions. See Wolpert [2006] for a similar connection between mean field and bounded rational game theory. Interestingly, this shows that extending a LIMID to have perfect recall (by extending the observation domains of the decision nodes) can be considered a convex relaxation of the LIMID.

We can also introduce an "annealing" parameter to control the convexity of the objective function, giving us the following corollary.

Corollary 4.3. *For any ϵ, if $\boldsymbol{\tau}^*$ is a global optimum of*

$$\max_{\boldsymbol{\tau} \in \mathbb{M}} \{\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}) - (1 - \epsilon) \sum_{i \in D} H(x_i | \boldsymbol{x}_{\mathrm{pa}(i)})\}. \tag{4.24}$$

and $\boldsymbol{\delta}^* = \{\tau^*(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \colon i \in D\}$ is a deterministic strategy, then $\boldsymbol{\delta}^*$ is an optimal strategy for MEU.

Proof. See Appendix. □

The annealing parameter ϵ in Corollary 4.3 increases the convexity of the (negative) objective

Problem Type	**Primal Form**	**Variational Form**
Max-inference	$\max_{\boldsymbol{x}} \exp(\theta(\boldsymbol{x}))$	$\max_{\boldsymbol{\tau}\in\mathbb{M}} \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle$
Sum-inference	$\sum_{\boldsymbol{x}} \exp(\theta(\boldsymbol{x}))$	$\max_{\boldsymbol{\tau}\in\mathbb{M}} \{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}; \boldsymbol{\tau}) \}$
Marginal MAP	$\max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} \exp(\theta(\boldsymbol{x}))$	$\max_{\boldsymbol{\tau}\in\mathbb{M}} \{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}_A \vert \boldsymbol{x}_B ; \boldsymbol{\tau}) \}$
Decision Making (perfect recall)	$\sum_{\boldsymbol{x}_{r_1}} \max_{\boldsymbol{x}_{d_1}} \cdots \sum_{\boldsymbol{x}_{rm}} \exp(\theta(\boldsymbol{x}))$	$\max_{\boldsymbol{\tau}\in\mathbb{M}} \{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{o_i \in R} H(x_{o_i} \vert \boldsymbol{x}_{o_{1:i-1}} ; \boldsymbol{\tau}) \}$
Decision Making (imperfect recall)	$\max_{\delta\in\Delta} \sum_{\boldsymbol{x}} \exp(\theta(\boldsymbol{x})) \delta(\boldsymbol{x}_D \vert \boldsymbol{x}_R)$	$\max_{\boldsymbol{\tau}\in\mathbb{M}} \{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}; \boldsymbol{\tau}) - \sum_{i \in D} H(x_i \vert \boldsymbol{x}_{\mathrm{pa}(i)} ; \boldsymbol{\tau}) \}$
Powered Sum	$\sum_{x_n}^{w_n} \cdots \sum_{x_1}^{w_1} \exp(\theta(\boldsymbol{x}))$	$\max_{\boldsymbol{\tau}\in\mathbb{M}} \{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{i=1}^{n} w_i H(x_i \vert \boldsymbol{x}_{i+1:n} ; \boldsymbol{\tau}) \}$

Table 4.1: Summarizing the variational forms of various inference problems. The results of sum-inference and max-inference are well known, while the rests are our contributions in this thesis. All of the variational forms differ only on their entropy terms.

function, but at the risk of losing optimality of the strategy: for sufficiently large ϵ, e.g., $\epsilon \geq 1$, the negative objective in (4.24) is a strictly convex function, but $\boldsymbol{\delta}^*$ is unlikely to be deterministic nor optimal (notice, if $\epsilon = 1$, (4.24) reduces to standard marginalization). As ϵ is decreased towards zero, $\boldsymbol{\delta}^*$ becomes more deterministic, but (4.24) becomes more non-convex and hence harder to solve. See Theorem 6.1 for a similar result on marginal MAP. In Chapter 7, we derive several optimization approaches by manipulating this annealing parameter.

4.4 Conclusion

This chapter introduced a spectrum of variational forms, which are summarized in Table 4.1 and extend those known for standard sum-inference and max-inference problems. These

results form the basis for our algorithmic developments in the following chapters:

1. The variational form of the sequential powered sum (and its properties) in Theorem 4.1 are exploited in Chapter 5 to tighten a novel weighted mini-bucket bound that we propose, which generalizes the mini-bucket algorithm from Section 3.1.2. It also serves to connect our weighted mini-bucket bound to convex variational methods, such as tree reweighted belief propagation and conditional entropy decomposition.
2. The variational form for marginal MAP in Corollary 4.1 allows us to develop a spectrum of efficient belief propagation-type algorithms for marginal MAP inference in Chapter 6, which greatly enhance the state of the art for this problem.
3. Similarly, the result for decision making in Theorem 4.3 forms the basis for our algorithmic developments in Chapter 7, enabling a set of powerful algorithms for solving decision networks. Importantly, our novel decision-making algorithms are applicable even in systems with imperfect recall (i,e., LIMIDs), for which few efficient algorithms beyond the greedy single policy update exist.

Chapter 5

Weighted Mini-Bucket Elimination

In this section, we introduce Hölder's inequality as a generalization of the mini-bucket elimination (MBE) bound, and propose a more general *weighted* mini-bucket (WMB) elimination algorithm. Compared to standard MBE, which approximates intractable summation operators using upper (resp. lower) bounds built on max (resp. min) operators, weighted mini-bucket builds its approximation using the more general powered sum, and returns an upper or lower bound depending on the signs of the weights. In addition, we also study iterative tightening methods to find the optimal weight values, as well as the optimal factor splitting (or reparameterization) in weighted mini-bucket.

By using the variational representation of the sequential powered sum in Theorem 4.1, we show that the dual form of weighted mini-bucket is equivalent to that of tree-reweighted belief propagation (TRBP), and more generally conditional entropy decomposition variational methods, revealing a close connection between the mini-bucket *elimination* method and convex *variational* methods. Importantly, our weighted mini-bucket technique inherits many of the benefits of both variational and elimination approaches, resulting in significant advantages over either — compared with mini-bucket, we can iteratively improve the parameters, leading to tighter bounds; compared with TRBP, we provide a concise and relatively tight primal bound with far fewer parameters. In experiments, we show that WMB effectively trades off iterative updates with clique size to greatly improve the overall bound quality.

This chapter is organized as follows. Section 5.1 introduces Hölder's and reverse Hölder's Inequalities. Section 5.2 proposes weighted mini-bucket elimination, and discuss its connection with the tree reweighted (TRW) primal bound (Section 5.2.2). Section 5.3 studies iterative methods for tightening upper bounds and lower bounds, respectively. Section 5.4 provides the dual form of the weighted mini-buck bound, and develops its connection with the TRW dual bound and conditional entropy decomposition. We present experiments in Section 5.5, then discuss conclusions and outline future directions in Section 5.6.

5.1 Hölder's and Reverse Hölder's Inequalities

Generally speaking, the partition function of a factorized distribution $Z = \sum_{\boldsymbol{x}} \prod_\alpha \psi_\alpha(\boldsymbol{x}_\alpha)$ involves calculating the (high dimensional) sum of products of functions. Hölder's and reverse Hölder's inequalities provide general tools for constructing bounds or approximations to the sum of products, and form the foundation for our method.

Proposition 5.1 (**Hölder's and reverse Hölder's Inequalities**). *Let $f_r(x)$, $r = 1 \dots p$ be a set of positive functions over the discrete variable x, and let $w = [w_1, w_2, \dots, w_R]$ be a vector of non-zero weights. Define*

$$\mathcal{W}^+ = \big\{w \colon \sum_r w_r = 1, \textit{ and } w_r > 0,\ \forall r = 1, \dots, R\big\},$$

$$\mathcal{W}^- = \bigcup_{k=1}^{p} \mathcal{W}_k^-, \qquad \textit{where} \qquad \mathcal{W}_k^- = \big\{w \colon \sum_r w_r = 1, \textit{ and } w_k > 0,\ w_r < 0\ \forall r \neq k\big\},$$

that is, $\mathcal{W}^+$ is the probability simplex, while $\mathcal{W}^-$ is the set of normalized weights with exactly

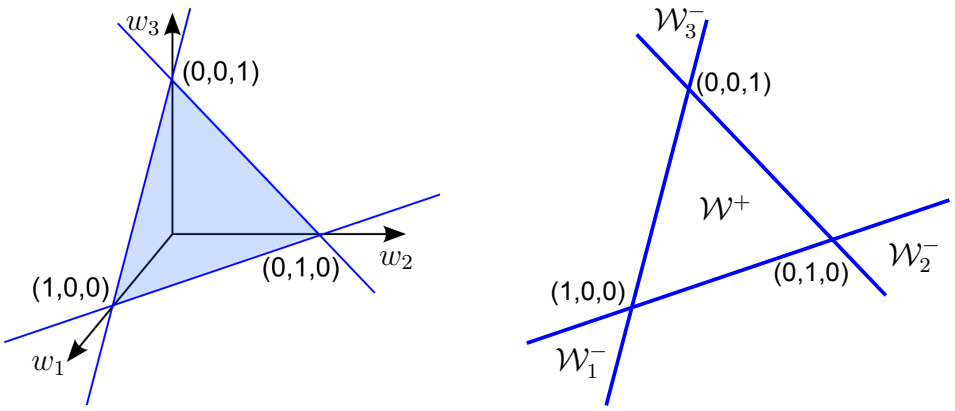


Figure 5.1: Illustrating the weight domains for Hölder's and reversed Hölder's inequalities. For three weights (w_1, w_2, w_3), we restrict our attention to the plane defined by normalized w_r, i.e., $\mathcal{W} = \{w\colon \sum_r w_r = 1\}$, on which the blue lines lie. Hölder's inequality holds on the probability simplex $\mathcal{W}^+$ in which all the weights are positive; reverse Hölder's inequality holds on $\mathcal{W}_1^- \cup \mathcal{W}_2^- \cup \mathcal{W}_3^-$, where exactly one weight is positive. However, neither inequality holds in the unmarked domains between $\mathcal{W}_r^-$ where exactly two weights are positive.

one positive element (see Figure 5.1 for a visualization). Then we have

$$\sum_x \prod_r f_r(x) \le \prod_i \sum_x^{w_r} f_r(x), \qquad \textit{when } w \in \mathcal{W}^+, \tag{5.1}$$

$$\sum_x \prod_r f_r(x) \ge \prod_i \sum_x^{w_r} f_r(x), \qquad \textit{when } w \in \mathcal{W}^-. \tag{5.2}$$

where as usual, $\sum_x^w f(x) = (\sum_x f(x)^{1/w})^w$ *is the weighted or powered sum (see Section 4.2). In both cases, the equalities hold if there is some common function* $g(x)$ *such that* $f_r(x)^{1/w_r} = g(x)$, $\forall r = 1, \ldots, R$.

Proof. See Hardy et al. [1988]. □

The directions of the inequalities above depend on the sign of the weights: using weights with all positive elements yields the well-known Hölder's inequality, while using weights with exactly one positive element (e.g., $w = [2, -1/(R-1), \ldots, -1/(R-1)]$) yield the reverse Hölder's inequality. If the weight vector satisfies neither of the above conditions

(e.g., $w = [1, 1, -1/(R-2), \ldots, -1/(R-2)]$), then no inequalities hold in general. Figure 5.1 shows an illustration of the various domains and where the inequalities hold.

The weights should also be properly normalized (i.e., $\sum_r w_r = 1$), so that the function values are not over- (or under-) counted. For example, although the inequality (5.1) still holds with the all-one weight ($w = [1, \ldots, 1]$), that is,

$$\sum_x \prod f_r(x) \leq \prod \sum_x f_r(x),$$

this gives a very loose (almost trivial) upper bound because the terms are over-counted on the right-hand side (to see this, take $f_r(x) = 1$; then the right hand side becomes $|\mathcal{X}|^R$, while the left hand side is $|\mathcal{X}|$, where $|\mathcal{X}|$ is the domain size of x). On the other hand, one may form a reasonable approximation by using a more careful normalization,

$$\frac{1}{|\mathcal{X}|} \sum_x \prod_r f_r(x) \approx \frac{1}{|\mathcal{X}|^p} \prod \sum_x f_r(x).$$

However, this is in general neither an upper nor a lower bound; see Sutton and McCallum [2009] for an application of such a type of approximation, and its connection with Bethe entropy approximation.

5.2 Weighted Mini-Bucket Elimination

The inequalities in (5.1)-(5.2) can be viewed as approximating the sum of products of functions using products of powered sums over the individual functions; in this sense, they can be used to provide upper (resp. lower) bounds for the partition function in a manner similar to mini-bucket elimination. We illustrate the idea using our running example from the

description of mini-bucket elimination in Section 3.1.2,

$$\psi_{23}^{new}(x_2, x_3) = \sum_{x_1} \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3),$$

for which Hölder's and reversed Hölder's inequalities give

$$\sum_{x_1}^{w_1} \psi_{12}(x_1, x_2) \cdot \sum_{x_1}^{w_2} \psi_{13}(x_1, x_3) \ \gtreqless \ \sum_{x_1} \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3), \tag{5.3}$$

where w_1 and w_2 are weights satisfying $w_1 + w_2 = 1$, and the direction of the inequality depends on the sign of the weights $[w_1, w_2]$.

Recall that the powered sum $\sum_x^w f(x) = (\sum_x f(x)^{1/w})^w$ approaches $\max_x f(x)$ as $w \to 0^+$, and $\min_x f(x)$ as $w \to 0^-$. Then it is easy to see that (5.3) is an immediate generalization of the mini-bucket upper bound (3.6) and lower bound (3.5), in which we take $w_1 = 1$, $w_2 \to 0+$ (corresponding to a max operator) and $w_1 = 1$, $w_2 \to 0^-$ (corresponding to a min operator), respectively.

Based on the above observation, it is straightforward to generalize mini-bucket elimination to the *weighted mini-bucket elimination* (WMB) presented in Algorithm 5.1, by replacing the naïve mini-bucket bound with Hölder's inequality. This algorithm uses the same basic procedure as standard mini-bucket, except that `sum`/`max` are replaced by `powered sum`, whose weights sum to one for each variable.

Remark. To implement Algorithm 5.1 for lower bounds ($w \in \mathcal{W}^-$), one needs to select a single replicate of each node to take on a positive weight; in relation to standard mini-bucket in Algorithm 3.3, this corresponds to selecting a single mini-bucket on which the sum operator is applied in (3.7). On the other hand, for obtaining upper bounds ($w \in \mathcal{W}^+$), weighted mini-bucket avoids the corresponding selection problem since all the weights should take positive values, and are thus treated equally.

Algorithm 5.1 Weighted mini-bucket elimination

Input: Factors of a graphical model $\mathbf{F} = \{\psi_\alpha(\boldsymbol{x}_\alpha)\}$, an elimination order $o = [x_1, \cdots, x_n]$, and an *ibound*.
Output: An upper (or lower) bound on the partition function Z.
for $i \leftarrow 1$ to n **do**

1. Find the factors involving variable x_i: $\mathbf{B}_i \leftarrow \{\psi \colon \psi \in \mathbf{F}, x_i \in \text{scope}(\psi)\}$.
2. Partition $\mathbf{B}_i$ into R_i subgroups $\{\mathbf{B}_{ir}\}$, such that $\cup_{r=1}^{R_i} \mathbf{B}_{ir} = \mathbf{B}_i$ and

$$|\cup_{\psi \in \mathbf{B}_{ir}} \text{scope}(\psi)| \leq \textit{ibound} + 1 \quad \text{for all } r = 1, \ldots, R_i.$$

Assign each $\mathbf{B}_{ir}$ with a weight w_{ir}, such that $\sum_r w_{ir} = 1$.
for $r \leftarrow 1 \cdots R_i$ **do**

$$\psi_{ir}^{new}(\boldsymbol{x}_{\pi_{ir}}) \leftarrow \Big[\sum_{x_i} \prod_{\psi \in \mathbf{B}_{ir}} \psi(\boldsymbol{x}_{\text{scope}(\psi)})^{1/w_{ir}} \Big]^{w_{ir}}, \tag{5.4}$$

where the variable scope is $\pi_{ir} = \cup_{\psi \in \mathbf{B}_{ir}} \text{scope}(\psi) \setminus \{i\}$.
end for

3. $\mathbf{F} \leftarrow (\mathbf{F} \setminus \mathbf{B}_i) \cup \{\psi_{ir}^{new} \colon r = 1, \cdots, R_i\}$.

end for
The partition function bound is $\hat{Z} = \prod_{\psi \in \mathbf{F}} \psi$. (All the factors are constant (i.e., $\text{scope}(\psi) = \emptyset$) when the elimination are completed).

Note: $\hat{Z}$ is an upper or lower bound of the partition function depending on the signs of the weights w_i^r; see Proposition 5.1.
Note: when the w_i^r are set to be either 1 or $0^+/0^-$, the algorithm reduces to the standard mini-bucket elimination of Algorithm 3.3.

5.2.1 Weighted Mini-Bucket as Powered Sum Inference

Just as we showed in Section 3.1.2 that standard MBE was equivalent to a sum-max-sum (or sum-min-sum) form, we can also represent our weighted mini-bucket bound as a sequential powered sum inference on the augmented model defined by splitting variables. This characterizes the weighted mini-bucket bound as an explicit function of the augmented model parameters and the weights, and enables us to make connection with tree reweighted BP (Section 5.2.2 and Section 5.4), as well as develop efficient algorithms that optimize the parameters and weights to get the tightest bounds (Section 5.3).

To be specific, let $\bar{p}(\bar{\boldsymbol{x}}) \propto \prod_{\bar{\alpha}\in\bar{\mathcal{I}}} \bar{\psi}(\bar{\boldsymbol{x}}_{\bar{\alpha}}) = \exp(\sum_{\bar{\alpha}\in\bar{\mathcal{I}}} \theta_{\bar{\alpha}}(\bar{\boldsymbol{x}}_{\bar{\alpha}}))$ be the augmented model on the replicated variables $\bar{\boldsymbol{x}}$ obtain by Algorithm 3.5, and $\bar{\boldsymbol{w}} = [\bar{\boldsymbol{w}}_1, \ldots, \bar{\boldsymbol{w}}_n]$, where $\bar{\boldsymbol{w}}_i = [\bar{w}_{i^1}, \ldots, \bar{w}_{i^{R_i}}]$, are the weights on assigned on $\bar{\boldsymbol{x}}$. Following the elimination steps in Algorithm 5.1, the log of the resulting bound can be written into the form of a sequential powered sum on $\bar{p}(\bar{\boldsymbol{x}})$ (see Theorem 4.1), that is,

$$\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) = \log \sum_{\bar{x}_{\bar{n}}}^{\bar{w}_{\bar{n}}} \cdots \sum_{\bar{x}_1}^{\bar{w}_1} \exp\Big(\sum_{\bar{\alpha}\in\bar{\mathcal{I}}} \theta_{\bar{\alpha}}(\bar{\boldsymbol{x}}_{\bar{\alpha}})\Big) \tag{5.5}$$

where the powered sums (with different weights) are applied recursively on the variables along the ordering $\bar{\boldsymbol{x}} = [\bar{x}_{1^1}, \ldots, \bar{x}_{1^{R_1}}, \ldots, \bar{x}_{n^1}, \ldots \bar{x}_{n^{R_n}}]$. In this sense, weighted mini-bucket elimination uses the $\bar{\boldsymbol{w}}$-*weighted partition function* of the augmented model to approximate the regular partition function of the original model.

Theorem 4.1 in Chapter 4 examined some of the important properties of $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$: we showed that $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ is a joint convex function of $(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ for positive weights, and derived its derivates w.r.t. both $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$, as well as a variational dual representation. In the sequel, Sections 5.2.2–5.4 leverage these results to analyze weighted mini-bucket, connecting it to convex variational methods such as TRBP, and develop efficient algorithms to optimize $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$ to get tight bounds.

5.2.2 Connection to TRW Primal Bounds

Our weighted mini-bucket works by splitting the variables node by node, forming the augmented model $\bar{p}(\bar{\boldsymbol{x}})$ whose induced width is controlled by the *ibound*. As we illustrated in Section 3.1.2 and in particular Figure 3.3, this process can be graphically interpreted as splitting the nodes to break the loops in the original Markov graph, forming a"covering graph" (the Markov graph of the augmented model) to approximate the original graph. A related but different notion of graph splitting has been used in tree reweighted BP (TRBP)

[Wainwright et al., 2005] (see also Section 3.2.4), which splits the original graph into a set of spanning trees. In this section, we connect and compare the TRW primal bound and our weighted mini-bucket bound. We further discuss TRW in Section 5.3 and Section 5.4, where we make connections with the TRBP algorithm and the dual form of the TRW bound.

We start by reviewing the TRW primal bound briefly. For notational convenience, we restrict the discussion to pairwise models with a Markov graph G in this section. Let $\mathcal{T} = \{T\}$ be a collection of spanning trees of G which are assigned a set of positive weights $\{w^T\}$ satisfying $\sum_{T\in\mathcal{T}} w^T = 1$ (that is, $\{w^T\} \in \mathcal{W}^+$). We split the original natural parameter $\boldsymbol{\theta}$ into the sum of $\{\boldsymbol{\theta}^T : T \in \mathcal{T}\}$, where each $\boldsymbol{\theta}^T$ is defined on tree T and together satisfy $\boldsymbol{\theta} = \sum_{T\in\mathcal{T}} \boldsymbol{\theta}^T$. The TRW primal bound, as derived via Jensen's inequality in Section 3.2.4[1], can be formed using the powered sum notation as follows,

$$\Psi_{trw}(\{\boldsymbol{\theta}^T\}, \{w^T\}) = \log \prod_{T\in\mathcal{T}} \sum_{\boldsymbol{x}}^{w^T} \exp(\theta^T(\boldsymbol{x})). \tag{5.6}$$

To more closely connect to our weighted mini-bucket, we can assume the sum for each tree T is performed on a copy $\boldsymbol{x}^T$ of variable $\boldsymbol{x}$. This creates an augmented variable vector $\boldsymbol{x}^{\mathcal{T}} = [\boldsymbol{x}^{T_1}, \ldots, \boldsymbol{x}^{T_K}]$, where T_k is the k-th tree in $\mathcal{T}$, and a large augmented model $\bar{p}(\boldsymbol{x}^{\mathcal{T}}) \propto \exp(\sum_k \theta^{T_k}(\boldsymbol{x}^{T_k}))$, consisting of a set of disconnected spanning trees (that is, a "forest"). Then the TRBP primal bound is rewritten as a sequential weighted sum over $\bar{p}(\boldsymbol{x}^T)$,

$$\Psi_{trw}(\{\boldsymbol{\theta}^T\}, \{w^T\}) = \log \prod_{T\in\mathcal{T}} \sum_{\boldsymbol{x}}^{w^T} \exp(\theta^T(\boldsymbol{x})) = \log \sum_{\boldsymbol{x}^{T_K}}^{w^{T_K}} \cdots \sum_{\boldsymbol{x}^{T_1}}^{w^{T_1}} \exp(\sum_k \theta^{T_k}(\boldsymbol{x}^{T_k})). \tag{5.7}$$

Therefore, we can view TRBP as splitting each variable into $K = |\mathcal{T}|$ copies, and assigning the k-th copy $\boldsymbol{x}^{T_k} = [x_1^{T_k}, \ldots, x_n^{T_k}]$ of all the variables with a tied positive weight w^{T_k}.

[1]The TRW primal bound can also be derived by Hölder's inequality. To see this, note that the log partition function is $\Phi(\theta) = \log \sum_{\boldsymbol{x}} \exp(\theta(\boldsymbol{x})) = \log \sum_{\boldsymbol{x}} \prod_T \exp(\theta^T(\boldsymbol{x}))$; applying Hölder's inequality by treating each $\exp(\theta^T(\boldsymbol{x}))$ as the $f_r(x)$ in (5.1) gives the bound in (5.6).

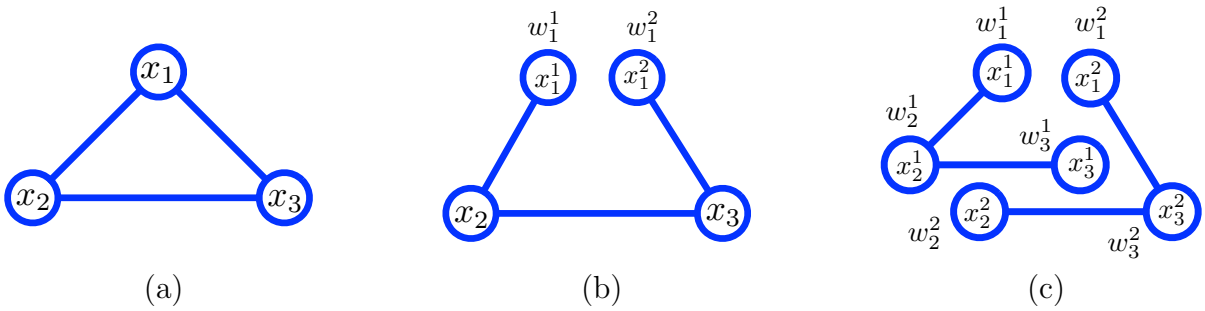


Figure 5.2: Comparing the weighted mini-bucket and TRW primal bounds in Example 5.1. (a) The original Markov graph as used in Example 5.1. (b) A covering tree of the graph in (a) by splitting node x_1. (c) A "overly-split" covering tree that further splits x_2 and x_3. It consists of two connected components, each of which is a spanning trees of the original graph in (a). If the weights on each tree are tied, that is, $w^1 = w_1^1 = w_2^1 = w_3^1$ and $w^2 = w_1^2 = w_2^2 = w_3^2$, then it corresponds to a TRW primal bound over these two spanning trees with weight w^1 and w^2, respectively.

The idea is best illustrated in a toy example as follows.

Example 5.1. *Consider a three-node pairwise model* $p(\boldsymbol{x}) \propto \exp(\theta_{12}(x_1, x_2) + \theta_{23}(x_2, x_3) + \theta_{13}(x_1, x_3))$ *shown in Figure 5.2(a), whose exact partition function equals*

$$Z = \sum_{x_3} \sum_{x_2} \sum_{x_1} \exp(\theta_{12}(x_1, x_2) + \theta_{23}(x_2, x_3) + \theta_{13}(x_1, x_3)).$$

In this example, we demonstrate two weighted mini-bucket bounds on $p(\boldsymbol{x})$ *(as illustrated in Figure 5.2(b)-(c)) and discuss the connection to a TRW primal bound. We only discuss weighted mini-bucket upper bounds, so that all weights are positive.*

(1). Figure 5.2(b) illustrates a weighted mini-bucket procedure that splits variable x_1 *into two copies and forms a "covering tree" (ibound* $= 1$*); a corresponding WMB bound is*

$$\hat{Z} = \sum_{x_3} \sum_{x_2} \sum_{x_1^2}^{w_1^2} \sum_{x_1^1}^{w_1^1} \exp(\theta_{12}(x_1^1, x_2) + \theta_{13}(x_1^2, x_3) + \theta_{23}(x_2, x_3)).$$

where $[w_1^1, w_1^2]$ *are weights on the replicates* $[x_1^1, x_1^2]$*. Here, no splitting is applied to* x_2 *or* x_3*, because splitting* x_1 *breaks the cycle, so that the augmented graph is a tree.*

(2). If we continue to split x_2 and x_3 (although it is unnecessary), we get the "over-split" covering tree shown in Figure 5.2(c), consisting of two connected components, each being a spanning tree of the original graph. A corresponding weighted mini-bucket bound is

$$\hat{Z}' = \sum_{x_3^2}^{w_3^2} \sum_{x_3^1}^{w_3^1} \sum_{x_2^2}^{w_2^2} \sum_{x_2^1}^{w_2^1} \sum_{x_1^2}^{w_1^2} \sum_{x_1^1}^{w_1^1} \exp(\theta_{12}(x_1^1, x_2^1) + \theta_{23}^1(x_2^1, x_3^1) + \theta_{13}(x_1^2, x_3^2) + \theta_{23}^2(x_2^2, x_3^2)),$$

which can be obtained from $\hat{Z}$ by applying Holder's inequality on the sum of x_2 and then of x_3, so that it is at most as tight as $\hat{Z}$, that is, $\hat{Z} \leq \hat{Z}'$. Note that four additional weights $[w_2^1, w_2^2]$ and $[w_3^1, w_3^2]$ are defined on the replicates of x_2 and x_3; the edge potential θ_{23} is also split into θ_{23}^1 and θ_{23}^2, satisfying $\theta_{23}^1 + \theta_{23}^2 = \theta_{23}$.

The values of $[\theta_{23}^1, \theta_{23}^2]$ should be chosen to make the bound $\hat{Z}'$ as tight as possible. Indeed, as we make clear in Section 5.4, there is an optimal value of $[\theta_{23}^1, \theta_{23}^2]$ (for any fixed weights), such that $\hat{Z} = \hat{Z}'$. That is, $\hat{Z}'$ is as good as $\hat{Z}$, once the values of $[\theta_{23}^1, \theta_{23}^2]$ are optimized.

(3). Because the first replicates $\boldsymbol{x}^1 = [x_1^1, x_2^1, x_3^1]$ and the second replicates $\boldsymbol{x}^2 = [x_1^2, x_2^2, x_3^2]$ are disconnected, their powered sums can be performed independently, and $\hat{Z}'$ can be rewritten

$$\hat{Z}' = \Bigg[\sum_{x_3^1}^{w_3^1} \sum_{x_2^1}^{w_2^1} \sum_{x_1^1}^{w_1^1} \exp(\theta_{12}(x_1^1, x_2^1) + \theta_{23}^1(x_2^1, x_3^1)) \Bigg] \cdot \Bigg[\sum_{x_3^2}^{w_3^2} \sum_{x_2^2}^{w_2^2} \sum_{x_1^2}^{w_1^2} \exp(\theta_{13}(x_1^2, x_3^2) + \theta_{23}^2(x_2^2, x_3^2)) \Bigg],$$

If we further assume all the first replicates $\boldsymbol{x}^1 = [x_1^1, x_2^1, x_3^1]$ share the same weight w^1, that is, $w^1 = w_1^1 = w_2^1 = w_3^1$, and similarly $w^2 = w_1^2 = w_2^2 = w_3^2$, then $\hat{Z}'$ reduces to

$$\Bigg[\sum_{\boldsymbol{x}^1}^{w^1} \exp(\theta_{12}(x_1^1, x_2^1) + \theta_{23}^1(x_2^1, x_3^1)) \Bigg] \cdot \Bigg[\sum_{\boldsymbol{x}^2}^{w^2} \exp(\theta_{13}(x_1^2, x_3^2) + \theta_{23}^2(x_2^2, x_3^2)) \Bigg],$$

where $\boldsymbol{x}^1 = [x_1^1, x_2^1, x_3^1]$ are all summed with weight w^1 and $\boldsymbol{x}^2 = [x_1^2, x_2^2, x_3^2]$ all with w^2. Comparing with (5.6), we can readily see that $\hat{Z}'$ is equivalent to a TRW primal bound with the two spanning trees in Figure 5.2(c), with weights w^1 and w^2, respectively.

In conclusion, the TRW primal bounds $\Psi_{trw}(\{\boldsymbol{\theta}^T\}, \{w^T\})$ can be interpreted as weighted mini-bucket bounds with "over-split" covering graphs consisting of a forest of spanning trees, with all replicates in a given tree sharing a single weight. However, a large number of spanning trees is often necessary to obtain tight bounds in practice, making it computationally inefficient to use the TRW primal bound directly.

In addition, the TRW primal bound also requires optimizing the values of $\{\theta^T\}$ and $\{w^T\}$ to get tight bounds. As described in Section 3.2.4, Wainwright et al. [2005] addressed both the representation and optimization issues by instead using the TRW variational (dual) form,

$$\min_{\{w_T\}} \min_{\{\theta^T\}} \Psi_{trw}(\{\theta^T\}, \{w^T\}) = \min_{\{\rho_{ij}\}} \max_{\boldsymbol{\tau} \in \mathbb{L}} \big\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{i \in V} H_i(\tau_i) - \sum_{(ij) \in E} \rho_{ij} I_{ij}(\tau_{ij}) \big\} \tag{5.8}$$

where $\{\rho_{ij} := \sum_{T:\, (ij) \in E^T} w^T\}$ are the edge appearance probabilities associated with $\{w^T\}$. This transforms the joint minimization over $\{\theta^T\}$ and $\{w_T\}$ into a minimax problem on the marginals $\boldsymbol{\tau}$ and $\{\rho_{ij}\}$, whose dimensions do not depend on the number of spanning trees. Wainwright et al. [2005] addressed the inner maximization over $\boldsymbol{\tau}$ using a TRW belief propagation (TRBP) algorithm, and optimized $\{\rho_{ij}\}$ using a conditional gradient descent algorithm. One disadvantage of this approach is that each gradient step on $\{\rho_{ij}\}$ is required to fully optimize $\boldsymbol{\tau}$ (e.g., pass TRBP messages until convergence), making the search over $\{\rho_{ij}\}$ relatively slow. For this reason, in practice implementations often use fixed, usually randomly chosen, weights for TRBP in practice, limiting the power of the TRW bounds.

On the other hand, because our weighted mini-bucket bound $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ is much more compact and computationally efficient, we can instead directly minimize it jointly over $(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$, enabling more efficient algorithms, especially in terms of optimizing over the weights. In the next section, we discuss optimization algorithms on $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$, for both positive and negative weights. Then in Section 5.4 we discuss a similar dual form for $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$, which allows us to compare the TRW bounds and weighted mini-bucket bounds in a more principled way.

Remark. A similar connection can be drawn with our own negative tree reweighted (TRW) BP [Liu and Ihler, 2010], which, by using a reverse Jensens inequality, shows that the same tree reweighted form in (5.6) gives a lower bound on the log-partition function when the weights $\{w^T\}$ take values in $\mathcal{W}^-$, that is, a single spanning tree takes a positive weight, and all the other trees take negative weights. Obviously, the negative TRW bound can be viewed as a "over-split" weighted mini-bucket bound with weights in $\mathcal{W}^-$, where a single positively-weighted spanning tree ($w^T > 0$) includes all the positive replicates of the weighted mini-bucket. Negative tree reweighted BP uses a similar dual optimization to tighten (maximize) the lower bound, and correspondingly has similar problems to standard TRBP.

5.3 Tightening Weighted Mini-Bucket Bounds

The sequential powered sum form $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ in (5.5) characterizes the weighted mini-bucket bound as an explicit function of the augmented natural parameter $\bar{\boldsymbol{\theta}}$ and the weights $\bar{\boldsymbol{w}}$; it is therefore possible to jointly optimize $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$ to get the tightest bounds. This section derives efficient tightening algorithms.

We start with Section 5.3.1, framing the tightening problem for both minimizing the upper bounds and maximizing the lower bounds: minimizing the upper bounds corresponds to a convex optimization, while maximizing the lower bounds gives a challenging non-concave optimization. In Section 5.3.2, we augment the weighted mini-bucket algorithm in Algorithm 5.1 with an additional backward elimination that calculates the derivative $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ w.r.t. $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$, framed as an forward-backward message passing algorithm over the junction tree of the augment model (Algorithm 5.2). With the derivatives calculated, we can in principle call any black box solver to optimize $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$; however, this may not be computationally efficient because each derivative calculation requires a full forward-backward sweep. This motivates us to derive a more efficient algorithm in Section 5.3.3 that updates

$\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$ more frequently, while passing messages. In particular, we show that the optimal $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$ should satisfy *moment matching* and *entropy matching* conditions, respectively, and derive efficient fixed-point or gradient-based updates for $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$ that are inserted into the forward-backward messaging passing updates.

5.3.1 Upper vs. Lower Bounds

To tighten our bounds, we can minimize the upper bound or maximize the lower bound:

$$\text{Upper bound:} \qquad \min_{\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}} \bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) \quad \text{s.t.} \quad \bar{\boldsymbol{\theta}} \in \overline{\Theta}, \bar{\boldsymbol{w}} \in \overline{\mathcal{W}}^{+}, \tag{5.9}$$

$$\text{Lower bound:} \qquad \max_{\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}} \bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) \quad \text{s.t.} \quad \bar{\boldsymbol{\theta}} \in \overline{\Theta}, \bar{\boldsymbol{w}} \in \overline{\mathcal{W}}^{-}, \tag{5.10}$$

where $\overline{\Theta}$ is the set of augmented natural parameters $\bar{\boldsymbol{\theta}} = \{\bar{\theta}_{\bar{\alpha}} \colon \bar{\alpha} \in \bar{\mathcal{I}}\}$ that are consistent with the original model $p(\boldsymbol{x}) \propto \exp(\sum_{\alpha \in \mathcal{I}} \theta_\alpha(\boldsymbol{x}_\alpha))$ in that

$$\overline{\Theta} = \Big\{ \bar{\boldsymbol{\theta}} \colon \quad \sum_{\bar{\alpha} \in \bar{\mathcal{I}}} \bar{\theta}_{\bar{\alpha}}(\bar{\boldsymbol{x}}_{\bar{\alpha}}) = \sum_{\alpha \in \mathcal{I}} \theta_\alpha(\boldsymbol{x}_\alpha), \quad \text{when } \bar{x}_{i^r} = x_i, \quad \forall i, r \Big\}.$$

The $\overline{\mathcal{W}}^{+}$ and $\overline{\mathcal{W}}^{-}$ are precisely the sets of weights $\bar{\boldsymbol{w}}$ that make the Hölder's and reverse Hölder's inequalities in Proposition 5.1 hold, respectively, that is,

$$\overline{\mathcal{W}}^{+} = \Big\{ \bar{\boldsymbol{w}} \colon \sum_r \bar{w}_{i^r} = 1, \quad \bar{w}_{i^r} \geq 0, \quad \forall i, r \Big\},$$

$$\overline{\mathcal{W}}^{-} = \cup \overline{\mathcal{W}}^{-}_{[\kappa_1 \cdots \kappa_n]} \qquad \text{where} \quad \overline{\mathcal{W}}^{-}_{[\kappa_1 \cdots \kappa_n]} = \Big\{ \bar{\boldsymbol{w}} \colon \sum_r \bar{w}_{i^r} = 1, \quad \bar{w}_{i^{\kappa_i}} \geq 0, \quad \bar{w}_{i^r} \leq 0, \forall r \neq \kappa_i \Big\}.$$

In other words, $\overline{\mathcal{W}}^{+}$ requires all the weights to be positive, while $\overline{\mathcal{W}}^{-}$ requires exactly one replicate for each node (or variable) to have positive weight, and is formed as the union of $\overline{\mathcal{W}}^{-}_{[\kappa_1 \cdots \kappa_n]}$, indexed by the array $[\kappa_1 \cdots \kappa_n]$, in which the positive weight is assigned to the κ_i-th replicate for node i.

Because $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ is a joint convex function of $(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ for $\bar{\boldsymbol{w}} \in \overline{\mathcal{W}}^+$ (see Theorem 4.1), and both $\overline{\Theta}$ and $\overline{\mathcal{W}}^+$ are convex sets, minimizing of the upper bound (5.9) is a convex optimization whose global optimum can be calculated efficiently. On the other hand, the maximization of the lower bound (5.10) is not a concave optimization, and it is significantly more challenging to find the global optimum. In particular, $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ is not a concave function for $\bar{\boldsymbol{w}} \in \overline{\mathcal{W}}^-$, and even more importantly, $\overline{\mathcal{W}}^-$ is a non-convex set that is very difficult to search over: it is a union of a combinatorial number ($\prod_i R_i$, where R_i is the number of replicates of i) of disconnected subsets $\overline{\mathcal{W}}^-_{[\kappa_1 \cdots \kappa_n]}$, and finding the global optimum is at least as difficult as solving the combinatorial optimization problem of finding the optimal choice of positive replicates $[\kappa_1^* \cdots \kappa_n^*]$. However, from a practical point of view, it may not be wise to spend too much of our computational resources here: although intellectually appealing to achieve the global optimum by searching among the possible subdomains, it may be more effective to expend computational effort on improving other, more dominant aspects of the approximation, such as the *ibound*.

5.3.2 Weighted Mini-Bucket as Forward-Backward Message Passing

To solve (5.9)-(5.10) efficiently, we need efficient algorithms to calculate $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ and its derivatives. Using the results in Theorem 4.1, we show that the derivatives of $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ w.r.t. $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$ correspond to the marginal probability and conditional entropy of a special "weighted" distribution over $\bar{\boldsymbol{x}}$ constructed based on the augmented model $\bar{p}(\boldsymbol{x}) \propto \exp(\bar{\theta}(\bar{\boldsymbol{x}}))$ and the weights $\bar{\boldsymbol{w}}$. We then present a forward-backward message passing algorithm that calculates $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ via a forward pass identical to weighted mini-bucket in Algorithm 5.1, and calculates the derivatives via an additional backward pass.

We use the results from Theorem 4.1, which we re-state here for convenience:

Theorem 5.1. *(1). The derivatives of* $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ *w.r.t.* $\bar{\boldsymbol{\theta}}$ *and* $\bar{\boldsymbol{w}}$ *are*

$$\frac{\partial}{\partial \bar{\theta}_{\bar{\alpha}}} \bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) = \bar{p}_w(\bar{\boldsymbol{x}}_{\bar{\alpha}}), \tag{5.11}$$

$$\frac{\partial}{\partial \bar{w}_k} \bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) = H_w(\bar{x}_k | \bar{\boldsymbol{x}}_{k+1:\bar{n}} \,;\, \bar{p}_w), \tag{5.12}$$

where $\bar{p}_w(\bar{\boldsymbol{x}}_{\bar{\alpha}})$ *and* $H_w(\bar{x}_k|\bar{\boldsymbol{x}}_{k+1:\bar{n}} \,;\, \bar{p}_w)$ *are marginals and conditional entropies based on a joint distribution* $\bar{p}_w(\bar{\boldsymbol{x}})$*, which is defined via a chain rule expansion as follows:*

$$\bar{p}_w(\bar{\boldsymbol{x}}) = \prod_{k=1}^{\bar{n}} \bar{p}_w(\bar{x}_k | \bar{x}_{k+1:\bar{n}}; \bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}), \qquad \bar{p}_w(\bar{x}_k | \bar{\boldsymbol{x}}_{k+1:\bar{n}}) = \frac{(Z_{i-1}(\bar{\boldsymbol{x}}_{i:\bar{n}} \;;\; \bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}))^{1/\bar{w}_i}}{(Z_i(\bar{\boldsymbol{x}}_{i+1:\bar{n}} \;;\; \bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}))^{1/\bar{w}_i}}, \tag{5.13}$$

where the Z_i *are partial powered sums up to* $\bar{\boldsymbol{x}}_{1:i}$*,*

$$Z_i(\bar{\boldsymbol{x}}_{i+1:\bar{n}} \;;\; \bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) = \sum_{\bar{x}_i}^{\bar{w}_i} \cdots \sum_{\bar{x}_1}^{\bar{w}_1} \exp(\sum_{\bar{\alpha} \in \bar{\mathcal{I}}} \bar{\theta}(\bar{\boldsymbol{x}}_{\bar{\alpha}})).$$

Note that we have $\sum_{\bar{x}_i} Z_{i-1}^{1/\bar{w}_i} = Z_i^{1/\bar{w}_i}$ *by recursion; this makes the conditional distribution* $\bar{p}_w(\bar{x}_i|\bar{\boldsymbol{x}}_{i+1:\bar{n}})$ *properly normalized, that is,* $\sum_{\bar{x}_i} \bar{p}_w(\bar{x}_i|\bar{\boldsymbol{x}}_{i+1:\bar{n}}) = 1$.

(2). If all the elements of $\bar{\boldsymbol{w}}$ *are positive, then* $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ *is a convex function with respect to* $\bar{\boldsymbol{\theta}}$ *and* $\bar{\boldsymbol{w}}$ *jointly.*

Proof. See Theorem 4.1. □

Remark. The $\bar{p}_w(\bar{\boldsymbol{x}})$ in Theorem 5.1 should be distinguished from the augmented model $\bar{p}(\bar{\boldsymbol{x}})$: the augmented model $\bar{p}(\bar{\boldsymbol{x}})$ is obtained directly by splitting variables within the mini-bucket procedure (which is independent of the weights), while the distribution $\bar{p}_w(\bar{\boldsymbol{x}})$ is the result of combining $\bar{p}(\bar{\boldsymbol{x}})$ with the weights $\bar{\boldsymbol{w}}$.

Given the factorization form $\bar{p}(\bar{\boldsymbol{x}}) = \prod_{\bar{\alpha} \in \bar{\mathcal{I}}} \bar{\psi}(\bar{\boldsymbol{x}}_{\bar{\alpha}})$, the calculation of $\bar{p}_w(\bar{\boldsymbol{x}})$ in (5.13) can be organized in a forward-backward message-passing algorithm on the junction tree of

$\bar{p}(\bar{\boldsymbol{x}})$ obtained by triangulation along order $[\bar{x}_{1^1}, \ldots, \bar{x}_{1^{R_i}}, \ldots, \bar{x}_{n^{R_n}}]$. This is shown in Algorithm 5.2: the forward pass is exactly equivalent to weighted mini-bucket in Algorithm 5.1, with $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ equaling the WMB bound; the backward pass computes the marginals of distribution $\bar{p}_w(\bar{\boldsymbol{x}})$. Since the induced width of $\bar{p}(\bar{\boldsymbol{x}})$ is bounded by *ibound*, the complexity of Algorithm 5.2 is $O(\bar{n} \exp(ibound + 1))$. Note that the backward elimination described here can also be applied to regular mini-bucket for an approximation to the marginal probabilities.

Using the values and derivatives as output from Algorithm 5.2, we could directly apply black-box optimization routines to optimize the bound. This approach is particularly promising for tightening the upper bounds with positive weights in (5.9), since it is a convex optimization with simple constraint conditions. However, since $\bar{\Phi}_w$ is calculated using a relatively expensive forward-backward message-passing algorithm, it seems most effective to update $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$ while computing messages. Note that the messages in Algorithm 5.2 converge after one forward-backward propagation for fixed $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$, but updating $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$ can make the messages "loopy". This idea yields Algorithm 5.6, discussed in the next section. A surprisingly powerful version of this message update framework is explored in our experiments (Section 5.5), by non-iteratively optimizing "on the fly" during only a single forward pass.

5.3.3 Tightening via Moment and Entropy Matching

Using the derivative results in Theorem 5.1 and the KKT conditions with respect to the constraints in $\overline{\Theta}$ and $\overline{\mathcal{W}}^+$ (or $\overline{\mathcal{W}}^-$), we show that optimizing $\bar{\boldsymbol{\theta}}$ corresponds to matching the marginals of the replicates, while optimizing $\bar{\boldsymbol{w}}$ corresponds to matching their conditional entropies; both conditions have the intuition of encouraging the replicates to behave similarly. The notion of moment matching has been a well studied for variational methods [e.g., Wainwright et al., 2005], and is closely related to reparameterization [Wainwright et al., 2003a]. However, entropy matching is less well known, and as we show in the experiments, it can significantly improve the accuracy of the bounds, most noticeably the upper bound.

Algorithm 5.2 Calculating the WMB bound, $\bar{\Phi}_w$, and its derivatives (see Theorem 5.1)

Input: Augmented factors $\{\bar{\psi}_{c_k}\colon c_k \in \bar{\mathcal{I}}\}$ and their junction tree $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$ as output from Algorithm 3.5. A weight vector $\bar{\boldsymbol{w}} = [\bar{\boldsymbol{w}}_1, \ldots, \bar{\boldsymbol{w}}_n]$. Elimination order $\bar{o} = [1^1, \ldots, 1^{R_1}, \ldots, n^1, \ldots, n^{R_n}]$.

Output: $\bar{\Phi}_w$ and its derivative as per Theorem 5.1.

1. **Forward:** (*for computing the value of $\bar{\Phi}_w$, equivalent to WMB in Algorithm 5.1*)

$$\text{For } k = 1 \text{ to } \bar{n}: \qquad m_{k\to l}(\bar{\boldsymbol{x}}_{c_l}) = \Big[\sum_{\bar{x}_k} \big(\bar{\psi}_{c_k}(\bar{\boldsymbol{x}}_{c_k}) \prod_{j\in \text{child}(k)} m_{j\to k}(\bar{\boldsymbol{x}}_{c_k})\big)^{1/\bar{w}_k}\Big]^{\bar{w}_k}, \tag{5.14}$$

where $l = \text{pa}_{\bar{\mathcal{G}}}(k)$ is the parent of c_k in $\bar{\mathcal{G}}$ along order $\bar{o}$, and $\text{child}(k) := \{k'\colon k \in \text{pa}_{\bar{\mathcal{G}}}(k')\}$ is the set of its children.

2. **Backward:** (*for computing the gradient of $\bar{\Phi}_w$*)

$$\text{For } k = \bar{n} \text{ to } 1: \qquad m_{l\to k}(\bar{\boldsymbol{x}}_{c_k}) = \Big[\sum_{\bar{\boldsymbol{x}}_{c_l/c_k}} [\bar{\psi}_{c_l}(\bar{\boldsymbol{x}}_{c_l}) m_{\sim l}(\bar{\boldsymbol{x}}_{c_l})]^{1/\bar{w}_l} m_{k\to l}(\bar{\boldsymbol{x}}_{c_l})^{-1/\bar{w}_k}\Big]^{\bar{w}_k}, \tag{5.15}$$

where $l = \text{pa}_{\bar{\mathcal{G}}}(k)$ is the parent of c_k and $m_{\sim l}(\bar{\boldsymbol{x}}_{c_l}) := \prod_{j\in\partial_{\bar{\mathcal{G}}}(l)} m_{j\to l}(\bar{\boldsymbol{x}}_{c_l})$ is the product all messages sent to c_l from its neighborhood $\partial_{\bar{\mathcal{G}}}(l) = \{l'\colon (ll') \in \bar{\mathcal{E}}\}$.
(*Note that backward messages depend on the forward ones, but not vice versa.*)

3. **Return:** The log weighted mini-bucket bound $\bar{\Phi}_w$ is calculated by

$$\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) = \log \prod_{k:\text{pa}_{\bar{\mathcal{G}}}(k)=\emptyset} \sum_{\bar{x}_k}^{\bar{w}_k} \bar{\psi}_{c_k} m_{\sim k}. \tag{5.16}$$

The derivatives of $\bar{\Phi}_w$ are calculated by (5.11)-(5.12), with

$$\bar{p}_w(\bar{\boldsymbol{x}}_{c_k}) \propto \big(\bar{\psi}_{c_k}(\bar{\boldsymbol{x}}_{c_k}) m_{\sim k}(\bar{\boldsymbol{x}}_{c_k})\big)^{1/w_k}. \tag{5.17}$$

Optimizing $\bar{\boldsymbol{\theta}}$ as Moment Matching

We consider the problem of minimizing the WMB upper bound over its parameterization, $\min_{\bar{\boldsymbol{\theta}}\in\overline{\Theta}} \bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$, with fixed weights $\bar{\boldsymbol{w}} > 0$ in this section. Consider a node $i \in V$ and its replicates $\{i^r\}$. For a given initial $\bar{\boldsymbol{\theta}}$, we have the freedom to reallocate factors among the replicates, as discussed in Section 3.1.2, such that the original model is unchanged – e.g., we can update $\bar{\boldsymbol{\theta}}$ to $\bar{\boldsymbol{\theta}} + \sum_r \vartheta_{i^r}(\bar{x}_{i^r})$, where $\{\vartheta_{i^r}(\bar{x}_{i^r})\}$ is a set of factor on $\{\bar{x}_{i^r}\}$ satisfying

Algorithm 5.3 Fixed-point update on $\bar{\boldsymbol{\theta}}$ (or $\bar{\boldsymbol{\psi}} = \exp(\bar{\boldsymbol{\theta}})$) to tighten the WMB bound at node i

1. Calculate marginals $\{\bar{p}_w(\bar{x}_{ir}) : r = 1, \cdots, R_i\}$ based on the messages in Algorithm 5.2,

$$\bar{p}_w(\bar{x}_{ir}) = \sum_{\bar{\boldsymbol{x}}_{c_{ir} \setminus i^r}} \left[\bar{\psi}_{c_{ir}}(\bar{\boldsymbol{x}}_{c_{ir}})\ m_{\sim i^r}(\bar{\boldsymbol{x}}_{c_i^r})\right]^{1/\bar{w}_{ir}}$$

2. Compute a weighted geometric average of the marginals,

$$p_{avg}(x_i) = \Big[\prod_r \bar{p}_w(\bar{x}_{ir} = x_i)^{\bar{w}_{ir}}\Big]^{1/\sum_r \bar{w}_{ir}}$$

3. Update $\bar{\boldsymbol{\psi}}$ (or equivalently, $\bar{\boldsymbol{\theta}} = \log \bar{\boldsymbol{\psi}}$):

$$\bar{\psi}_{c_{ir}}(\bar{\boldsymbol{x}}_{c_{ir}}) \leftarrow \bar{\psi}_{c_{ir}}(\bar{\boldsymbol{x}}_{c_{ir}}) \Big[\frac{p_{avg}(\bar{x}_{ir})}{\bar{p}_w(\bar{x}_{ir})}\Big]^{\bar{w}_{ir}} \tag{5.18}$$

$\sum_r \vartheta_{ir}(\bar{x}_{ir} = x_i) = 0$, for any r and x_i. The problem of finding an optimal ϑ-update can be framed as

$$\min_{\vartheta}\ \bar{\Phi}_w(\bar{\boldsymbol{\theta}} + \sum_r \vartheta_{ir},\ \bar{\boldsymbol{w}}), \quad \text{s.t.} \quad \sum_r \vartheta_{ir}(\bar{x}_{ir} = x_i) = 0, \quad \forall x_i \in \mathcal{X}_i.$$

Using Theorem 5.1, one can show that the KKT condition of $\{\vartheta_{ir}\}$ is

$$\bar{p}_w(\bar{x}_{ir} = x_i) = \lambda(x_i), \quad \text{for } \forall r = 1, \ldots, R_i, \quad x_i \in \mathcal{X}_i \tag{5.19}$$

where $\lambda(x_i)$ is the Lagrange multiplier for the constraint $\sum_r \vartheta_{ir}(\bar{x}_{ir} = x_i) = 0$. This parallels the *moment matching*, or *marginal matching* condition in Wainwright et al. [2005]. Intuitively speaking, it implies that the marginals $\{\bar{p}_w(\bar{x}_{ir})\}$ of the replicates $\{\bar{x}_{ir}\}$ should be equal. Since the original "correct" model corresponds to forcing all the replicates $\{\bar{x}_{ir}\}$ to be equal, the moment matching condition can be viewed as a "relaxation" of this exact equality, so that they are only equal in distribution. Note that satisfying (5.19) guarantees global optimality for upper bounds with positive weights due to convexity; it is a necessary, but usually not sufficient, condition for optimality in the non-convex, lower bound setting.

Updating $\bar{\boldsymbol{\theta}}$. In order to update $\bar{\boldsymbol{\theta}}$ to achieve the moment matching condition (5.19), we can use the fixed-point update given in Algorithm 5.3, where an "average" marginal $p_{avg}(x_i)$ is computed by taking the (weighted) geometric mean of marginals $p_{avg}(x_i)$ of the replicates $\{\bar{p}_w(\bar{x}_{ir})\}$, and then adjusting $\bar{\boldsymbol{\theta}}$ to correct for the difference between each $\bar{p}_w(\bar{x}_{ir} = x_i)$ and the mean $p_{avg}(x_i)$. It is easy to show that this update keeps $\bar{\boldsymbol{\theta}}$ inside $\overline{\Theta}$, the set of augmented models consistent with the original model, and that its fixed point satisfies the moment matching condition (5.19).

Many other update choices are equally valid. For example, applying the projected gradient algorithm to update $\bar{\boldsymbol{\theta}}$ results in a particularly simple form:

$$\bar{\theta}_{c_{ir}}(\bar{\boldsymbol{x}}_{c_{ir}}) \leftarrow \bar{\theta}_{c_{ir}}(\bar{\boldsymbol{x}}_{c_{ir}}) + \mu \left[\bar{p}_w(\bar{x}_{ir}) - \frac{1}{R_i} \sum_r \bar{p}_w(\bar{x}_{ir}) \right]$$

where each $\bar{\theta}_{c_{ir}}$ is adjusted according to difference between $\bar{p}_w(\bar{x}_{ir})$ and their arithmetic mean; the μ is the step size. Other types of optimization algorithms can also be used. For example, applying second-order, Newton-like methods could be used to improve the convergence speed.

Remarks. (1). Most belief propagation based algorithms, including loopy BP and tree reweighted BP, can be treated as reparameterizing the distribution to achieve some notion of moment matching. For example, the tree reweighted BP message passing is derived to solve the variational optimization over $\boldsymbol{\tau}$ on the right side of (5.8), which is equivalent to optimizing $\{\boldsymbol{\theta}^T\}$ on its left side. In this sense, our updates to $\bar{\boldsymbol{\theta}}$ are effectively message passing for our framework; we revisit these connections in Section 5.3.3 and Section 5.4.

(2). In addition to sharing replicates of single variables x_i, sometimes the different mini-buckets may also share larger sets of replicated variables, for which $\bar{\boldsymbol{\theta}}$ can also be reallocated in the same way.

Optimizing $\bar{w}$ as Entropy Matching

We next consider the problem of optimizing the weights $\bar{\boldsymbol{w}}$, with fixed augmented model parameter $\bar{\boldsymbol{\theta}}$. By the KKT conditions and Theorem 5.1, we can show that the optimal weights $\bar{\boldsymbol{w}}$, for both the upper and lower bounds in (5.9)-(5.10), should satisfy an *entropy matching* condition:

$$\bar{w}_{i^r}(H_{i^r|\prec} - H_{i|\prec}) = 0, \ \forall i^r \in \bar{V}, \tag{5.20}$$

where $H_{i^r|\prec} = H(\bar{x}_{i^r}|\bar{\boldsymbol{x}}_{\pi_{i^r}} \ ; \ \bar{p}_w)$ is the conditional entropy of the rth replicate, as per (5.12), and $H_{i|\prec} = \sum_r \bar{w}_{i^r} H_{i^r|\prec}$ is the (weighted) average conditional entropy over the replicates. Thus at a stationary point, either the weight $\bar{w}_{i^r}$ is zero, or the replicate's conditional entropy equals the average. See Appendix A.1 for the proof. Although the entropy matching condition applies in both cases, the sets $\overline{\mathcal{W}}^+$ and $\overline{\mathcal{W}}^-$ associated with upper and lower bounds are very different, and thus their update algorithms should be considered separately.

Updating Positive Weights. To update $\{\bar{w}_{i^r}\}$ in $\overline{\mathcal{W}}^+$, one straightforward approach is to transform the optimization to an unconstrained problem by representing the weights as $\bar{w}_{i^r} = \exp(u_{i^r})/\sum_r \exp(u_{i^r})$, for $u_{i^r} \in \mathbb{R}$. The gradient of $\bar{\Phi}_w$ with respect to the transformed weights u_i^r is

$$\frac{\partial \bar{\Phi}_w}{\partial u_{i^r}} = \bar{w}_{i^r}(H_{i^r|\prec} - H_{i|\prec}).$$

Taking a gradient step on u_{i^r} and updating $\bar{w}_{i^r}$ correspondingly yields the log-gradient algorithm given in Algorithm 5.4. This update ensures that the weights $\bar{\boldsymbol{w}}$ remain in $\mathcal{W}^+$, and its fixed point satisfies the KKT condition (5.20).

As with $\bar{\boldsymbol{\theta}}$, other optimization methods of $\bar{\boldsymbol{w}}$ are equally valid. For example, we can derive a conditional gradient update as,

$$r^* = \arg\min_r H_{i^r|\prec}, \qquad \bar{w}_{i^r} \to \bar{w}_{i^r} + \mu(\mathbf{1}[r = r^*] - \bar{w}_{i^r}),$$

Algorithm 5.4 Log-gradient update of weights $\bar{\boldsymbol{w}}_i = \{\bar{w}_{ir}\}$ at node i for minimizing the WMB upper bound

1. Compute the conditional entropy $H_{ir|\prec} = H(\bar{x}_{ir}|\bar{\boldsymbol{x}}_{\mathrm{pa}_{\bar{\mathcal{G}}}(ir)}\ ;\ \bar{p}_w)$, for all replicates $r \in 1,\ldots,R_i$ based on the marginals $\{\bar{p}_w(\bar{\boldsymbol{x}}_{c_{ir}})\}$ calculated from Algorithm 5.2.

2. Update weights:

$$\begin{aligned} &\bar{w}_{ir} \leftarrow \bar{w}_{ir} \exp\Big[-\mu\, \bar{w}_{ir}\big(H_{ir|\prec} - \sum_r \bar{w}_{ir} H_{ir|\prec}\big)\Big], && \forall r = 1,\ldots,R_i, \\ &\bar{w}_{ir} \leftarrow \bar{w}_{ir} / \textstyle\sum_r \bar{w}_{ir}, && \forall r = 1,\ldots,R_i. \end{aligned} \tag{5.21}$$

where μ is a step size.

where $\delta[\cdot]$ is the indicator function, and μ is a step size, which is required to satisfy $0 < \mu < 1$ in this case. Note that this update simply finds the replicate with lowest conditional entropy, and increases its weight by an amount specified by the step size. In Section 5.4, we elaborate on the close connection between our WMB bound and tree-reweighted belief propagation; this conditional gradient update is roughly analagous to the weight update for TRBP given in Wainwright et al. [2005]. However, in our experiments we use the log-gradient update, as it appeared to converge better than the conditional gradient method.

Updating Negative Weights. Updating the weights in $\overline{\mathcal{W}}^-$ is considerably more challenging, mainly due to the difficulty of selecting the choice of positive replicates; this is essentially equivalent to jumping between the various disconnected subdomains that comprise $\overline{\mathcal{W}}^-$. However, it is possible to find local optima when the positive replicate is fixed. Assume κ_i is the unique replicate of i that is assigned to have a positive weight ($\bar{w}_{i\kappa_i} \geq 0$). Let us apply the transformation of variables, $\bar{w}_{ir} = \exp(u_{ir})$ for $r \neq \kappa_i$ and $\bar{w}_{i\kappa_i} = 1 - \sum_{r\neq\kappa_i} \bar{w}_{ir}$, where $u_{ir} \in \mathbb{R}$. Taking a gradient step on u_{ir} and updating the original weights $\bar{w}_{ir}$ correspondingly, we get the log-gradient update shown in Algorithm 5.5, which is analogous to Algorithm 5.4 for optimizing the upper bound.

To decide which replicate should be assigned positive weight, we can run the local search algorithm for each possible choice of κ_i, and then select the best one. Note that this corre-

Algorithm 5.5 Log-gradient update of weights $\bar{\boldsymbol{w}}_i = \{\bar{w}_{i^r}\}$ at node i for maximizing the WMB lower bound (with the unique positive replicate fixed to be i^{κ_i})

1. Compute the conditional entropy $H_{i^r|\prec} = H(\bar{x}_{i^r}|\bar{\boldsymbol{x}}_{\mathrm{pa}_{\bar{\mathcal{G}}}(i^r)}\,;\,\bar{p}_w)$ for all replicates i^r, based on the marginals $\{\bar{p}_w(\bar{\boldsymbol{x}}_{c_{i^r}})\}$ calculated from Algorithm 5.2.

2. Update weights:

$$\begin{aligned} &\bar{w}_{i^r} \leftarrow \bar{w}_{i^r} \exp\big(\epsilon\, \bar{w}_{i^r}(H_{i^r|\prec} - H_{i^{\kappa_i}|\prec})\big), \qquad \text{for all } r \neq \kappa_i \\ &\bar{w}_{i^{\kappa_i}} \leftarrow 1 - \textstyle\sum_{r\neq\kappa_i} \bar{w}_{i^r}. \end{aligned} \tag{5.22}$$

where ϵ is a step size.

sponds to a greedy update on κ_i, keeping the positive weight choice $\{\kappa_{i'} : i' \neq i\}$ of the other nodes fixed.

Putting it All Together

The updates of $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$ can be inserted into the forward-backward calculations in Algorithm 5.2 in different ways, leading to different optimization algorithms; Algorithm 5.6 outlines a general framework, with two significant variants depending on whether the backward messages are pre-updated or not. Note that the value of the backward messages $m_{\mathrm{pa}_{\bar{\mathcal{G}}}(i^r)\to i^r}$ depend on the current forward messages $m_{i^r\to\mathrm{pa}_{\bar{\mathcal{G}}}(i^r)}$ as well as the subsequent backward messages (see (5.15)), all of which become invalid when the earlier forward messages are updated. To make the backward messages $m_{\mathrm{pa}_{\bar{\mathcal{G}}}(i^r)\to i^r}$ valid again, one needs to update $m_{i^r\to\mathrm{pa}_{\bar{\mathcal{G}}}(i^r)}$ as well as all forward, and then backward, messages on the entire graph below $\{\bar{x}_i^r\}$; in general, this is computationally expensive and impractical, and the old message values are simply used anyway. The optional step of Algorithm 5.6 implements an approximate version of this sweep, which pre-updates $m_{i^r\to\mathrm{pa}_{\bar{\mathcal{G}}}(i^r)}$ and then immediately $m_{\mathrm{pa}_{\bar{\mathcal{G}}}(i^r)\to i^r}$, but temporarily ignores any additional changes in messages in the lower part of the graph. Note that in contrast, it is always unnecessary to pre-update the forward messages, because the value of forward messages does not depend on the backward messages, as shown in (5.14). See Figure 5.3 for an illustration of the various updates.

Algorithm 5.6 Tightening the weighted mini-bucket bound $\bar{\Phi}_w$

Input: The original graphical model $p(\boldsymbol{x}) \propto \prod_{\alpha \in \mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha)$; the clique set $\bar{\mathcal{I}} = \{c_{i^r}\}$ of an augmented model $\bar{p}_w(\bar{\boldsymbol{x}})$ and its junction tree $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$ as outputted from Algorithm 3.4.
Output: An optimal $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$, and the weighted mini-bucket bound $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$.
Initialize $\bar{\boldsymbol{\theta}}$, $\bar{\boldsymbol{w}}$ and messages $\{m_{i \to j}\}$.
repeat
  for $i \leftarrow 1$ to n **do**
    pre-update backward messages (optional)**:**
    update $m_{i^r \to \mathrm{pa}_{\bar{\mathcal{G}}}(i^r)}$ and then $m_{\mathrm{pa}_{\bar{\mathcal{G}}}(i^r) \to i^r}$ by (5.14)-(5.15) for all replicates i^r of i.
    Reallocate / reweight:
    Update $\bar{\boldsymbol{\theta}}$ and/or $\bar{\boldsymbol{w}}$ by the fixed point step in Algorithm 5.3, the log-gradient step in Algorithm 5.4 or 5.5, or other methods.
    Update forward messages:
    Update $m_{i^r \to \mathrm{pa}_{\bar{\mathcal{G}}}(i^r)}$ by (5.14) for all replicates i^r of i.
  end for
  Calculate the bound $\bar{\Phi}_w$ by (5.16).
  for $i \leftarrow n$ to 1 **do**
    Reallocate / reweight:
    Update $\bar{\boldsymbol{\theta}}$ and/or $\bar{\boldsymbol{w}}$ by the fixed point step in Algorithm 5.3, the log-gradient step in Algorithm 5.4 or 5.5, or other methods.
    Update backward messages:
    Update $m_{\mathrm{pa}_{\bar{\mathcal{G}}}(i^r) \to i^r}$ by (5.15) for all replicates of i.
  end for
until stopping criterion satisfied.

Remarks. (1). As we discussed earlier, the update on $\bar{\boldsymbol{\theta}}$ is closely related to the message updates in TRBP. In fact, we can show that for pairwise models, Algorithm 5.6 with only $\bar{\boldsymbol{\theta}}$-updates (without the reweighting and the optional pre-update) corresponds to the sequential schedule of TRBP [Wainwright et al., 2005] (to see this, substitute (5.18) into (5.14) and (5.15), and cancel $\bar{\psi}_{c_k}$). However, with the optional pre-update step, or the weight updates, it is not equivalent to existing algorithms. In practice, we find that the pre-update step can sometimes help convergence, but not always. See Section 5.4 for more discussion on the connection to TRBP.

(2). Similarly, our update to the weights $\bar{\boldsymbol{w}}$ corresponds to the optimization of the tree weights and the edge appearance probabilities in TRBP. However, TRBP optimizes their

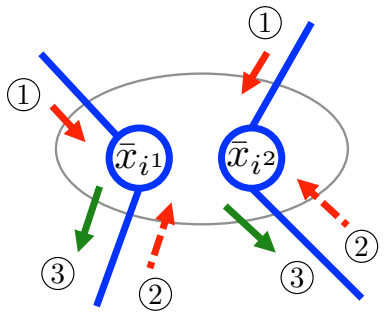


Figure 5.3: Illustrating Algorithm 5.6. The updates of $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$ depend on both the forward and backward incoming messages (① and ②). The backward messages ② depend on the current forward messages ③ as well as the subsequent backward messages (not shown in the picture), all of which depend on the earlier forward messages ①. Therefore, in the forward phase, both ② and ③ become outdated due to the update on ①. To make ② become valid again, one needs to pre-update ③ as well as all the forward and then backward messages on entire graph below $\{\bar{x}_i^r\}$. Since this is impractically expensive, the optional step of Algorithm 5.6 implements an approximate version of this process by only pre-updating $m_{i^r \to \mathrm{pa}_{\bar{\mathcal{G}}}(i^r)}$ and then immediately $m_{\mathrm{pa}_{\bar{\mathcal{G}}}(i^r) \to i^r}$, ignoring all the messages in the lower part of the graph. On the other hand, since the forward messages do not depend on the backward messages, a similar pre-update would not have any effect in the backward phase.

weights on the dual objective, and uses a double-loop weight optimization algorithm that is required to fully optimize $\bar{\boldsymbol{\theta}}$ before taking a gradient update on their weights (because the gradient is correct only when $\bar{\boldsymbol{\theta}}$ is fully optimized). Our algorithm has the significant advantage that we can simultaneously update $\bar{\boldsymbol{w}}$ and $\bar{\boldsymbol{\theta}}$ by directly optimizing the bound $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$, which is a jointly convex function on $(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$. This greatly enhances our ability to search for better weights, which as we show in our experiments can be surprisingly important.

(3). Standard mini-bucket can be viewed as a special case of our framework, which takes weights only on the vertices of $\overline{\mathcal{W}}^+$ and $\overline{\mathcal{W}}^-$. In this case, selecting weights corresponds to the combinatorial problem of selecting a single replicate (or mini-bucket) for each node on which the sum operator ($w = 1$) will be applied in (3.7). This perspective has different implications for upper bounds versus lower bounds due to the convexity/non-convexity dichotomy:

1. For minimizing upper bounds, relaxing to continuous weights makes the selection simpler, more accurate, and efficient due to the convexity of the weight optimization.

Although finding the combinatorial solution may be challenging, optimizing over the continuous weights is relatively simple, and must be at least as good as any vertex.

2. In contrast, for lower bounds, using continuous weights may still give more accuracy, but may increase the computational difficulty to some extent: it imposes an additional local optimization problem inside each subdomain (corresponding to the unique replicate eliminated by sum in regular MBE) in order to evaluate each discrete choice. Although we can obtain a better solution within a fixed subdomain using the negative weight updates in Algorithm 5.5, selecting the best subdomain remains combinatorial and difficult. It remains an open direction to derive more efficient methods here, possibly by incoporating our entropy matching condition into the search heuristic.

Choosing the Covering Structure

The tightness of the bound also depends on the structure of the covering graph (that is, the clique set $\bar{\mathcal{I}}$ of the augmented model $\bar{p}(\bar{\boldsymbol{x}})$), which stems from the bucket partitioning strategy and elimination order by following Algorithm 3.4. These elements are difficult to optimize, although some heuristics exist [e.g., Rollon and Dechter, 2010]. In this work, we use the simplest scope-based heuristic [Dechter and Rish, 2003], which greedily attempts to minimize the number of splits (replicates) that are required.

For completeness, we introduce the method briefly. We first sort the factors in decreasing order of their number of variables and set the mini-buckets to be empty. Then, for each factor, we either put it into an existing mini-bucket if the *ibound* is not violated, or put it into a new empty mini-bucket. To increase the degrees of freedom of $\bar{\boldsymbol{\theta}}$ over which we can optimize, we also add a *refill* phase at the end, which goes through each factor and each mini-bucket other than the one to which it is assigned, and increases the scope of the mini-bucket by the factor's domain (as if the factor had also been added to that mini-bucket) if doing so does not violate the *ibound*.

5.4 Duality and Connection to TRBP

For positive weights $\bar{\boldsymbol{w}} > 0$, Theorem 4.1 gives a variational representation for the weighted mini-bucket bound $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$. This allows us to draw a connection to convex variational methods such as tree-reweighted BP and conditional entropy decomposition.

To set up, let $\bar{\mathcal{G}}$ be the junction tree of $\bar{p}(\bar{\boldsymbol{x}})$ constructed by triangulating along order $[\bar{x}_{1^1}, \ldots, \bar{x}_{1^{R_1}}, \cdots, \bar{x}_{n^{R_n}}]$; note that the clusters of $\bar{\mathcal{G}}$ should equal $\bar{\mathcal{I}} = \{c_{i^r}\}$ as obtained from Algorithm 3.4. Let Γ be the mapping from the replicates to their corresponding original nodes, that is, $\Gamma(i^r) = i$ for $\forall i^r$. Then we can obtain a junction graph $\mathcal{G} := \Gamma(\bar{\mathcal{G}})$ on the original model $p(\boldsymbol{x})$, by applying Γ on the clusters and separators of $\bar{\mathcal{G}}$ and connecting the clusters associated with different replicates with proper separators so that the running intersection property is satisfied – In Mateescu et al. [2010], this is done by connecting the clusters of different replicates of x_i with separators that consist of just x_i, and form a chain or tree structure[2]. Note that while $\bar{\mathcal{G}}$ is a junction tree on the augmented graph, this will generally result in a loopy junction graph $\mathcal{G}$ over the original variables. Let $\mathbb{L}(\bar{\mathcal{G}})$ and $\mathbb{L}(\mathcal{G})$ be the locally consistent polytopes on $\bar{\mathcal{G}}$ and $\mathcal{G}$, respectively. The following Theorem is a direct result of Theorem 4.1.

Theorem 5.2. (1). *For fixed $\bar{\boldsymbol{w}} > 0$, $\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$ as a convex function of $\bar{\boldsymbol{\theta}}$ has dual representation,*

$$\bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) = \max_{\bar{\boldsymbol{\tau}} \in \mathbb{L}(\bar{\mathcal{G}})} \big\{ \langle \bar{\boldsymbol{\tau}}, \bar{\boldsymbol{\theta}} \rangle + \sum_{k=1}^{\bar{n}} \bar{w}_k H_w(\bar{x}_k | \bar{\boldsymbol{x}}_{c_k \setminus \{k\}}, \bar{\boldsymbol{\tau}}) \big\}, \tag{5.23}$$

where $H_w(\bar{x}_k | \bar{\boldsymbol{x}}_{c_k \setminus \{k\}}\,; \bar{\boldsymbol{\tau}})$ is the conditional entropy under $\bar{\boldsymbol{\tau}}$. The maximum is obtained iff $\bar{\tau}(\bar{\boldsymbol{x}}) = \bar{p}_w(\bar{\boldsymbol{x}} | \bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}})$.

[2]Because the clusters of different replicates of x_i may have overlapping variables other than x_i, the separators defined here may be only a subset of the intersections of the clusters, slightly generalizing the definition of junction graphs given in Section 3.1.3; see Mateescu et al. [2010] for more details.

(2). *For fixed* $\bar{\boldsymbol{w}} > 0$*, we have*

$$\min_{\bar{\boldsymbol{\theta}} \in \overline{\Theta}} \bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) = \max_{\boldsymbol{\tau} \in \mathbb{L}(\mathcal{G})} \big\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + \sum_{k=1}^{\bar{n}} \bar{w}_k H(x_{\Gamma(k)} | \boldsymbol{x}_{\Gamma(c_k \setminus \{k\})}, \boldsymbol{\tau}) \big\}. \tag{5.24}$$

i.e., the optimal choice of $\bar{\boldsymbol{\theta}}$ *has a dual form representable in terms of the polytope, marginals, and conditional entropies defined on the cliques of the junction graph* $\mathcal{G}$*.*

Proof (sketch). (1) is a direct result of Theorem 4.1(1) when applied on $\bar{p}(\bar{\boldsymbol{x}})$. For (2), we use the result of (1),

$$\begin{aligned} \min_{\bar{\boldsymbol{\theta}} \in \overline{\Theta}} \bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) &= \min_{\bar{\boldsymbol{\theta}} \in \overline{\Theta}} \max_{\bar{\boldsymbol{\tau}} \in \mathbb{L}(\bar{\mathcal{G}})} \big\{ \langle \bar{\boldsymbol{\tau}}, \bar{\boldsymbol{\theta}} \rangle + \sum_{k=1}^{\bar{n}} \bar{w}_k H_w(\bar{x}_k | \bar{\boldsymbol{x}}_{c_k \setminus \{k\}}, \bar{\boldsymbol{\tau}}) \big\} \\ &= \max_{\bar{\boldsymbol{\tau}} \in \mathbb{L}(\bar{\mathcal{G}})} \min_{\bar{\boldsymbol{\theta}} \in \overline{\Theta}} \big\{ \langle \bar{\boldsymbol{\tau}}, \bar{\boldsymbol{\theta}} \rangle + \sum_{k=1}^{\bar{n}} \bar{w}_k H_w(\bar{x}_k | \bar{\boldsymbol{x}}_{c_k \setminus \{k\}}, \bar{\boldsymbol{\tau}}) \big\}. \end{aligned}$$

where the max and min commute due to strong duality. The result (5.24) then follows by solving the inner minimization on $\bar{\boldsymbol{\theta}}$ (which is linear program). See Appendix A.2 for the detailed proof. □

Note that $\mathbb{L}(\bar{\mathcal{G}})$ is the locally consistent polytope over the replicated variables $\bar{\boldsymbol{x}}$, while $\mathbb{L}(\mathcal{G})$ is over the original variables $\boldsymbol{x}$. Theorem 5.2(2) gives a dual approach for finding the $\bar{\boldsymbol{\theta}}$-optimal bound. It has the form of a weighted sum of conditional entropies, and as we show in the sequel, it is equivalent to conditional entropy decomposition (CED) with certain CED-subgraphs that are all consistent with the same elimination order o; furthermore, if these conditional entropies are consistent with some hyper-tree order (which is always true for *ibound* = 1), it will also be a form of generalized TRBP. This suggests that the $\bar{\boldsymbol{\theta}}$-optimal bound in (5.24) is a restricted form of CED. However, just as CED uses a weighted combination of CED-subgraphs, we could similarly elect to use a weighted combination of more than one covering tree to achieve as tight a bound as CED. In this work, however, we

focus on the expressive power of a single covering tree, which enables a simple and efficient *primal* bound as well as the use of existing heuristics for clique choice. See the next Section for more discussion on the connection between CED and WMB.

5.4.1 Covering Trees vs. Spanning Trees

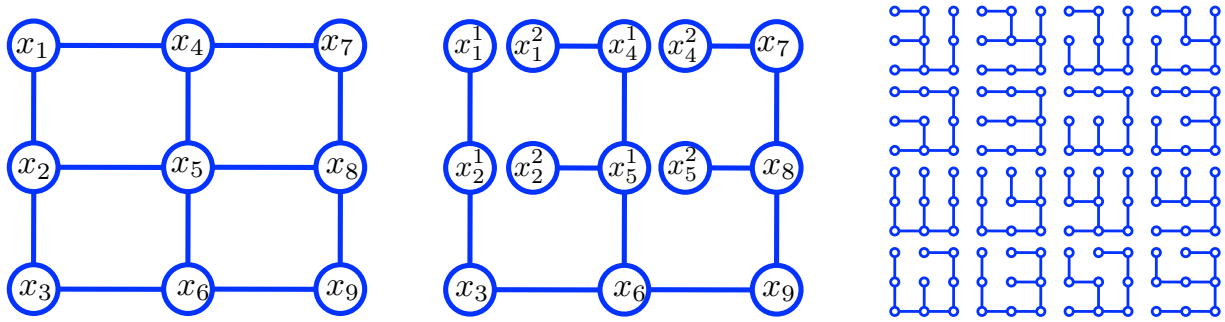
Although the $\bar{\boldsymbol{\theta}}$-optimal bound in (5.24) is equivalent to that of generalized TRBP or CED, WMB has significant practical advantages. First, it gives a concise and novel primal bound. Note that the primal form of TRBP is intractable unless the number of spanning trees is small. In the general CED, the CED-subgraphs corresponds to the spanning trees of TRBP, so CED is also intractable unless the number of CED-subgraphs is small; Globerson and Jaakkola [2007a] derived a primal form of CED based on geometric programming, but it is generally computationally infeasible. In contrast, WMB is able to use only a few parameters to capture the equivalent of a large collection of spanning trees or CED-subgraphs.

Let $\mathcal{B}_{\text{wmb}}(o)$ be the set of $\bar{\boldsymbol{\theta}}$-optimal bounds as shown in Theorem 5.2(2) by an arbitrary covering structure and weights $\bar{\boldsymbol{w}}$ with elimination order o, and $\mathcal{B}_{\text{ced}}(o)$ be the set of CED bounds with arbitrary CED-subgraphs and arbitrary weights all assigned with elimination order o. The following theorem clarifies the equivalence between WMB and CED.

Theorem 5.3. *$\mathcal{B}_{wmb}(o) = \mathcal{B}_{ced}(o)$, that is, the set of weighted mini-bucket $\bar{\boldsymbol{\theta}}$-optimal bounds with elimination order o is equivalent to the set of CED bounds with all CED-subgraphs assigned with elimination order o.*

Proof. Simply compare the WMB dual form (5.24) and the CED bound (3.42). □

We illustrate Theorem 5.3 with a simple example in Figure 5.4. For a 3×3 grid, the covering tree in panel (c) gives a bound equivalent to that of the collection of 16 spanning trees in panel (d). In general the covering tree can have many fewer degrees of freedom than its



(a) (b) (c)

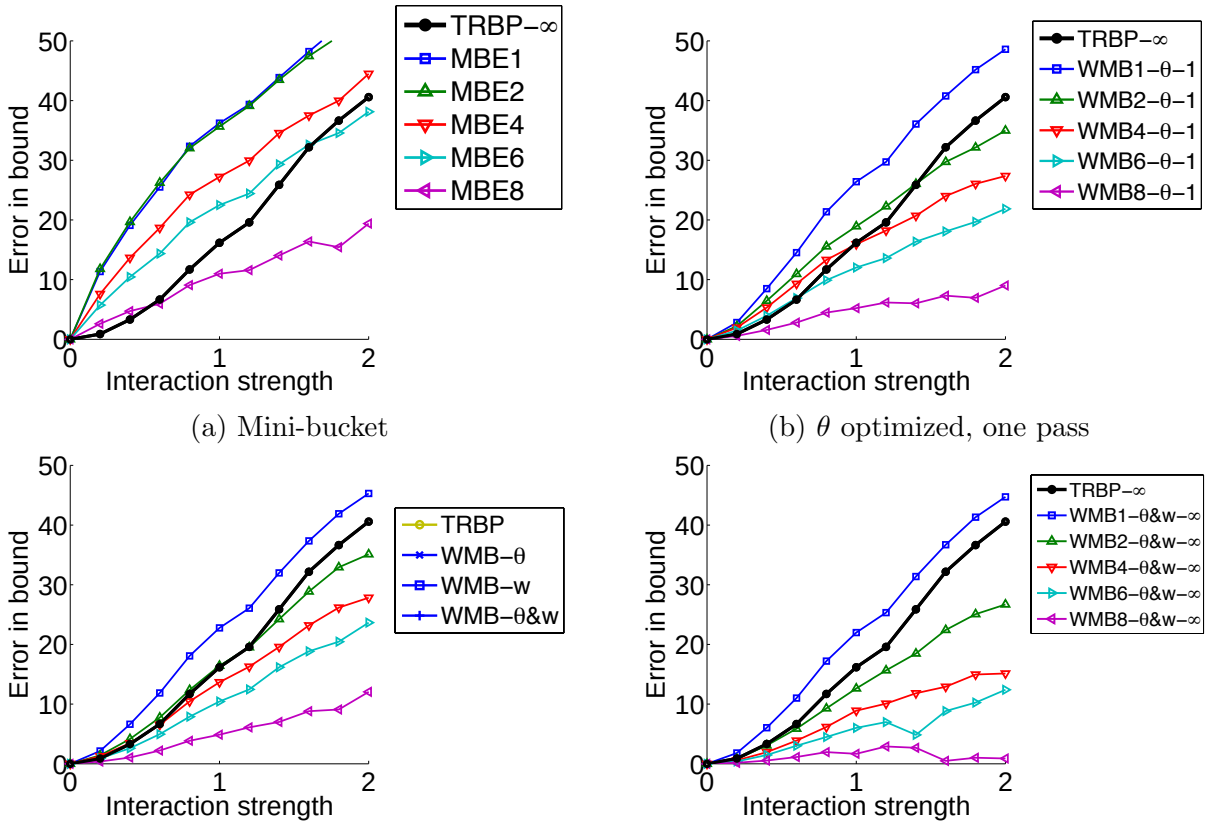
Figure 5.4: Covering trees vs. spanning trees. (a) A 3×3 grid. (b) A covering tree ($ibound = 1$). (c) The set of spanning trees that is equivalent to the covering tree in (b); a single covering tree can have the same representational power as a large collection of spanning trees (here, 16).

corresponding collection of trees or orders, e.g., the weights on the covering tree have only 4 degrees of freedom, while the weights on the spanning trees have 15 degrees of freedom.

Not only is the covering graph representation more efficient, it also improves the pre-convergence bound quality. In particular, the "extra" degrees of freedom in TRBP/CED can only loosen the bound; their removal corresponds to enforcing that they take on optimal values. These advantages improve our ability to represent and optimize a primal form of the bound with many spanning trees. Additionally, advantages over the dual form include (1) easily optimizing $\bar{\boldsymbol{w}}$ (since in TRBP, the gradient is technically valid only at convergence), (2) the ability to flexibly balance optimizing $\bar{\boldsymbol{\theta}}$, $\bar{\boldsymbol{w}}$, or increasing the *ibound*, and (3) providing a valid bound at any point, i.e., an *any-time* property.

5.5 Experiments

We tested our weighted mini-bucket approach using both positive or negative weights on synthetic Ising model networks, and on a number of real-world linkage analysis models from the UAI'08 competition to show its effectiveness compared to standard TRBP, mini-bucket and mean field.



(a) Mini-bucket

(b) θ optimized, one pass

(c) w optimized, one pass

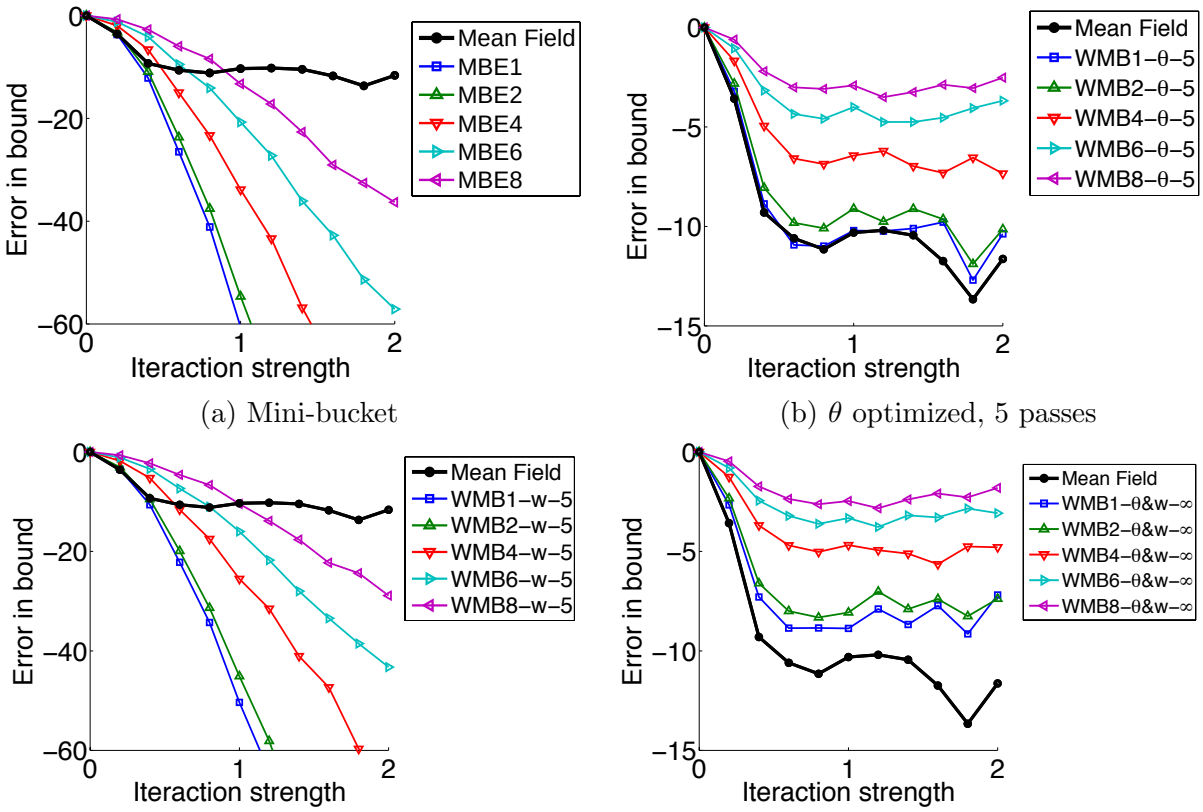
(d) Both fully optimized

Figure 5.5: Upper bound results on 10×10 Ising model with mixed interactions. (a) Mini-bucket is often worse than TRBP unless a high *ibound* is used, but (b),(c) weighted mini-bucket is often better even with only a single, forward pass. (d) Fully optimized WMB is slightly looser than TRBP for *ibound* $= 1$, but quickly surpasses TRBP as *ibound* increases. See also Figure 5.7.

Ising models. We generated random 10×10 Ising models (binary pairwise grids),

$$p(x|\theta) = \exp\Big[\sum_{i \in V} \theta_i x_i + \sum_{(i,j) \in E} \theta_{ij} x_i x_j - \Phi(\theta)\Big],$$

where $x_i \in \{-1, 1\}$ and we draw $\theta_i \sim \mathcal{N}(0, 0.1)$, $\theta_{ij} \sim \mathcal{N}(0, \sigma)$, while varying the strength $\sigma \in [0, 2]$. This distribution of parameter values gives rise to "mixed" interactions; the relative performance of the various methods on purely-attractive models (θ_{ij} positive) was similar and has been omitted.



(a) Mini-bucket

(b) θ optimized, 5 passes

(c) w optimized, 5 passes

(d) Both fully optimized

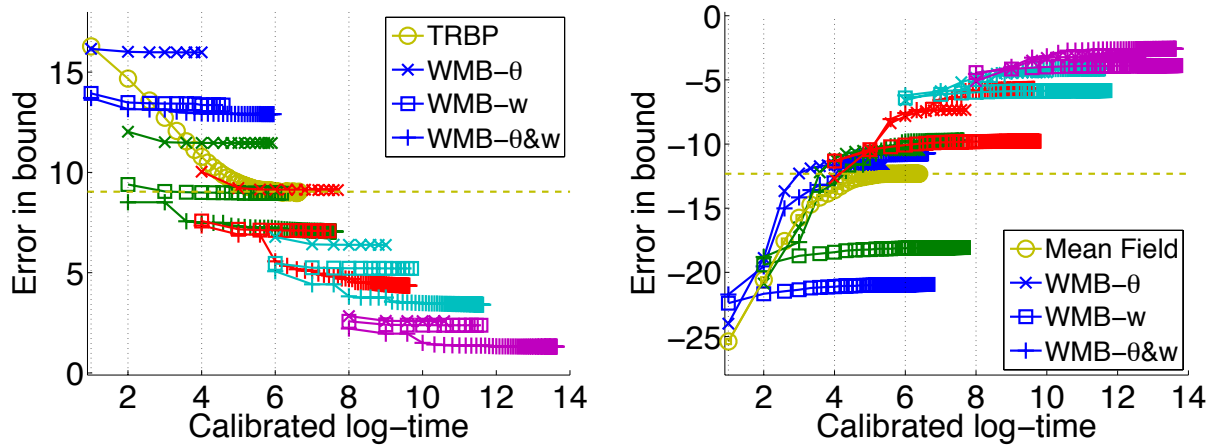
Figure 5.6: Lower bound results on 10×10 Ising model with mixed interactions. (a) Mini-bucket is often worse than the naïve mean field, and (c) optimizing weights seems does not help significantly in this case. But (b) optimizing θ greatly improves the bound within the first 5 iterations. (d) Fully optimized WMB outperforms naïve mean field in all the cases. See also Figure 5.7.

For WMB with either positive or negative weights, we use the column-first elimination order on the variables and scope-based heuristic for splitting as described in Section 5.3.3. The parameterization $\bar{\boldsymbol{\theta}}$ is initialized by splitting θ across replicates according to $\theta_{ir} = w_{ir}\theta_i$; this guarantees that $\sum_r \theta_{ir} = \theta_i$ since $\sum_r w_{ir} = 1$. For the upper bound we initialize the positive weights uniformly. For the lower bound, we assign the first replicate to have positive weight $w^+ = 1 + \beta$ where $\beta = 10$, and all other replicates are assigned uniform negative weights, i.e., $w_{ir} = (1 - w^+)/(R_i - 1)$. All the messages are initialized uniformly in both positive and negative settings.

Figure 5.5 shows several curves comparing the upper bound quality as σ is varied in $[0, 2]$. In each plot, TRBP-∞ represents TRBP when the weights are fully optimized using conditional gradient (this corresponds to the best possible bound with tree-width 1). The first panel shows the bound found by naïve mini-bucket (MBE) for various *ibound*s. We then show our proposed WMB bound when either $\bar{\boldsymbol{\theta}}$ or $\bar{\boldsymbol{w}}$ is modified during a *single* forward pass (so that the estimate has the same computational complexity as mini-bucket), and when the bound is fully optimized. Figure 5.6 shows similar curves for the lower bound qualities of WMB and naïve mean field. Figure 5.7 (a) and (b) respectively show a calibrated timing comparison on a single example (with $\sigma = 0.6$) for positive and negative weights, in which one unit of time equals a full iteration of TRBP or naïve mean field, or a single forward-backward pass of WMB. We use a gradient step on the edge appearance probability at each iteration of TRBP (with step size .1) to find the optimal bound.

With positive weights, the naïve mini-bucket bound is generally worse than `TRBP-`∞, unless used with a relatively large *ibound* (e.g., 8). In contrast, WBM with $\bar{\boldsymbol{\theta}}$ or $\bar{\boldsymbol{w}}$ updated using only one iteration performs almost or equally as well as `TRBP-`∞ using only $ibound = 2$. Perhaps surprisingly, within this problem setting we found that optimizing $\bar{\boldsymbol{w}}$ consistently gave tighter bounds than optimizing $\bar{\boldsymbol{\theta}}$. Although this is not always true for other models or lower bound settings (as we will show later), it does highlight the potential benefit of updating the weights. However, most implementations of TRBP [e.g., Mooij, 2010, Schmidt, 2007] do not include weight optimization even for pairwise models, and it becomes even more difficult for larger clique sizes.

The story for negative weights is quite different here. First, the basic mini-bucket method outperforms naïve mean field for small interaction strength σ_p, but gets significantly worse as σ_p increases. For our weighted mini-bucket bound, we find that updating $\bar{\boldsymbol{\theta}}$ or $\bar{\boldsymbol{w}}$ for only one iteration (as we did with positive weights) does not give impressive performance. However, the bounds are significantly improved when $\bar{\boldsymbol{\theta}}$ is updated for five iterations. Interestingly,



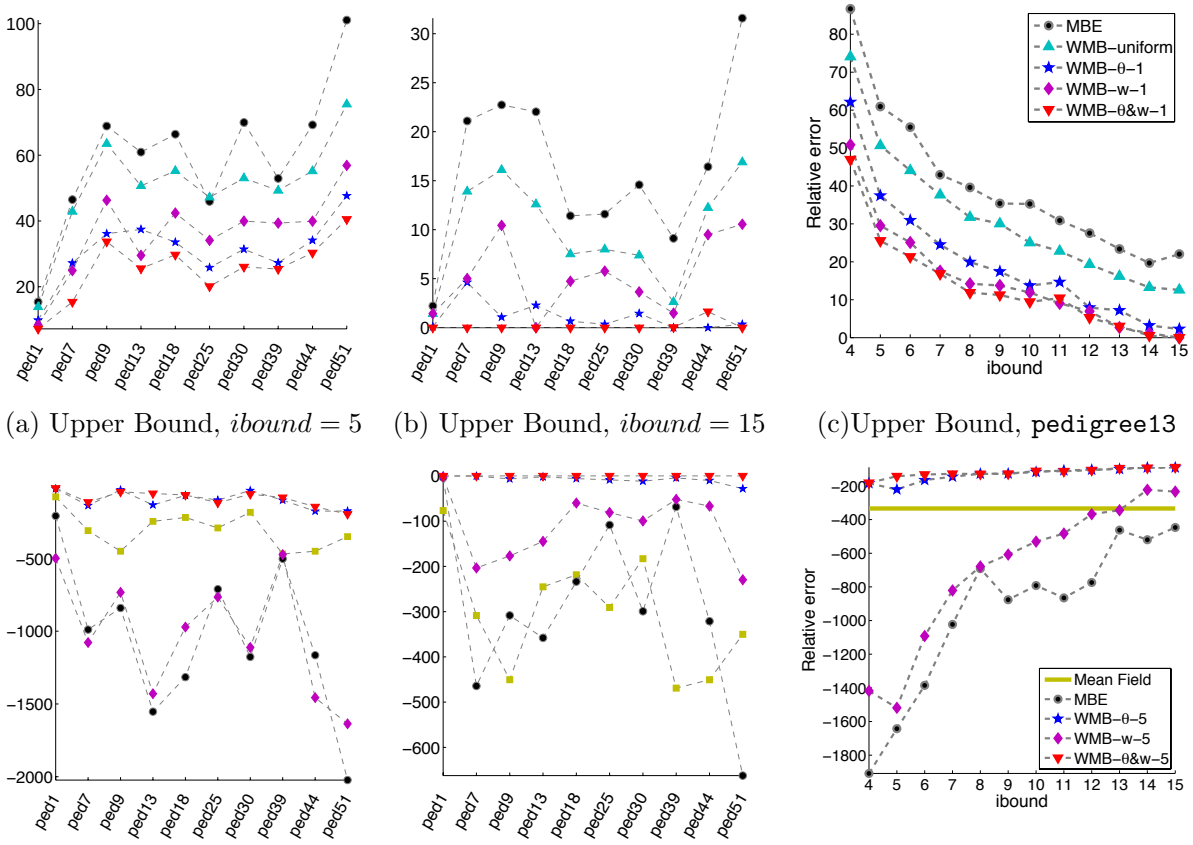
(a) Upper bound (b) Lower bound

Figure 5.7: Calibrated timing comparison on a single grid. Colors indicate increasing *ibound*, and symbols indicate update strategies (θ, w, or both); for example, see also Figure 5.5. One unit of time corresponds to a full iteration of TRBP or mean field (plus a weight update step for TRBP), or a single forward-backward pass of WMB with *ibound* = 1; trajectories indicate the current tightest bounds. In this instance, WMB with w-steps out-performs θ-steps for the upper bound (a), while exactly the opposite is true for the lower bound (b). Optimizing both $\theta\&w$ is usually best. The largest gain comes from increasing the *ibound*, e.g., for the upper bound, `WMB2-`$\theta\&w$ is better than `TRBP-`∞ in its first iteration.

updating $\bar{\boldsymbol{w}}$ in this case seems to not contribute greatly to the quality of the lower bound, the opposite situation to the positive weights setting. This could be because we do not search for the optimal set of positive replicates. When both $\bar{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{w}}$ are updated to convergence (or after a maximum of 50 iterations), our WMB bound significantly outperforms naïve mean field, even with *ibound* = 1.

Additionally, the most benefit occurred within the first few iterations of WMB, suggesting that running message-passing to convergence may be wasteful. It appears better to extract a primal WMB bound early, then increase *ibound* if possible. This shows the advantage of the primal bound's *any-time* property: the algorithm can stop prior to convergence and return a valid bound.

Linkage analysis. We also compared our algorithm to standard mini-bucket on models for pedigree linkage analysis from the UAI'08 approximate inference challenge. The models



(a) Upper Bound, $ibound = 5$ (b) Upper Bound, $ibound = 15$ (c)Upper Bound, `pedigree13`

(d) Lower Bound, $ibound = 5$ (e) Lower Bound, $ibound = 15$ (f) Lower Bound, `pedigree13`

Figure 5.8: Weighted mini-bucket bounds on linkage analysis networks. (a)-(b) We show the upper bound error relative to the tightest upper bound found on 10 problems at two different *ibound*s, for mini-bucket and four single-pass WMB estimates: uniform (no updates), w or θ only, and both. All four consistently outperform standard mini-bucket. (c) A typical instance, showing upper bound quality as *ibound* is increased. (d)-(f) show the corresponding plots for computing lower bounds, except that the WMB variants use 5 iterations rather than a single pass.

have ∼300-1000 nodes, with induced width of ∼20-30. For computing upper bounds, we compare MBE to WMB with only one forward elimination, giving both methods equal time complexity and varying $ibound \in [4 \dots 15]$. For computing lower bounds, we again find that running WMB for a few more iterations gives significant improvements compared to the first iteration, so we show WMB after 5 iterations. The implementation of MBE follows Rollon and Dechter [2010], which used advanced heuristics for bucket partitioning; our WMB continued to use naïve scope-based heuristics (see section 5.3.3).

Figure 5.8 shows that all versions of WMB significantly outperform standard mini-bucket. Even `WMB-uniform`, which performs no updates to $\bar{\boldsymbol{\theta}}$ or $\bar{\boldsymbol{w}}$, outperforms standard MBE (for `WMB-uniform` we use uniform initial weights $\bar{\boldsymbol{w}}$ and a uniform allocation of $\bar{\boldsymbol{\theta}}$ obtained by splitting each factor in the original $\boldsymbol{\theta}$ uniformly). Unlike in the Ising model experiments, here we find that `WMB-`w is not consistently better than `WMB-`θ.

5.6 Conclusions and Future Directions

We presented a weighted mini-bucket algorithm that unifies and extends elimination-based methods such as mini-bucket and variational approaches including tree-reweighted BP, negative tree-reweighed BP and conditional entropy decomposition. Our algorithm inherits significant benefits from both views: it is able to produce a bound at any point (often a few or even one iteration is sufficient); it compactly represents and can optimize over a large class of TRBP bounds; it can easily improve its bounds with either iterations or by increasing clique size; and it provides a unified method for calculating both upper and lower bounds. We also showed extensive experiments that demonstrate the flexibility of WMB for trading off increased clique sizes and iterative updates to obtain tight bounds in both the upper and lower bound settings. Interestingly, while most existing variational methods focus on the notion of reparameterization, corresponding to the update of $\bar{\boldsymbol{\theta}}$ and enforcement of moment matching in our method, we show that optimization of the weights, and enforcement of the entropy matching condition, may dramatically improve the accuracy of the method, particularly when computing upper bounds.

Future Directions. The WMB framework and algorithms open many potential directions. For example, our resulting bounds depend strongly on both the elimination ordering and the bucket partitioning policy, making it an important problem to develop more advanced heuristics for selecting both. In the case of computing lower bounds using negative weights,

it is likely important to develop efficient algorithms to select the best subdomains $\overline{\mathcal{W}}^{-}_{[\kappa_1 \cdots \kappa_n]}$ (i.e., the set of positive replicates); a similar issue arises in negative TRBP, for which Liu and Ihler [2010] proposed a selection heuristic that may be useful here. Additionally, the choices of weights that belong neither to $\overline{\mathcal{W}}^{+}$ nor $\overline{\mathcal{W}}^{-}$ do not provide upper or lower bounds, but may still be able to provide accurate approximations; an interesting problem is to consider how to select among these weights to yield accurate estimates. Another interesting problem is whether one can construct a better approximation given a (properly constructed) pair of upper and lower bounds (e.g., one simple strategy could be to average the upper and lower bounds in order to get a better approximation).

Algorithmic improvements are also possible. Our fixed point update on $\bar{\boldsymbol{\theta}}$ is fast and does not require any step size tuning, unlike gradient based updates. However, the fixed point update is not guaranteed to converge. It would be interesting to construct a convergent fixed-point variant. In addition, we only provided a gradient based update on the weight $\bar{\boldsymbol{w}}$; deriving similar fixed point updates on $\bar{\boldsymbol{w}}$ is less straightforward, but comprises another interesting open direction.

Finally, our techniques can be conveniently applied to the problem of learning graphical models from data, and in particular for trading off bias and variance effects; see, e.g., Gelfand et al. [2013] for a recent discussion.

Chapter 6

Variational Message Passing For Marginal MAP

It is relatively straightforward to extend most variable elimination-based methods, such as bucket or weighted mini-bucket elimination, to solve marginal MAP problems. However, extending variational approaches to this setting is less obvious and has not been previously studied; these extensions are enabled by the novel mixed-inference variational forms we developed in Chapter 4. In this chapter, we leverage the variational form of marginal MAP given in Corollary 4.1, and develop a spectrum of efficient message passing algorithms that both admit appealing theoretical properties and perform well in practice. Compared to variable elimination-based methods, our variational viewpoint allows us to define flexible approximation schemes to derive and analyze novel analogues of loopy BP and mean field like methods for marginal MAP, and to leverage the set of advanced approximation and optimization techniques that have been developed for variational inference over the years [e.g., Wainwright and Jordan, 2008].

In particular, we extend the Bethe and tree reweighted (TRW) entropy approximations and derive a novel "mixed-product" belief propagation (BP) that is a hybrid of max-product, sum-product, and special "argmax-product" message updates. We also derive a class of convergent algorithms based on the proximal point method, which have the form of iteratively

solving pure (or annealed) marginalization tasks. In addition, we discuss theoretical conditions under which our mixed-product BP algorithms obtain both global and local optimality guarantees. More general algorithms based on junction graph and factor graph representations are also derived to exploit higher order cliques. We further discuss the application of mean field type approximations to our variational form, highlighting their connection to expectation-maximization (EM). Our numerical experiments show that our methods can provide significantly better solutions than existing algorithms, including a similar hybrid message passing algorithm by Jiang et al. [2011] and a state-of-the-art algorithm based on local search methods.

This chapter is organized as follows. Section 6.1 reviews the definition and variational representation of marginal MAP, and studies its properties on what we term A-B trees, on which the pairwise entropy approximation is exact. We propose analogues of the Bethe and tree-reweighted approximations for marginal MAP in Section 6.2. A class of "mixed-product" message passing algorithms is proposed and analyzed in Section 6.3, and in Section 6.4 convergent alternatives are proposed based on proximal point methods. We then discuss the EM algorithm for marginal MAP and show that it corresponds to a mean-field-like approximation within our variational representation. We provide extensions of our algorithms to factor graphs and to junction graphs in Section 6.6 and Section 6.7, respectively. We discuss some related work in Section 6.8 and present numerical results in Section 6.9. Finally, the conclusion and future directions are discussed in Section 6.10. Some proofs and additional information can be found in Appendix B.

6.1 Background

We start by reviewing some background and stating several important definitions and results. For notational simplicity, in most of the chapter we will focus our discussion on pairwise

Markov random field models, that is,

$$p(\boldsymbol{x}) = \exp(\theta(\boldsymbol{x}) - \Phi(\theta)), \qquad \text{where} \qquad \theta(\boldsymbol{x}) = \sum_i \theta_i(x_i) + \sum_{(ij)\in E} \theta_{ij}(x_i, x_j), \tag{6.1}$$

where $G := (V, E)$ is the associated Markov graph. More general models with higher order factors will be discussed in Section 6.6 and Section 6.7[1]. In parallel, we will also make use of the multiplicative factorization form,

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{i\in V} \psi_i(x_i) \prod_{(ij)\in E} \psi_{ij}(x_i, x_j), \tag{6.2}$$

where

$$\psi_i(x_i) = \exp(\theta_i(x_i)), \qquad \psi_{ij}(x_i, x_j) = \exp(\theta_{ij}(x_i, x_j)), \qquad Z = \exp(\Phi(\theta))$$

are the singleton and pairwise factors, and partition function, respectively.

Let A be a subset of the nodes V, and $B = V \setminus A$ be the complement of A. The marginal MAP problem seeks a partial configuration $\boldsymbol{x}_B^*$ that maximizes its marginal probability $p(\boldsymbol{x}_B) = \sum_{\boldsymbol{x}_A} p(\boldsymbol{x})$. Formulated in terms of the exponential family form, we have

$$\Phi_{AB}(\boldsymbol{\theta}) = \max_{\boldsymbol{x}_B} Q(\boldsymbol{x}_B \,;\, \boldsymbol{\theta}), \qquad \text{where} \qquad Q(x_B \,;\, \boldsymbol{\theta}) = \log \sum_{\boldsymbol{x}_A} \exp[\theta(\boldsymbol{x})],$$

where the maximizing configuration $\boldsymbol{x}_B^*$ of $Q(\boldsymbol{x}_B \,;\, \boldsymbol{\theta})$ is called the marginal MAP solution. Corollary 4.1 provides an equivalent variational representation for Φ_{AB},

$$\Phi_{AB}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau}\in\mathbb{M}} \big\{ \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + H(\boldsymbol{x}_A | \boldsymbol{x}_B; \boldsymbol{\tau}) \big\}, \tag{6.3}$$

where $\mathbb{M}$ is the marginal polytope – the set of marginals $\boldsymbol{\tau} = \{\boldsymbol{\tau}_i, \boldsymbol{\tau}_{ij} \colon i \in V, (ij) \in E\}$ which

[1] In addition, any higher-order model can be transformed into a pairwise model [e.g., Wainwright and Jordan, 2008].

correspond to some valid joint distribution on $\boldsymbol{x}$. The methods we develop in this chapter are all based on approximating this variational form.

The variational form (6.3) is proven in Corollary 4.1 using the more general variational form of sequential powered sums in Theorem 4.1. Here, we also give a direct, independent proof.

Direct Proof of (6.3). For any $\boldsymbol{\tau} \in \mathbb{M}$ and its corresponding joint distribution $\tau(\boldsymbol{x})$, the conditional KL divergence between $\tau(\boldsymbol{x}_A|\boldsymbol{x}_B)$ and $p(x_A|\boldsymbol{x}_B;\boldsymbol{\theta}) := \exp(\theta(\boldsymbol{x}))/\sum_{\boldsymbol{x}_A}\exp(\theta(\boldsymbol{x}))$ is written as

$$\begin{aligned}
\mathrm{KL}(\boldsymbol{\tau}(\boldsymbol{x}_A|\boldsymbol{x}_B) \;||\; p(\boldsymbol{x}_A|\boldsymbol{x}_B)) &= \sum_{\boldsymbol{x}} \tau(\boldsymbol{x}) \log \frac{\tau(\boldsymbol{x}_A|\boldsymbol{x}_B)}{p(\boldsymbol{x}_A|\boldsymbol{x}_B \;;\; \boldsymbol{\theta})} \\
&= -H(\boldsymbol{x}_A|\boldsymbol{x}_B \;;\; \boldsymbol{\tau}) - \mathbb{E}_\tau[\log p(\boldsymbol{x}_A|\boldsymbol{x}_B \;;\; \boldsymbol{\theta})] \\
&= -H(\boldsymbol{x}_A|\boldsymbol{x}_B \;;\; \boldsymbol{\tau}) - \mathbb{E}_\tau[\theta(\boldsymbol{x})] + \mathbb{E}_\tau[\log \sum_{\boldsymbol{x}_A} \exp(\theta(\boldsymbol{x}))] \quad \geq \quad 0,
\end{aligned}$$

where the last inequality follows from the non-negativity of KL divergence, and is tight if and only if $\tau(\boldsymbol{x}_A|\boldsymbol{x}_B) = p(\boldsymbol{x}_A|\boldsymbol{x}_B \;;\; \boldsymbol{\theta})$. Therefore, we have for any $\tau(\boldsymbol{x})$,

$$\log \max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} \exp(\theta(\boldsymbol{x})) \geq \mathbb{E}_{\boldsymbol{\tau}}\Big[\sum_{\boldsymbol{x}_A} \exp(\theta(\boldsymbol{x}))\Big] \geq \mathbb{E}_\tau[\theta(\boldsymbol{x})] + H(\boldsymbol{x}_A|\boldsymbol{x}_B \;;\; \boldsymbol{\tau}).$$

It is easy to show that the two inequality signs are tight if and only if $\tau(\boldsymbol{x})$ equals $\tau^*(\boldsymbol{x}) = p(\boldsymbol{x}_A|\boldsymbol{x}_B;\boldsymbol{\theta}) \cdot \mathbf{1}[\boldsymbol{x}_B \in \arg\max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} \exp(\boldsymbol{x})]$. Substituting $\mathbb{E}_\tau[\theta(\boldsymbol{x})] = \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle$ completes the proof of (6.3). □

6.1.1 A-B Tree

Although either max-inference or sum-inference can be solved with computational complexity that grows linearly in the number of variables for tree-structured graphs, the marginal MAP can in general be NP-hard on trees. The difficulty arises because the max and sum operators

(a). A = {1,2} B = {3,4} (b). A = {1,2}, B = {3,4}

Figure 6.1: Examples of AB-trees. (a) is an A-B tree, but (b) is not. Note that they have the same graph structure, but different A, B nodes.

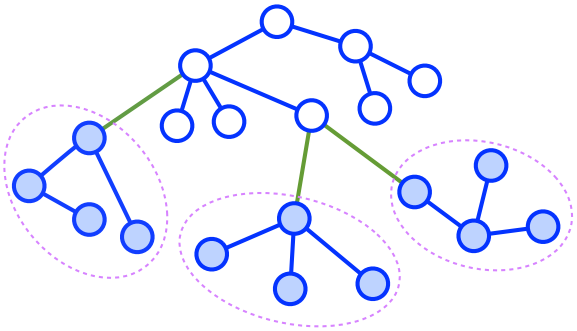


Figure 6.2: Illustrating the structure of A-B trees (in Proposition 6.1). Each connected component of G_A (shaded nodes) can connect with at most one edge in ∂_{AB} (green lines).

do not commute, which restricts the feasible elimination orders to those in which all the sum nodes are eliminated before any max nodes (see Figure 3.2 for an illustration). However, it is useful to characterize a more restrictive set of structural constraints that will ensure that marginal MAP remains tractable; to this end we define the concept of A-B trees.

Definition 6.1. *We call G an A-B* tree *if its A-B induced width ω_{AB} equals one, that is, there exists a tree order on G (in which each node has at most one parent), such that all the nodes in A rank earlier than the nodes in B. We call this tree order an A-B tree-order of G. See Figure 6.1 for illustrative examples.*

For further notation, let $G_A = (A, E_A)$ be the subgraph induced by nodes in A, i.e., $E_A = \{(ij) \in E \colon i \in A, j \in A\}$, and similarly for $G_B = (B, E_B)$. Let $\partial_{AB} = \{(ij) \in E \colon i \in A, j \in B\}$ be the *crossing edges* that join the sets A and B.

Structure of A-B trees. Obviously, if G is an A-B tree, then both G_A and G_B should be trees; in addition, G_A and G_B should be connected by a set of crossing edges ∂_{AB} satisfying certain constraints, as characterized in the following:

Proposition 6.1. *If G is an A-B tree, then both G_A and G_B are trees. In addition,*

(1). Any two edges of in ∂_{AB} can not be connected by edges or nodes of T in the sum part G_A. That is, each connected component of the sum part G_A can connect to at most one edge in ∂_{AB}.

(2). Conversely, any trees G_A, G_B and ∂_{AB} satisfying (1) above form an A-B tree.

Proof. These results are straightforward, ensuring existence of a summation-first tree order. For a pictorial illustration, see Figure 6.2. □

This suggests a constructive process for generating A-B trees – taking any two trees G_A and G_B, and then connecting each connected component of G_A to at most one node in G_B.

Marginal MAP on A-B trees. Obviously, marginal MAP on an A-B tree can be tractably solved with linear complexity by exact bucket elimination along the A-B tree-order. We show that its variational form (6.3) is also tractable in this case.

Lemma 6.1. *If G is an A-B tree, then*

(1). The locally consistent polytope equals the marginal polytope, that is, $\mathbb{M} = \mathbb{L}$.

(2). The conditional entropy has a pairwise decomposition,

$$H(\boldsymbol{x}_A|\boldsymbol{x}_B;\boldsymbol{\tau}) = \sum_{i\in A} H_i(\tau_i) \quad - \sum_{(ij)\in E_A\cup\partial_{AB}} I_{ij}(\tau_{ij}). \tag{6.4}$$

Proof. (1). If G is A-B tree, it is also a tree; see Proposition 3.4. (2). Because G is an A-B tree, both $p(\boldsymbol{x})$ and $p(\boldsymbol{x}_B)$ have tree-structured conditional dependency. We then have [see e.g., Wainwright and Jordan, 2008] that

$$H(\boldsymbol{x};\boldsymbol{\tau}) = \sum_{i\in V} H_i(\tau_i) - \sum_{(ij)\in E} I_{ij}(\tau_{ij}), \quad \text{and} \quad H(\boldsymbol{x}_B;\boldsymbol{\tau}) = \sum_{i\in B} H_i(\tau_i) - \sum_{(ij)\in E_B} I_{ij}(\tau_{ij}).$$

Equation (6.4) follows from the entropic chain rule, $H(\boldsymbol{x}_A|\boldsymbol{x}_B;\boldsymbol{\tau}) = H(\boldsymbol{x};\boldsymbol{\tau}) - H(\boldsymbol{x}_B;\boldsymbol{\tau})$. □

Remark. If G is a tree, but not an A-B tree, then the marginal distribution $p(\boldsymbol{x}_B)$ is in general not tree structured, and the conditional entropy $H(\boldsymbol{x}_A|\boldsymbol{x}_B;\boldsymbol{\tau}) = H(\boldsymbol{x};\boldsymbol{\tau}) - H(\boldsymbol{x}_B;\boldsymbol{\tau})$ is generally intractable due to the difficulty of calculating $H(\boldsymbol{x}_B;\boldsymbol{\tau})$.

6.2 Entropy Approximations

Lemma 6.1 suggests that the conditional entropy $H(\boldsymbol{x}_A|\boldsymbol{x}_B;\boldsymbol{\tau})$ in (6.3) is tractable on A-B trees, and can be decomposed into singleton and pairwise terms that are easy to deal with. This is not true for general graphs, but motivates a set of "Bethe" and "tree reweighted" approximations, analagous to the approximations for sum-inference.

6.2.1 Bethe-like Entropy Approximation

We define the following "Bethe-like" entropy approximation to approximate the variational form (6.3) for marginal MAP,

$$\Phi_{bethe}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau}\in\mathbb{L}} \Big\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle \; + \; \sum_{i\in A} H_i(\tau_i) \; - \sum_{(ij)\in E_A \cup \partial_{AB}} I_{ij}(\tau_{ij}) \Big\}, \tag{6.5}$$

where the entropy is a "truncated" version of the usual Bethe entropy approximation, in which the entropy and mutual information terms that involve only max nodes are truncated

(removed). If G is an A-B tree, Φ_{bethe} equals the true Φ_{AB}, giving an intuitive justification to the approximation. In the sequel we give more general theoretical conditions under which this approximation gives the exact solution; we also find empirically that it often gives surprisingly good solutions in practice. Similarly to the standard Bethe approximation, (6.5) leads to a nonconvex optimization problem, and we will derive both message passing algorithms and provably convergent algorithms to solve it.

6.2.2 Tree-reweighted Entropy Approximation

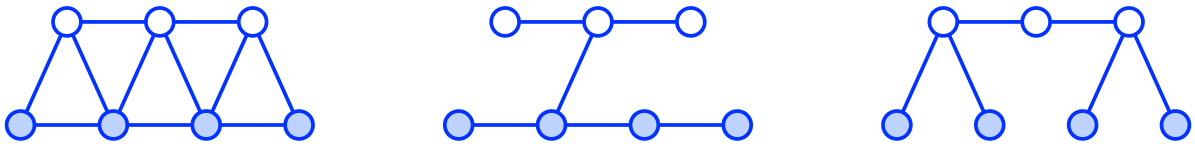
Building on the idea of tree reweighted (TRW) BP [Wainwright et al., 2005] (see Section 3.2.4 for an introduction), we also construct an approximation to the marginal MAP problem using a convex combination of A-B spanning trees (spanning trees of G that are A-B trees). Let $\mathcal{T}_{AB}$ be a collection of A-B spanning trees of G. We assign to each $T \in \mathcal{T}_{AB}$ a weight w^T satisfying $w^T \geq 0$ and $\sum_{T \in \mathcal{T}_{AB}} w^T = 1$. For each A-B spanning tree $T = (V, E^T)$, the entropy on tree T is given by

$$H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau}\ ;\ T) = \sum_{i \in A} H_i(\tau_i)\ -\ \sum_{(ij) \in E^T \setminus E_B} I_{ij}(\tau_{ij}).$$

As shown in Wainwright and Jordan [2008], the entropy $H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau}\ ;\ T)$ is always a concave function of $\boldsymbol{\tau}$ on $\mathbb{L}$, and $H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau}) \leq H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau}\ ;\ T)$ for all $\boldsymbol{\tau} \in \mathbb{M}$ and $T \in \mathcal{T}_{AB}$. More generally, we have

$$H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau}) \leq \sum_{T \in \mathcal{T}_{AB}} w^T H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau}\ ;\ T)\ =\ \sum_{i \in A} H_i(\tau_i)\ \ -\ \sum_{(ij) \in E_A \cup \partial_{AB}} \rho_{ij} I_{ij}(\tau_{ij}),$$

where $\rho_{ij} = \sum_{T:(ij) \in E^T} w^T$ are the edge appearance probabilities as defined in Wainwright and Jordan [2008]. Replacing $\mathbb{M}$ with $\mathbb{L}$ and $H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau})$ with this bound yields a TRW-like



(a) Not an A-B tree

(b) G_A-maximal A-B spanning tree

(c) ∂_{AB}-maximal A-B spanning tree

Figure 6.3: (a) is not an A-B tree, and (b) and (c) are, respectively, *G_A-maximal* and *∂_{AB}-maximal* A-B spanning trees of (a).

approximation of marginal MAP,

$$\Phi_{trw}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau}\in\mathbb{L}} F_{trw}(\boldsymbol{\tau},\boldsymbol{\theta}), \quad F_{trw}(\boldsymbol{\tau},\boldsymbol{\theta}) = \langle \boldsymbol{\theta},\boldsymbol{\tau}\rangle \; + \sum_{i\in A} H_i(\tau_i) \; - \sum_{(ij)\in E_A\cup\partial_{AB}} \rho_{ij} I_{ij}(\tau_{ij}). \quad (6.6)$$

Since $\mathbb{L}$ is an outer bound of $\mathbb{M}$, and F_{trw} is a concave upper bound of the true variational objective, we can guarantee that $\Phi_{trw}(\boldsymbol{\theta})$ is always an upper bound of $\Phi_{AB}(\boldsymbol{\theta})$. To our knowledge, this provides the first known convex relaxation for upper bounding marginal MAP. One can also optimize the weights $\{w^T : T \in \mathcal{T}_{AB}\}$ to obtain the tightest upper bound using methods similar to those used for standard TRW-BP [see Wainwright et al., 2005].

A-B spanning trees. The structural constraints of A-B trees, as illustrated in Proposition 6.1, makes the edge appearance probabilities of A-B spanning trees more restricted than those of the spanning trees used in sum-inference TRW-BP. As illustrated in Proposition 6.1, one can construct an A-B spanning tree by first selecting spanning trees in G_A and in G_B, respectively, and then joining each connected component of G_A with any node in G_B. Due to the constraint that at most one crossing edge is incident to each connected component of G_A, there is a trade-off between incorporating more edges in G_A versus in ∂_{AB}; two simple, extreme cases, as illustrated in Figure 6.3, stand out:

1. *G_A-maximal A-B spanning trees*, which include as many edges as possible in the spanning tree of G_A; this makes the number of edges that can be included in ∂_{AB} very small. Assuming G_A is a connected graph, then ∂_{AB} can include only one edge; if we

use a set of G_A-maximal spanning trees for TRW-BP in this case, the edge appearance probability $\{\rho_{ij}\}$ should satisfy

$$\sum_{(ij)\in\partial_{AB}} \rho_{ij} = 1, \quad \rho_{ij} \geq 0,$$

that is, the sum of the weights on the crossing edges ∂_{AB} equals one.

2. *∂_{AB}-maximal A-B spanning trees*, which include no edges in G_A, but a maximum number of edges in ∂_{AB} – edges that connect each node in G_A with a unique node in G_B. Therefore, if we use a set of ∂_{AB}-maximal spanning trees for TRW-BP, we should have $\rho_{ij} = 0$ for $\forall(ij) \in E_A$.

Intuitively, G_A-maximal spanning trees capture more information about the summation structures of G_A, while ∂_{AB}-maximal spanning trees capture more information about ∂_{AB}, relating the sum and max parts. Using only G_A-maximal spanning trees, for example, will ensure that if G_A is a tree, the summation component is exact (all $\rho_{ij} = 1$ for $(ij) \in E_A$); this will make it possible to provide some theoretical guarantees about the solution, such as discussed in the next section.

6.2.3 Global Optimality Guarantees

In this section, we demonstrate that the preceding approximations exhibit global optimality guarantees under some circumstances. For the purposes of this section, we assume that G_A is a tree, and hence the objective function is tractable to calculate for a given configuration $\boldsymbol{x}_B$. However, note that the optimization component typically remains intractable even in this case, because the marginal distribution over the max variables $p(\boldsymbol{x}_B)$ may have high induced width even when G_A is a tree (see, e.g., Figure 3.2). It is thus not obvious whether or when the Bethe and TRW approximations can provide global optimality guarantees.

In general, suppose we approximate $\Phi_{AB}(\boldsymbol{\theta})$ using the following pairwise approximation,

$$\Phi_{tree}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau}\in\mathbb{L}} \big\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle \; + \; \sum_{i\in A} H_i(\tau_i) - \sum_{(ij)\in E_A} I_{ij}(\tau_{ij}) - \sum_{(ij)\in \partial_{AB}} \rho_{ij} I_{ij}(\tau_{ij}) \big\}, \tag{6.7}$$

where the weights on the sum part, $\{\rho_{ij} \colon (ij) \in E_A\}$, have all been fixed to equal one. This choice ensures that the entropy of the sum part is "intact" in the approximation, while the weights of the edges that cross between A and B, $\boldsymbol{\rho}_{AB} = \{\rho_{ij} \colon (ij) \in \partial_{AB}\}$, can take arbitrary values, corresponding to different entropy approximation methods. If $\rho_{ij} = 1$ for $\forall (ij) \in \partial_{AB}$, the objective (6.7) equals the Bethe approximation Φ_{bethe}; alternatively, it corresponds to a TRW approximation Φ_{trw} if the $\{\rho_{ij}\}$ are taken to be edge appearance probabilities of a set of G_A-maximum spanning trees, which in general will have positive values less than one. In particular, if G_A is a connected graph, we should have (as shown in the last section) $\sum_{(ij)\in\partial_{AB}} \rho_{ij} = 1$, $\rho_{ij} \geq 0$. Interestingly, we show in Section 6.5 that if $\rho_{ij} \to +\infty$ for $\forall (ij) \in \partial_{AB}$, then Equation (6.7) is closely related to an EM algorithm.

Theorem 6.1. *Suppose the sum part G_A is a tree, and we approximate $\Phi_{AB}(\boldsymbol{\theta})$ using $\Phi_{tree}(\boldsymbol{\theta})$ defined in* (6.7)*. Assume that* (6.7) *is* globally *optimized; then:*

1. *We have $\Phi_{tree}(\boldsymbol{\theta}) \geq \Phi_{AB}(\boldsymbol{\theta})$. If there exists $\boldsymbol{x}_B^*$ such that $Q(\boldsymbol{x}_B^*; \boldsymbol{\theta}) = \Phi_{tree}(\boldsymbol{\theta})$, we have $\Phi_{tree}(\boldsymbol{\theta}) = \Phi_{AB}(\boldsymbol{\theta})$, and $\boldsymbol{x}_B^*$ is a globally optimal marginal MAP solution.*
2. *Suppose $\boldsymbol{\tau}^*$ is a* global *maximum of* (6.7)*, and $\{\tau_i^*(x_i) \colon i \in B\}$ have integer values, i.e., $\tau_i^*(x_i) = 0$ or 1. Then, $\{x_i^* = \arg\max_{x_i} \tau_i^*(x_i) \colon i \in B\}$ is a globally optimal solution of the marginal MAP problem* (6.1)*.*

Proof (sketch). (See Appendix B for the complete proof.) The fact that the sum part G_A is a tree guarantees the marginalization is exact. Showing that (6.7) is a relaxation of the maximization problem and applying standard relaxation arguments completes the proof. □

Remark. Theorem 6.1 does not directly require any condition on the values of $\boldsymbol{\rho}_{AB}$; however, different values of ρ_{AB} can affect whether the conditions of Theorem 6.1 will hold in

practice, and suggest a fundamental tradeoff that reveals the hardness of the problem:

1. On the one hand, the value of $\boldsymbol{\rho}_{AB}$ controls the concavity of the objective function in (6.7) and hence the difficulty of finding a global optimum; small enough $\boldsymbol{\rho}_{AB}$ (such as the valid tree-appearance probabilities in TRW) can ensure that (6.7) is a convex optimization, while larger $\boldsymbol{\rho}_{AB}$ (as in the Bethe approximation or EM settings) causes (6.7) to become non-convex, making it difficult to globally optimize the objective to apply Theorem 6.1.
2. On the other hand, the value of $\boldsymbol{\rho}_{AB}$ also controls how likely the solution is to be integral – larger ρ_{ij} emphasize the mutual information terms, which force the solution towards integral points. Thus the solution of the TRW entropy approximation is less likely to be integral than that of the Bethe entropy approximation, complicating the application of Theorem 6.1 to TRW solutions as well.

Therefore, the TRW approximation (e.g., $\sum_{(ij)\in\partial_{AB}} \rho_{ij} = 1$) and EM ($\rho_{ij} \to +\infty$; see Section 6.5) reflect two extrema of this tradeoff between concavity and integrality, respectively, while the Bethe approximation ($\rho_{ij} = 1$) appears to represent a reasonable compromise that often gives excellent performance in practice. In Section 6.3.2, we give a different set of local optimality guarantees that are derived from a reparameterization perspective.

6.3 Message Passing Algorithms for Marginal MAP

We now derive message-passing-style algorithms to optimize the "truncated" Bethe or TRW approximate objectives in (6.5) and (6.6). Instead of optimizing the exact objectives directly, we consider their "annealed" versions,

$$\max_{\boldsymbol{\tau}\in\mathbb{L}} \left\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \hat{H}(\boldsymbol{x}_A | \boldsymbol{x}_B; \boldsymbol{\tau}) + \epsilon \hat{H}(\boldsymbol{x}_B; \boldsymbol{\tau}) \right\},$$

where ϵ is a positive annealing coefficient (or temperature), and the $\hat{H}(\boldsymbol{x}_A|\boldsymbol{x}_B;\boldsymbol{\tau})$ and $\hat{H}(\boldsymbol{x}_B;\boldsymbol{\tau})$ are the generic pairwise approximations of $H(\boldsymbol{x}_A|\boldsymbol{x}_B;\boldsymbol{\tau})$ and $H(\boldsymbol{x}_B;\boldsymbol{\tau})$, respectively. That is,

$$\begin{aligned}\hat{H}(\boldsymbol{x}_A|\boldsymbol{x}_B;\boldsymbol{\tau}) &= \sum_{i\in A} H_i(\tau_i) \quad - \sum_{(ij)\in E_A\cup\partial_{AB}} \rho_{ij} I_{ij}(\tau_{ij}), \\ \hat{H}(\boldsymbol{x}_B;\boldsymbol{\tau}) &= \sum_{i\in B} H_i(\tau_i) \quad - \sum_{(ij)\in E_B} \rho_{ij} I_{ij}(\tau_{ij}),\end{aligned} \tag{6.8}$$

where different values of the pairwise weights $\{\rho_{ij}\}$ can correspond to either the Bethe approximation ($\rho_{ij} = 1$) or the TRW approximation (edge appearance probabilities, with $\rho_{ij} \leq 1$). This yields a generic pairwise variational optimization problem,

$$\max_{\boldsymbol{\tau}\in\mathbb{L}} \big\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{i\in V} w_i H_i(\tau_i) - \sum_{(ij)\in E} w_{ij} I_{ij}(\tau_{ij}) \big\}, \tag{6.9}$$

where the weights $\{w_i, w_{ij}\}$ are determined by the temperature ϵ and $\{\rho_{ij}\}$ via

$$w_i = \begin{cases} 1 & \forall i \in A \\ \epsilon & \forall i \in B, \end{cases} \qquad w_{ij} = \begin{cases} \rho_{ij} & \forall (ij) \in E_A \cup \partial_{AB} \\ \epsilon\rho_{ij} & \forall (ij) \in E_B. \end{cases} \tag{6.10}$$

The general form represented by (6.9) provides a unified treatment for approximating sum-inference, max-inference and mixed, marginal MAP problems simply by selecting different weights. Specifically,

1. If $w_i = 1$ for all $i \in V$, Eq. (6.9) corresponds to the sum-inference problem and the sum-product BP objectives and algorithms.
2. If $w_i \to 0^+$ for all $i \in V$ (and the corresponding $w_{ij} \to 0^+$), Eq. (6.9) corresponds to the max-inference problem and the max-product linear programming objective and algorithms.
3. If $w_i = 1$ for $\forall i \in A$ and $w_i = 0$ for $\forall i \in B$ (and the corresponding $w_{ij} \to 0^+$), Eq. (6.9)

Algorithm 6.1 Annealed BP for marginal MAP for pairwise models

Input: A marginal MAP problem on a pairwise model $p(\boldsymbol{x}) \propto \prod_{i \in V} \psi_i \prod_{(ij) \in E} \psi_{ij}$ on graph $G = (V, E)$; an annealing parameter scheme $\{\lambda^t\}$.
Output: An approximate marginal MAP solution $\boldsymbol{x}_B^*$.
Define the pairwise weights $\{\rho_{ij} \colon (ij) \in E\}$, e.g., $\rho_{ij} = 1$ for Bethe or valid appearance probabilities for TRW. Initialize the messages $\{m_{i \to j} \colon (ij) \in E\}$.
for iteration t **do**
1. Update ϵ by e.g. $\epsilon = 1/t$, and correspondingly the weights $\{w_i, w_{ij}\}$ by (6.10).
2. Perform the message passing update in (6.11) for all edges $(ij) \in E$.

end for
Calculate the singleton beliefs $b_i(x_i)$ and decode the solution $\boldsymbol{x}_B^*$,

$$x_i^* = \arg\max_{x_i} b_i(x_i), \quad \forall i \in B, \qquad \text{where} \quad b_i(x_i) \propto \psi_i(x_i) m_{\sim i}(x_i).$$

corresponds to the marginal MAP problem; in the sequel, we derive "mixed-product" BP algorithms.

Note the different roles of the singleton and pairwise weights: the singleton weights $\{w_i \colon i \in V\}$ define the type of inference problem, while the pairwise weights $\{w_{ij} \colon (ij) \in E\}$ determine the approximation method (e.g., Bethe vs. TRW).

We now derive a message passing algorithm for solving the generic problem (6.9), using a Lagrange multiplier method similar to Yedidia et al. [2005] or Wainwright et al. [2005].

Proposition 6.2. *Assuming w_i and w_{ij} are strictly positive, the stationary points of* (6.9) *satisfy the fixed point condition of the following message passing update,*

$$\textit{Message Update:} \qquad m_{i \to j}(x_j) \leftarrow \Big[\sum_{x_i} (\psi_i(x_i) m_{\sim i}(x_i))^{\frac{1}{w_i}} \Big(\frac{\psi_{ij}(x_i, x_j)}{m_{j \to i}(x_i)} \Big)^{\frac{1}{w_{ij}}} \Big]^{w_{ij}}, \tag{6.11}$$

Marginal Decoding:

$$\tau_i(x_i) \propto \big[\psi_i(x_i) m_{\sim i}(x_i) \big]^{\frac{1}{w_i}}, \qquad \tau_{ij}(x_i, x_j) \propto \tau_i(x_i) \tau_j(x_j) \left[\frac{\psi_{ij}(x_i, x_j)}{m_{i \to j}(x_j) m_{j \to i}(x_i)} \right]^{\frac{1}{w_{ij}}}, \tag{6.12}$$

where $m_{\sim i}(x_i) := \prod_{k \in \partial(i)} m_{k \to i}(x_i)$ is the product of messages sent into node i, and $\partial(i)$ is the

set of neighboring nodes of i.

Proof (sketch). (See Appendix B for the complete proof.) Note that (6.12) is simply the KKT condition of (6.9), with the log of the message $\log m_{i\to j}$ being the Lagrange multipliers. Plugging (6.12) into the local consistency constraint $\sum_{x_i} \tau_{ij}(x_i, x_j) = \tau_j(x_j)$ gives (6.11). □

The message update (6.11) is mostly similar to TRW-BP of Wainwright et al. [2005], except that it incorporates general singleton weights w_i. The marginal MAP problem can be solved by running (6.11) with $\{w_i, w_{ij}\}$ defined by (6.10) and a scheme for choosing the temperature ϵ, either directly setting ϵ to be a small constant, or gradually decreasing (or annealing) ϵ to zero across iterations, e.g., by $\epsilon = 1/t$ where t is the iteration. Algorithm 6.1 describes the details for the annealing method.

6.3.1 Mixed-Product Belief Propagation

Directly taking $\epsilon \to 0^+$ in message update (6.11), we can get an interesting "mixed-product" BP algorithm that is a hybrid of the max-product and sum-product message updates, with a novel "argmax-product" message update that is specific to marginal MAP problems. This algorithm is listed in Algorithm 6.2, and described by the following proposition:

Proposition 6.3. *As ϵ approaches zero from the positive side, that is, $\epsilon \to 0^+$, the message update* (6.11) *reduces to the update in* (6.13)-(6.15) *in Algorithm 6.2.*

Proof. For messages from $i \in A$ to $j \in A \cup B$, we have $w_i = 1$, $w_{ij} = \rho_{ij}$; the result is obvious.

For messages from $i \in B$ to $j \in B$, we have $w_i = \epsilon$, $w_{ij} = \epsilon\rho_{ij}$. The result follows from the

Algorithm 6.2 Mixed-product BP for marginal MAP for pairwise models

Input: A marginal MAP problem on a pairwise model $p(\boldsymbol{x}) \propto \prod_{i\in V}\psi_i \prod_{(ij)\in E}\psi_{ij}$ on graph $G=(V,E)$.

Output: An approximate marginal MAP solution $\boldsymbol{x}_B^*$.

Define the pairwise weights $\{\rho_{ij}\colon (ij)\in E\}$ and initialize messages $\{m_{i\to j}\colon (ij)\in E\}$ as in Algorithm 6.1.

for iteration t **do**

for edge $(ij)\in E$ **do**

Perform different message updates depending on the node type of the source and destination,

$A\to A\cup B$: (sum-product)
$$m_{i\to j}(x_j) \leftarrow \big[\sum_{x_i}(\psi_i(x_i)m_{\sim i}(x_i))(\frac{\psi_{ij}(x_i,x_j)}{m_{j\to i}(x_i)})^{1/\rho_{ij}}\big]^{\rho_{ij}}, \tag{6.13}$$

$B\to B$: (max-product)
$$m_{i\to j}(x_j) \leftarrow \max_{x_i}(\psi_i(x_i)m_{\sim i}(x_i))^{\rho_{ij}}(\frac{\psi_{ij}(x_i,x_j)}{m_{j\to i}(x_i)}), \tag{6.14}$$

$B\to A$: (argmax-product)
$$m_{i\to j}(x_j) \leftarrow \big[\sum_{x_i\in\mathcal{X}_i^*}(\psi_i(x_i)m_{\sim i}(x_i))(\frac{\psi_{ij}(x_i,x_j)}{m_{j\to i}(x_i)})^{1/\rho_{ij}}\big]^{\rho_{ij}}, \tag{6.15}$$

where the set $\mathcal{X}_i^* = \arg\max_{x_i}\psi_i(x_i)m_{\sim i}(x_i)$ and $m_{\sim i}(x_i)=\prod_{k\in\partial(i)} m_{ki}(x_i)$.

end for

end for

Calculate the singleton beliefs $b_i(x_i)$ and decode the solution $\boldsymbol{x}_B^*$,

$$x_i^* = \arg\max_{x_i} b_i(x_i), \quad \forall i\in B, \text{ where } b_i(x_i)\propto \psi_i(x_i)m_{\sim i}(x_i).$$

---

zero temperature limit formula

$$\lim_{\epsilon\to 0^+}[\sum f(x_i)^{1/\epsilon}]^{\epsilon} = \max_{x_i} f(x_i), \qquad \text{where} \qquad f(x_i) = (\psi_i(x_i)m_{\sim i}(x_i))^{\rho_{ij}}(\frac{\psi_{ij}(x_i,x_j)}{m_{j\to i}(x_i)}).$$

Finally, for messages from $i\in B$ to $j\in A$, we have $w_i=\epsilon$, $w_{ij}=\rho_{ij}$. One can show that

$$\lim_{\epsilon\to 0^+}\Big[\frac{\psi_i(x_i)m_{\sim i}(x_i)}{\max_{x_i}\psi_i(x_i)m_{\sim i}(x_i)}\Big]^{1/\epsilon} = \mathbf{1}(x_i\in\mathcal{X}_i^*),$$

where $\mathcal{X}_i^* = \arg\max_{x_i}\psi_i(x_i)m_{\sim i}(x_i)$. Plugging this into (6.11) and dropping the constant term, we obtain the message update in (6.15). □

Algorithm 6.2 has an intuitive interpretation: the sum-product and max-product messages in (6.13) and (6.14) correspond to the marginalization and maximization steps, respectively. The special "argmax-product" messages in (6.15) serve to "coordinate" the sum-product and max-product messages – they restrict the max nodes to the currently decoded local marginal MAP solutions $\mathcal{X}_i^* = \arg\max \psi_i(x_i) m_{\sim i}(x_i)$, and pass the posterior beliefs back to the sum part. Note that the summation notation in (6.15) can be ignored if $\mathcal{X}_i^*$ has only a single optimal state.

Traditional algorithms for marginal MAP, such as local search or EM, usually have a "double-loop" form: the inner loop fully solves the marginalization sub-problem with a fixed max variable configuration $\boldsymbol{x}_B$, and the outer loop takes a maximization or search step to update the max variable configuration. In contrast, our mixed-product BP reformulates the calculation into a set of distributed message passing updates, and allows us to take simultaneous movements on the marginalization (by sum-product) and maximization (by max-product) sub-problems, and coordinate them (by argmax-product), all in a parallel fashion; this avoids fully solving the expensive marginalization inner loops, and makes our algorithm much more computationally efficient. This advantage is inherited from our general variational framework, which by its form, naturally integrates the marginalization and maximization sub-problems into a joint optimization problem.

Interestingly, Algorithm 6.2 also bears similarity to a recent hybrid message passing method of Jiang et al. [2011], which differs from Algorithm 6.2 only in replacing the special argmax-product messages (6.15) with regular max-product messages. We make a detailed comparison of these two algorithms in Section 6.3.3, and show that it is in fact the argmax-product messages (6.15) that lends our algorithm several appealing optimality guarantees.

6.3.2 Reparameterization and Local Optimality Guarantees

An important interpretation of the sum-product and max-product BP is the reparameterization viewpoint [Wainwright et al., 2003a, Weiss et al., 2007]: message passing updates can be viewed as moving probability mass between local pseudo-marginals (or beliefs), in a way that leaves their product a reparameterization of the original distribution, while ensuring some consistency conditions at the fixed points. Such viewpoints are theoretically important, as they are useful for proving optimality guarantees for BP algorithms. In this section, we show that the mixed-product BP in Algorithm 6.2 has a similar reparameterization interpretation, based on which we establish a local optimality guarantee for mixed-product BP.

To start, we define a set of "mixed-beliefs" as

$$b_i(x_i) \propto \psi_i(x_i) m_{\sim i}(x_i), \qquad b_{ij}(x_{ij}) \propto b_i(x_i) b_j(x_j) \left[\frac{\psi_{ij}(x_i, x_j)}{m_{i \to j}(x_j) m_{j \to i}(x_i)} \right]^{1/\rho_{ij}}. \tag{6.16}$$

The marginal MAP solution should be decoded from $x_i^* \in \arg\max_{x_i} b_i(x_i), \forall i \in B$, as is typical in max-product BP. Note that the above mixed-beliefs $\{b_i, b_{ij}\}$ are different from the local marginals $\{\tau_i, \tau_{ij}\}$ defined in (6.12); they can be seen as softened versions of $\{\tau_i, \tau_{ij}\}$.Their relationship is explicitly clarified in the following proposition.

Proposition 6.4. *The* $\{\tau_i, \tau_{ij}\}$ *in* (6.12) *and the* $\{b_i, b_{ij}\}$ *in* (6.16) *are related via,*

$$\begin{cases} b_i \propto \tau_i & \forall i \in A, \\ b_i \propto (\tau_i)^\epsilon & \forall i \in B \end{cases} \qquad \begin{cases} b_{ij} \propto b_i b_j (\frac{\tau_{ij}}{\tau_i \tau_j}) & \forall (ij) \in E_A \cup \partial_{AB} \\ b_{ij} \propto b_i b_j (\frac{\tau_{ij}}{\tau_i \tau_j})^\epsilon & \forall (ij) \in E_B. \end{cases}$$

Proof. The result follows from a simple algebraic transformation between (6.12) and (6.16). □

Therefore, as $\epsilon \to 0^+$, the τ_i $(= b_i^{1/\epsilon})$ for $i \in B$ will typically concentrate their mass on a deterministic configuration, but b_i may continue to have "soft" (non-deterministic) values.

We now show that the mixed-beliefs $\{b_i, b_{ij}\}$ have a reparameterization interpretation.

Theorem 6.2. *At the fixed point of mixed-product BP in Algorithm 6.2 , the mixed-beliefs defined in* (6.16) *satisfy*

Reparameterization:

$$p(\boldsymbol{x}) \propto \prod_{i \in V} b_i(x_i) \prod_{(ij) \in E} \Big[\frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)}\Big]^{\rho_{ij}}. \tag{6.17}$$

Mixed-consistency:

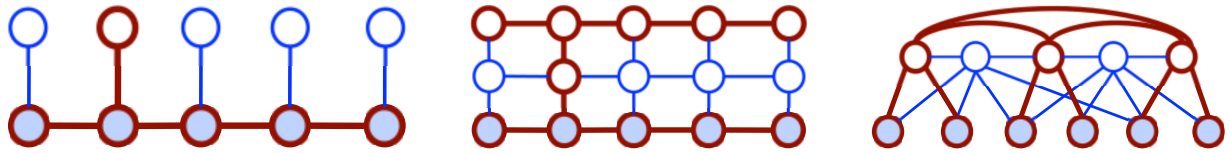
$$\text{(a)} \qquad \sum_{x_i} b_{ij}(x_i, x_j) = b_j(x_j), \qquad \forall i \in A, j \in A \cup B, \tag{6.18}$$

$$\text{(b)} \qquad \max_{x_i} b_{ij}(x_i, x_j) = b_j(x_j), \qquad \forall i \in B, j \in B, \tag{6.19}$$

$$\text{(c)} \qquad \sum_{x_i \in \arg\max b_i} b_{ij}(x_i, x_j) = b_j(x_j), \qquad \forall i \in B, j \in A. \tag{6.20}$$

Proof. Directly substitute the definition (6.16) into the message update (6.13)-(6.15). □

The three mixed-consistency constraints exactly map to the three types of message updates in Algorithm 6.2. Constraints (a) and (b) enforce the regular sum- and max- consistency of the sum- and max- product messages in (6.13) and (6.14), respectively. Constraint (c) corresponds to the argmax-product message update in (6.15): it enforces that the marginals should be consistent after x_i is assigned to the currently decoded solution, $x_i = \arg\max_{x_i} b_i(x_i) = \arg\max_{x_i} \sum_{x_j} b_{ij}(x_i, x_j)$, corresponding to solving a local marginal MAP problem on $b_{ij}(x_i, x_j)$. It turns out that this special constraint is a crucial ingredient of mixed-product BP, enabling us to prove guarantees on the strong local optimality of the solution.



(a) (b) (c)

Figure 6.4: Examples of semi-A-B subtrees. The shaded nodes represent sum nodes, while the unshaded are max nodes. In each graph, a semi-A-B subtree is labeled by red bold lines. Under the conditions of Theorem 6.3, the fixed point of mixed-product BP is locally optimal up to jointly perturbing all the max nodes in any semi-A-B subtree of G.

Local Optimality Guarantees

We are now ready to give a local optimality guarantee for mixed-product BP. First, some notation is required. Suppose C is a subset of max nodes in B. Let $G_{C\cup A} = (C \cup A, E_{C\cup A})$ be the subgraph of G induced by nodes $C \cup A$, where $E_{C\cup A} = \{(ij) \in E \colon i, j \in C \cup A\}$. We call $G_{C\cup A}$ a semi-A-B subtree of G if the edges in $E_{C\cup A} \backslash E_B$ form an A-B tree. In other words, $G_{C\cup A}$ is a semi-A-B subtree if it is an A-B tree when ignoring any edges entirely within the max set B. See Figure 6.4 for examples of semi-A-B subtrees.

Recall that a set of weights $\{\rho_{ij}\}$ is said to be *provably convex* if there exist positive constants $\kappa_{i\to j}$, such that $\sum_{i'\in\partial(i)} \kappa_{i'\to i} \le 1$ and $\kappa_{i\to j} + \kappa_{j\to i} \le \rho_{ij}$; if $\{\rho_{ij}\}$ is probably convex, then $H(\boldsymbol{x};\boldsymbol{\tau}) = \sum_i H_i(\tau_i) - \sum_{ij} \rho_{ij} I_{ij}(\tau_{ij})$ is a guaranteed to be a concave function of $\boldsymbol{\tau}$. See Heskes [2006], Weiss et al. [2007] and Section 3.2.4 for an introduction.

Theorem 6.3. *Suppose C is a subset of B such that $G_{C\cup A}$ is a semi-A-B subtree, and the weights $\{\rho_{ij}\}$ satisfy*

1. *$\rho_{ij} = 1$ for $(ij) \in E_A$;*
2. *$0 \le \rho_{ij} \le 1$ for $(ij) \in E_{C\cup A} \cap \partial_{AB}$;*
3. *$\{\rho_{ij} \colon (ij) \in E_{C\cup A} \cap E_B\}$ is provably convex.*

At the fixed point of mixed-product BP in Algorithm 6.2, if the mixed-beliefs on the max nodes $\{b_i, b_{ij} \colon i, j \in B\}$ defined in (6.16) all have unique maxima, then there exists a B-configuration $\boldsymbol{x}^*_B$ satisfying $x^*_i = \arg\max b_i$ for $\forall i \in B$ and $(x^*_i, x^*_j) = \arg\max b_{ij}$ for

$\forall (ij) \in E_B$, and $\boldsymbol{x}_B^*$ is locally optimal in the sense that $Q(\boldsymbol{x}_B^*; \boldsymbol{\theta})$ is not smaller than any B-configuration that differs from $\boldsymbol{x}_B^*$ only on C, that is, $Q(\boldsymbol{x}_B^*; \boldsymbol{\theta}) = \max_{\boldsymbol{x}_C} Q([\boldsymbol{x}_C, x^*_{B\setminus C}]; \boldsymbol{\theta})$.

Proof (sketch). (See Appendix B for the complete proof.) The mixed-consistency constraint (c) in (6.20) and the fact that $G_{C\cup A}$ is a semi-A-B subtree enables the summation part to be eliminated away. The remaining part only involves the max nodes, and the method in Weiss et al. [2007] for analyzing standard MAP can be applied. □

Remark. The proof of Theorem 6.3 relies on transforming the marginal MAP problem to a standard MAP problem by eliminating the summation part. Therefore, variants of Theorem 6.3 may be derived using other global optimality conditions of convexified belief propagation or linear programming algorithms for MAP, such as those in Werner [2007, 2010] or Wainwright et al. [2003b]. We leave this to future work.

For $G_{C\cup A}$ to be a semi-A-B subtree, the sum part G_A must be a tree, which Theorem 6.3 assumes implicitly. For the hidden Markov chain in Figure 2.3, Theorem 6.3 implies only local optimality up to Hamming distance one (or, coordinate-wise optimality), because any semi-A-B subtree of G in Figure 2.3 can contain at most one max node. However, Theorem 6.3 is in general much stronger, especially when the sum part is not fully connected, or when the max part has interior regions disconnected from the sum part. See Figure 6.4(b)-(c) for examples.

6.3.3 The Importance of the Argmax-product Message Updates

Jiang et al. [2011] proposed a similar hybrid message passing algorithm, repeated here as Algorithm 6.3, which differs from our mixed-product BP only in replacing our argmax-product message update (6.15) with the standard max-product message update (6.14). We show in this section that this seemingly minor difference gives Algorithm 6.3 very different

Algorithm 6.3 Hybrid message passing by Jiang et al. [2011]

Input: A marginal MAP problem on a pairwise model $p(\boldsymbol{x}) \propto \prod_{i\in V}\psi_i \prod_{(ij)\in E}\psi_{ij}$ on graph $G = (V, E)$.

Output: An approximate marginal MAP solution $\boldsymbol{x}_B^*$.

1. Message Update:

$$\begin{aligned} &A \to A \cup B: \\ &\text{(sum-product)} \end{aligned} \qquad m_{i\to j}(x_j) \leftarrow \big[\sum_{x_i}(\psi_i(x_i)m_{\sim i}(x_i))\big(\frac{\psi_{ij}(x_i,x_j)}{m_{j\to i}(x_i)}\big)^{1/\rho_{ij}}\big]^{\rho_{ij}},$$

$$\begin{aligned} &A \to A \cup B: \\ &\text{(max-product)} \end{aligned} \qquad m_{i\to j}(x_j) \leftarrow \max_{x_i}(\psi_i(x_i)m_{\sim i}(x_i))^{\rho_{ij}}\big(\frac{\psi_{ij}(x_i,x_j)}{m_{j\to i}(x_i)}\big).$$

2. Decoding: $x_i^* = \arg\max_{x_i} b_i(x_i)$ for $\forall i \in B$, where $b_i(x_i) \propto \psi_i(x_i)m_{\sim i}(x_i)$.

properties, and fewer optimality guarantees, than our mixed-product BP.

Similar to our mixed-product BP, Algorithm 6.3 also satisfies the reparameterization property in (6.17) (with beliefs $\{b_i, b_{ij}\}$ defined by (6.16)); it also satisfies a set of similar, but crucially different, consistency conditions at its fixed points,

$$\sum_{x_i} b_{ij}(x_i, x_j) = b_j(x_j), \qquad \forall i \in A, j \in A \cup B,$$

$$\max_{x_i} b_{ij}(x_i, x_j) = b_j(x_j), \qquad \forall i \in B, j \in A \cup B,$$

which exactly map to the max- and sum-product message updates in Algorithm 6.3.

Despite its striking similarity, Algorithm 6.3 has very different properties, and does not share the appealing variational interpretation and optimality guarantees that we have demonstrated for mixed-product BP. First, it is unclear whether Algorithm 6.3 can be interpreted as a fixed point algorithm for maximizing our, or a similar, variational objective function. Second, it does not inherit the same optimality guarantees stated in Theorem 6.3, despite its similar reparameterization and consistency conditions. These disadvantages are caused by the lack of the special argmax-product message update and its associated mixed-consistency condition in (6.20), which was a critical ingredient of the proof of Theorem 6.3.

More detailed insights into Algorithm 6.3 and mixed-product BP can be obtained by considering the special case when the full graph G is an undirected tree. We show that in this case, Algorithm 6.3 can be viewed as optimizing a set of *approximate* objective functions, obtained by rearranging the max and sum operators into orders that require less computational cost, while mixed-product BP attempts to maximize the *exact* objective function by message updates that effectively perform some "asynchronous" coordinate descent steps. In the sequel, we use an illustrative toy example to explain the main ideas.

Example 6.1. *Consider a marginal MAP problem on a four node chain-structured graphical model, $x_3 - x_1 - x_2 - x_4$, where the sum and max sets are $A = \{1, 2\}$ and $B = \{3, 4\}$, respectively (see Figure 6.1b). We analyze how Algorithm 6.3 and mixed-product BP in Algorithm 6.2 behave on this toy example, when both taking Bethe weights ($\rho_{ij} = 1$ for $(ij) \in E$).*

Algorithm 6.3 (Jiang et al. [2011]). *Since G is a tree, one can show that Algorithm 3 (with Bethe weights) terminates after a full forward and backward iteration (e.g., messages passed along $x_3 \to x_1 \to x_2 \to x_4$ and then $x_4 \to x_2 \to x_1 \to x_3$). By tracking the messages, one can write its final decoded solution in a closed form,*

$$x_3^* = \arg\max_{x_3} \sum_{x_1} \sum_{x_2} \max_{x_4} [\exp(\theta(\boldsymbol{x}))], \qquad x_4^* = \arg\max_{x_4} \sum_{x_2} \sum_{x_1} \max_{x_3} [\exp(\theta(\boldsymbol{x}))],$$

On the other hand, the true marginal MAP solution is given by,

$$x_3^* = \arg\max_{x_3} \max_{x_4} \sum_{x_1} \sum_{x_2} [\exp(\theta(\boldsymbol{x}))], \qquad x_4^* = \arg\max_{x_4} \max_{x_3} \sum_{x_2} \sum_{x_1} [\exp(\theta(\boldsymbol{x}))].$$

Here, Algorithm 6.3 approximates the exact marginal MAP problem by rearranging the max and sum operators into an elimination order that makes the calculation easier. A similar property holds for the general case when G is undirected tree: Algorithm 3 (with Bethe weights) terminates in a finite number of steps, and its output solution x_i^* effectively maxi-

mizes an approximate objective function obtained by reordering the max and sum operators along a tree-order (see Definition 6.1) that is rooted at node i. The performance of the algorithm should be related to the error caused by exchanging the order of max and sum operators. However, exact optimality guarantees are likely difficult to show because it maximizes an inexact objective function. In addition, since each component x_i^* uses a different order of elimination, and hence maximizes a different surrogate objective function, it is unclear whether the joint B-configuration $\boldsymbol{x}_B^* = \{x_i^* : i \in B\}$ given by Algorithm 6.3 maximizes a single consistent objective function.

Algorithm 6.2 (mixed-product). *On the other hand, the mixed-product belief propagation in Algorithm 6.2 may not terminate in a finite number of steps, nor does it necessarily yield a closed form solution when G is an undirected tree. However, Algorithm 6.2 proceeds in an attempt to optimize the exact objective function. In this toy example, we can show that the true solution is guaranteed to be a fixed point of Algorithm 6.2. Let $b_3(x_3)$ be the mixed-belief on x_3 at the current iteration, and $x_3^* = \arg\max_{x_3} b_3(x_3)$ its unique maxima. After a message sequence passed from x_3 to x_4, one can show that $b_4(x_4)$ and x_4^* update to*

$$x_4^* = \arg\max_{x_4} b_4(x_4), \qquad b_4(x_4) = \sum_{x_2} \sum_{x_1} \exp(\theta([x_3^*, x_{\neg 3}])) = \exp(Q([x_3^*, x_4]; \boldsymbol{\theta})),$$

where we maximize the exact objective function $Q([x_3, x_4]; \boldsymbol{\theta})$ with fixed $x_3 = x_3^*$. Therefore, on this toy example, one sweep ($x_3 \to x_4$ or $x_4 \to x_3$) of Algorithm 6.2 is effectively performing a coordinate descent step, which monotonically improves the true objective function towards a local maximum. In more general models, Algorithm 6.2 differs from sequential coordinate descent, and does not guarantee monotonic convergence. But, it can be viewed as a "parallel" version of coordinate descent, which ensures the stronger local optimality guarantees shown in Theorem 6.3.

6.4 Proximal Point Algorithms

An obvious disadvantage of mixed-product BP is its lack of convergence guarantees, even when G is an undirected tree. In this section, we apply a proximal point approach [e.g., Martinet, 1970, Rockafellar, 1976] to derive convergent algorithms that directly optimize our variational objectives, which take the form of transforming marginal MAP into a sequence of pure (or annealed) sum-inference tasks. Similar methods have been applied to standard sum-inference [Yuille, 2002] and max-inference [Ravikumar et al., 2010].

For the purpose of illustration, we first consider the problem of maximizing the *exact* marginal MAP variational objective, $F_{mix}(\boldsymbol{\tau}, \boldsymbol{\theta}) = \langle \boldsymbol{\tau}, \boldsymbol{\theta} \rangle + H(\boldsymbol{x}_A | \boldsymbol{x}_B; \boldsymbol{\tau})$. The proximal point algorithm works by iteratively optimizing a smoothed problem,

$$\boldsymbol{\tau}^{t+1} = \arg\min_{\boldsymbol{\tau} \in \mathbb{M}} \{ -F_{mix}(\boldsymbol{\tau}, \boldsymbol{\theta}) + \lambda^t D(\boldsymbol{\tau} || \boldsymbol{\tau}^t) \},$$

where $\boldsymbol{\tau}^t$ is the solution at iteration t, and λ^t is a positive coefficient. Here, $D(\cdot||\cdot)$ is a distance, called the proximal function, which forces $\boldsymbol{\tau}^{t+1}$ to be close to $\boldsymbol{\tau}^t$; typical choices of $D(\cdot||\cdot)$ are Euclidean or Bregman distances or ψ-divergences [e.g., Teboulle, 1992, Iusem and Teboulle, 1993]. Proximal algorithms have nice convergence guarantees: the objective series $\{F_{mix}(\boldsymbol{\tau}^t\ ,\ \boldsymbol{\theta})\}$ is guaranteed to be non-increasing at each iteration, and $\{\boldsymbol{\tau}^t\}$ converges to an optimal solution, under some regularity conditions. See, for example, Rockafellar [1976], Tseng and Bertsekas [1993], Iusem and Teboulle [1993]. The proximal algorithm is closely related to the majorize-minimize (MM) algorithm [Hunter and Lange, 2004] and the convex-concave procedure [Yuille, 2002].

For our purpose, we take the proximal distance $D(\cdot||\cdot)$ to be a KL divergence between

Algorithm 6.4 Proximal point algorithm for marginal MAP (exact)

Input: A marginal MAP problem $\max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} \exp(\theta(\boldsymbol{x}))$.
Output: An (exact) marginal MAP solution $\boldsymbol{x}_B^*$.
Initialize local marginals $\boldsymbol{\tau}^0$.
for iteration t **do**

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta} + \lambda^t \log \boldsymbol{\tau}_B^t, \tag{6.21}$$

$$\boldsymbol{\tau}^{t+1} = \arg\max_{\tau \in \mathbb{M}} \{\langle \boldsymbol{\tau}, \boldsymbol{\theta}^{t+1} \rangle + H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau}) + \lambda^t H(\boldsymbol{x}_B; \boldsymbol{\tau})\}, \tag{6.22}$$

end for
Decoding: $x_i^* = \arg\max_{x_i} \tau_i(x_i)$ for $\forall i \in B$.
Note: we may replace $H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau})$ and $H(\boldsymbol{x}_B; \boldsymbol{\tau})$ in (6.22) *with their approximations versions, such as that in* (6.8)*; see Algorithm 6.5.*

distributions on the max nodes,

$$D(\boldsymbol{\tau}||\boldsymbol{\tau}^t) = \mathrm{KL}(\tau_B(\boldsymbol{x}_B)||\tau_B^t(\boldsymbol{x}_B)) = \sum_{\boldsymbol{x}_B} \tau_B(\boldsymbol{x}_B) \log \frac{\tau_B(\boldsymbol{x}_B)}{\tau_B^t(\boldsymbol{x}_B)}.$$

In this case, the proximal point algorithm reduces to Algorithm 6.4, which iteratively solves a smoothed variational objective, with natural parameter $\boldsymbol{\theta}^t$ updated at each iteration. Intuitively, the proximal inner loop (6.21)-(6.22) essentially "adds back" the truncated entropy term $H_B(\boldsymbol{\tau})$, while canceling its effect by adjusting $\boldsymbol{\theta}$ in the opposite direction. Typical choices of λ^t include $\lambda^t = 1$ (constant) and $\lambda^t = 1/t$ (harmonic). Note that the proximal approach is distinct from an annealing method, which would require that the annealing coefficient vanish to zero.

Interestingly, if we take $\lambda^t = 1$, then the inner maximization problem (6.22) reduces to the standard log-partition function duality (4.2), corresponding to a pure marginalization task. This has the interpretation of transforming the marginal MAP problem into a sequence of standard sum-inference problems.

In practice we also approximate $H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau})$ and $H(\boldsymbol{x}_B; \boldsymbol{\tau})$ by the pairwise entropy decomposition $\hat{H}(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau})$ and $\hat{H}(\boldsymbol{x}_B; \boldsymbol{\tau})$ in (6.8), and solve the inner optimization (6.22) by the

Algorithm 6.5 Proximal point algorithm for marginal MAP (pairwise approximation)

Input: A marginal MAP problem on a pairwise model $p(\boldsymbol{x}) \propto \exp(\sum_i \theta_i + \sum_{(ij)\in E} \theta_{ij})$.
Output: An approximate marginal MAP solution $\boldsymbol{x}_B^*$.
Define proximal parameter $\{\lambda^t\}$ and $\{\rho_{ij} \colon (ij) \in E\}$ (for example, $\lambda^t = 1$ and Bethe weights: $\rho_{ij} = 1$).
Initialize local marginals $\boldsymbol{\tau}^0 = \{\tau_i, \tau_{ij} \colon i \in A \cup B, (ij) \in E\}$.
for iteration t **do**

1. For each $i \in B$ and $(ij) \in E_B$, update:

$$\theta_{ij}^{t+1} = \theta_{ij} + \lambda^t \log \boldsymbol{\tau}_{ij}^t, \qquad \theta_i^{t+1} = \theta_i + \lambda^t \log \boldsymbol{\tau}_i^t,$$

2. Run message passing in Algorithm 6.1 with fixed $\epsilon = \lambda^t$ until convergence. Update $\boldsymbol{\tau}^{t+1} = \{\tau_i, \tau_{ij}\}$ by (6.12).

end for
Decoding: $x_i^* = \arg\max_{x_i} \tau_i(x_i)$ for $\forall i \in B$.

weighted message passing in Algorithm 6.1; this gives Algorithm 6.5.

Note that the approximate version in Algorithm 6.5 does not necessarily correspond to a valid proximal point algorithm. However, if $-\hat{H}(\boldsymbol{x}_B; \boldsymbol{\tau})$ is provably convex in the sense of Definition 3.8, such that it can be reformed into

$$\hat{H}(\boldsymbol{x}_B; \boldsymbol{\tau}) = \sum_{i \in B} \kappa_i H(x_i) + \sum_{(ij)\in E_B} (\kappa_{i\to j} H(x_i|x_j) + \kappa_{ij} H(x_i, x_j)),$$

where κ_{ij}, $\kappa_{i\to j}$, κ_i are all non-negative numbers, then the resulting approximate algorithm can still be interpreted as a proximal algorithm that maximizes $\hat{F}_{mix}(\boldsymbol{\tau}, \boldsymbol{\theta})$ with a proximal function defined as

$$\begin{aligned} D_{pair}(\boldsymbol{\tau}||\boldsymbol{\tau}^t) = &\sum_{i\in B} \kappa_i \mathrm{KL}[\tau_i(x_i)||\tau_i^0(x_i)] \\ &+ \sum_{(ij)\in E_B} \kappa_{i\to j} \mathrm{KL}[(\tau_{ij}(x_i|x_j)||\tau_{ij}^0(x_i|x_j)] + \kappa_{ij} \mathrm{KL}[(\tau_{ij}(x_i, x_j)||\tau_{ij}^0(x_i, x_j)]. \end{aligned}$$

In this case, Algorithm 6.4 is still a valid proximal algorithm and inherits their convergence guarantees. In practice, however, one often uses approximations that are not provably con-

vex. An interesting special case is when both $H(\boldsymbol{x}_A|\boldsymbol{x}_B;\boldsymbol{\tau})$ and $H(\boldsymbol{x}_B;\boldsymbol{\tau})$ are approximated by a Bethe approximation, in which case the optimization (6.22) can be solved using standard sum-product BP. Although the Bethe form for $H(\boldsymbol{x}_A|\boldsymbol{x}_B;\boldsymbol{\tau})$ and $H(\boldsymbol{x}_B;\boldsymbol{\tau})$ is not provably convex unless G is tree structured (or has some other special properties), we find in practice that this approximation gives very accurate solutions, even on general loopy graphs where its convergence is no longer theoretically guaranteed.

The global convergence guarantees of the proximal point algorithm may also fail if the inner update (6.22) is not solved exactly. It should also be possible to develop globally convergent algorithms without inner loops using the techniques that have been developed for full marginalization or MAP problems [e.g., Meltzer et al., 2009, Hazan and Shashua, 2010, Jojic et al., 2010, Savchynskyy et al., 2010], but we leave this to future work.

6.5 Connections to EM

A natural algorithm for solving the marginal MAP problem is to use the expectation-maximization (EM) algorithm, by treating $\boldsymbol{x}_A$ as the hidden variables and $\boldsymbol{x}_B$ as the "parameters" to be maximized. In this section, we show that the EM algorithm can be seen as a coordinate ascent algorithm on a mean field variant of our framework.

We start by introducing a "non-convex" generalization of the variational form (6.3).

Corollary 6.1. *Let $\mathbb{M}^o$ be the subset of the marginal polytope $\mathbb{M}$ corresponding to the distributions in which $\boldsymbol{x}_B$ are clamped to some deterministic values, that is,*

$$\mathbb{M}^o = \{\boldsymbol{\tau} \in \mathbb{M} : \exists \boldsymbol{x}_B^* \in \mathcal{X}_B, \text{ such that } \tau(\boldsymbol{x}_B) = \mathbf{1}(\boldsymbol{x}_B = \boldsymbol{x}_B^*)\}.$$

Then the variational form (6.3) *remains exact if the marginal polytope $\mathbb{M}$ is replaced by any*

$\mathbb{N}$ *satisfying* $\mathbb{M}^o \subseteq \mathbb{N} \subseteq \mathbb{M}$*, that is,*

$$\Phi_{AB} = \max_{\boldsymbol{\tau} \in \mathbb{N}} \{\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}_A | \boldsymbol{x}_B; \boldsymbol{\tau})\}.$$

Proof. For an arbitrary marginal MAP solution $\boldsymbol{x}_B^*$, the $\boldsymbol{\tau}^*$ with $\tau^*(\boldsymbol{x}) = p(\boldsymbol{x} | \boldsymbol{x}_B = \boldsymbol{x}_B^*; \boldsymbol{\theta})$ is an optimum of (6.3) and satisfies $\boldsymbol{\tau}^* \in \mathbb{M}^o$. Therefore, restricting the optimization on $\mathbb{M}^o$ (or any $\mathbb{N}$) does not change the maximum value of the objective function. □

Remark. Among all $\mathbb{N}$ satisfying $\mathbb{M}^o \subseteq \mathbb{N} \subseteq \mathbb{M}$, the marginal polytope $\mathbb{M}$ is the smallest (and the unique) convex set that includes $\mathbb{M}^o$, i.e., it is the convex hull of $\mathbb{M}^o$.

To connect to EM, we define $\mathbb{M}^\times$, the set of distributions in which $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ are independent, that is, $\mathbb{M}^\times = \{\boldsymbol{\tau} \in \mathbb{M} \colon \tau(\boldsymbol{x}) = \tau(\boldsymbol{x}_A)\tau(\boldsymbol{x}_B)\}$. Since $\mathbb{M}^o \subset \mathbb{M}^\times \subset \mathbb{M}$, the dual optimization (6.3) remains exact when restricted to $\mathbb{M}^\times$, that is,

$$\Phi_{AB}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau} \in \mathbb{M}^\times} \{\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}_A | \boldsymbol{x}_B; \boldsymbol{\tau})\} = \max_{\boldsymbol{\tau} \in \mathbb{M}^\times} \{\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}_A; \boldsymbol{\tau})\},$$

where the second equality holds because $H(\boldsymbol{x}_A | \boldsymbol{x}_B; \boldsymbol{\tau}) = H(\boldsymbol{x}_A; \boldsymbol{\tau})$ for $\boldsymbol{\tau} \in \mathbb{M}^\times$.

Although $\mathbb{M}^\times$ is no longer a convex set, it is natural to consider a coordinate update that alternately optimizes $\tau(\boldsymbol{x}_A)$ and $\tau(\boldsymbol{x}_B)$,

$$\begin{aligned} &\text{Updating sum part :} && \boldsymbol{\tau}_A^{t+1} \leftarrow \operatorname*{argmax}_{\boldsymbol{\tau}_A \in \mathbb{M}_A} \{\langle \mathbb{E}_{\tau_B^t}(\boldsymbol{\theta}), \boldsymbol{\tau}_A \rangle + H(\boldsymbol{x}_A; \boldsymbol{\tau}_A)\}, \\ &\text{Updating max part :} && \boldsymbol{\tau}_B^{t+1} \leftarrow \operatorname*{argmax}_{\boldsymbol{\tau}_B \in \mathbb{M}_B} \langle \mathbb{E}_{\tau_A^{t+1}}(\boldsymbol{\theta}), \boldsymbol{\tau}_B \rangle, \end{aligned} \tag{6.23}$$

where $\mathbb{M}_A$ and $\mathbb{M}_B$ are the marginal polytopes over $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$, respectively. Note that the sum and max step each happen to be the dual of a sum-inference and max-inference problem, respectively. If we go back to the primal, and update the primal configuration $\boldsymbol{x}_B$

instead of $\boldsymbol{\tau}_B$, (6.23) can be rewritten into

$$\begin{aligned} \text{E step}: &\qquad \tau_A^{t+1}(\boldsymbol{x}_A) \leftarrow p(\boldsymbol{x}_A|\boldsymbol{x}_B^t;\boldsymbol{\theta}), \\ \text{M step}: &\qquad \boldsymbol{x}_B^{t+1} \leftarrow \arg\max_{\boldsymbol{x}_B} \mathbb{E}_{\tau_A^{t+1}}(\boldsymbol{\theta}), \end{aligned}$$

which is exactly the EM update, viewing $\boldsymbol{x}_B$ as parameters and $\boldsymbol{x}_A$ as hidden variables. Similar connections between EM and the coordinate ascent method on variational objectives have been discussed in Neal and Hinton [1998] and Wainwright and Jordan [2008].

When the E-step or M-step are intractable, one can insert various approximations. In particular, approximating $\mathbb{M}_A$ by a mean-field inner bound $\mathbb{M}_A^{mf}$ leads to variational EM. An interesting result is obtained by using the Bethe approximation to solve the E-step and a linear relaxation to solve the M-step; in this case, the EM-like update is equivalent to solving

$$\max_{\boldsymbol{\tau}\in\mathbb{L}^\times} \big\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{i\in A} H_i(\tau_i) \ - \sum_{(ij)\in E_A} I_{ij}(\tau_{ij}) \big\}, \tag{6.24}$$

where $\mathbb{L}^\times$ is the subset of $\mathbb{L}$ in which $\tau_{ij}(x_i, x_j) = \tau_i(x_i)\tau_j(x_j)$ for $(ij) \in \partial_{AB}$. Equivalently, $\mathbb{L}^\times$ is the subset of $\mathbb{L}$ in which $I_{ij}(\tau_{ij}) = 0$ for $(ij) \in \partial_{AB}$. Therefore, (6.24) can be treated as a special case of the pairwise entropy approximation in (6.7) by taking $\rho_{ij} \to +\infty$, forcing the solution τ^* to fall into $\mathbb{L}^\times$. As we discussed in Section 6.2.3, EM represents an extreme of the tradeoff between convexity and integrality implied by Theorem 6.1, which strongly encourages vertex solutions by sacrificing convexity, and hence is likely to become stuck in local optima.

6.6 Factor Graph BP for Marginal MAP

In the preceding sections, we have restricted our discussion to pairwise models and pairwise entropy approximations, mainly for the purpose of clarity. It is straightforward to extend

our methodology and derive similar algorithms on more general models with higher order factors. In this section, we discuss the extension to factor graphs, and derive a mixed-product factor graph BP. We will also discuss the case of junction graph in Section 6.7. Other higher order methods, like generalized BP [Yedidia et al., 2005] or their convex variants [Wainwright et al., 2005, Wiegerinck, 2005], can be derived similarly, but are left for future work.

Given a factorized distribution $p(\boldsymbol{x}) \propto \prod_{\alpha\in\mathcal{I}} \psi_\alpha(\boldsymbol{x}_\alpha)$, its factor graph is a bipartite graph $G = (V, \mathcal{I}, E)$ consists of variable nodes $V = \{i\}$, factor nodes $\mathcal{I} = \{\alpha\}$, and edges between factors and their associated variables $E = \{(i, \alpha) \in V \times \mathcal{I} \colon i \in \alpha\}$. We denote by $\partial(i)$ the set of factors that includes node i, that is, $\partial(i) = \{\alpha \in \mathcal{I} \colon i \in \alpha\}$. For factor graphs, the following entropy decomposition derives classical sum-product BP,

$$H(\boldsymbol{x}; \boldsymbol{\tau}) \;\approx\; \sum_{\alpha\in\mathcal{I}} H(\boldsymbol{x}_\alpha; \tau_\alpha) - \sum_{i\in V} (|\partial(i)| - 1) H(x_i; \tau_i) \;=\; \sum_{i\in V} H(x_i, ; \tau_i) - \sum_{\alpha\in\mathcal{I}} \tilde{I}(\boldsymbol{x}_\alpha; \tau_\alpha),$$

where $\tilde{I}(\boldsymbol{x}_\alpha; \tau_\alpha)$ is a "mutual information" defined by

$$\tilde{I}(\boldsymbol{x}_\alpha; \tau_\alpha) = \sum_{i\in\alpha} H(x_i; \tau_i) - H(\boldsymbol{x}_\alpha; \tau_\alpha).$$

For our marginal MAP problem, denote by α_A and α_B the sum and max variables included in factor α, that is, $\alpha_A = \alpha \cap A$, $\alpha_B = \alpha \cap B$ and $\alpha_A \cup \alpha_B = \alpha$. We can approximate $H(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau})$ and $H(\boldsymbol{x}_B; \boldsymbol{\tau})$, respectively, via

$$\tilde{H}(\boldsymbol{x}_A|\boldsymbol{x}_B; \boldsymbol{\tau}) = \sum_{i\in A} H(x_i; \tau_i) - \sum_{\alpha\in\mathcal{I}} \tilde{I}(\boldsymbol{x}_{\alpha_A}|\boldsymbol{x}_{\alpha_B}; \tau_\alpha),$$
$$\tilde{H}(\boldsymbol{x}_B; \boldsymbol{\tau}) = \sum_{i\in B} H(x_i; \tau_i) - \sum_{\alpha\in\mathcal{I}} \tilde{I}(\boldsymbol{x}_\beta; \tau_\alpha).$$

Algorithm 6.6 Mixed-product factor graph BP for marginal MAP

Input: Graphical model $p(\boldsymbol{x}) \propto \prod_{i\in V}\psi_i(x_i)\prod_{\alpha\in\mathcal{I}}\psi_\alpha(\boldsymbol{x}_\alpha)$; Sum nodes A and max nodes $B = V \setminus A$. For each factor α, let $\alpha_A = \alpha \cap A$ and $\sigma_B = \alpha \cap B$.

Output: an approximate marginal MAP solution $\boldsymbol{x}_B^*$.

1. Passing messages between the factors and variables until convergence:

$$\text{All Variables to Factors:} \quad m_{i\to\alpha}(x_i) \propto \prod_{\alpha'\in\partial(i)\setminus\{\alpha\}} m_{\alpha'\to i}(x_i)$$

$$\text{Factors to Max Variables:} \quad m_{\alpha\to i}(x_i) \propto \max_{\boldsymbol{x}_{\alpha_B\setminus\{i\}}} \sum_{\boldsymbol{x}_{\alpha_A}} \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{i'\in\alpha\setminus\{i\}} m_{i'\to\alpha}(x_{i'})$$

$$\text{Factors to Sum Variables:} \quad m_{\alpha\to i}(x_i) \propto \sum_{\boldsymbol{x}_{\alpha\setminus\{i\}}} \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{i'\in\alpha\setminus\{i\}} m_{i'\to\alpha}(x_{i'}) \times \mathbf{1}[\boldsymbol{x}_{\alpha_B} \in \mathcal{X}^*_{\alpha_B}]$$

$$\text{where} \quad \mathcal{X}^*_{\alpha_B} = \arg\max_{\boldsymbol{x}_{\alpha_B}} \Big\{ b_\alpha(\boldsymbol{x}_{\alpha_B}) \equiv \sum_{\boldsymbol{x}_{\alpha_A}} \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{i\in\alpha} m_{i\to\alpha}(x_i) \Big\}.$$

2. Decoding solution: $x_i^* = \arg\max_x \big\{ b_i(x_i) \equiv \prod_{\alpha\in\partial(i)} m_{\alpha\to i}(x_i) \big\},\ \forall i \in B.$

where

$$\tilde{I}(\boldsymbol{x}_{\alpha_A}|\boldsymbol{x}_{\alpha_B};\tau_\alpha) = \sum_{i\in\alpha_A} H(x_i;\tau_\alpha) - H(\boldsymbol{x}_{\alpha_A}|\boldsymbol{x}_{\alpha_B};\tau_\alpha).$$

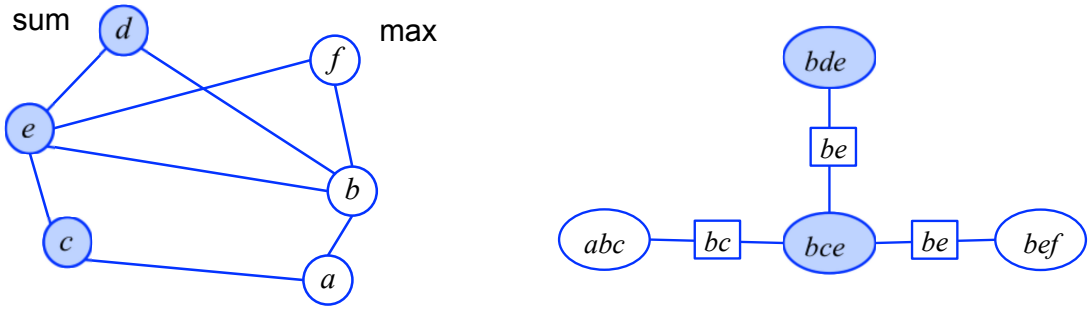
We then approximate the marginal MAP variational optimization via

$$\max_{\boldsymbol{\tau}} \sum_i \theta_i(x_i)\tau_i(x_i) + \sum_\alpha \theta_\alpha(\boldsymbol{x}_\alpha)\tau_\alpha(\boldsymbol{x}_\alpha) + \tilde{H}(\boldsymbol{x}_A|\boldsymbol{x}_B;\boldsymbol{\tau}) + \epsilon\tilde{H}(\boldsymbol{x}_B;\boldsymbol{\tau}) \tag{6.25}$$

$$\text{s.t.} \quad \sum_{\boldsymbol{x}_{\alpha\setminus\{i\}}} \tau_\alpha(\boldsymbol{x}_\alpha) = \tau_i(x_i), \quad \tau_\alpha(\boldsymbol{x}_\alpha) \geq 0, \quad \sum_{x_i} \tau_i(x_i) = 1, \quad \forall i \in V, \alpha \in \mathcal{I}, \boldsymbol{x} \in \mathcal{X}.$$

where ϵ is an annealing parameter. Using the Lagrangian multiplier method similar to that used in Algorithm 6.2 for pairwise models, and setting $\epsilon \to 0^+$, we get the factor graph mixed-product BP shown in Algorithm 6.6. See Appendix B.4 for detailed derivation.

Similar to the pairwise mixed-product BP in Algorithm 6.2 , Algorithm 6.6 also admits a



(a) (b)

Figure 6.5: (a) An example marginal MAP problem, where d, c, e are sum nodes (shaded) and a, b, f are max nodes. (b) A junction graph of (a) with the sum clusters (shaded), where max node a is assigned to max cluster $\{abc\}$ and f to $\{bef\}$; the node b can be assigned to either $\{abc\}$ or $\{bef\}$.

similar intuitive interpretation: the messages from factor α to max variables ($i \in B$) simply eliminate the variables in α by a hybrid of sum-product (on α_A) and max-product (on α_B); while the messages from factor to sum variables perform a argmax-product update similar to that of the pairwise case. Similar reparameterization and optimality properties can be established analogously to the pairwise case.

6.7 Junction Graph BP for Marginal MAP

In this section, we derive a mixed-product BP algorithm on junction graphs [Mateescu et al., 2010, Koller and Friedman, 2009]. Consider a general factorized distribution $p(\boldsymbol{x}) \propto \prod_{\alpha \in \mathcal{C}} \Phi_\alpha(\boldsymbol{x}_\alpha)$. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a junction graph of $p(\boldsymbol{x})$, with clusters $\mathcal{C} = \{c_k \colon k \in \mathcal{V}\}$ and separators $\mathcal{S} = \{s_{kl} \colon (kl) \in \mathcal{E}\}$. To represent the marginal MAP problem under the junction graph, we associate each max variable ($i \in B$) with a *unique* cluster c_k (called a its *max-clusters*) which include i (i.e., $i \in c_k$). Let $\mathcal{B}$ be the set of max clusters; correspondingly, call $\mathcal{A} = \mathcal{V} \setminus \mathcal{B}$ the *sum-clusters*. See Figure 6.5 for an illustration.

Let π_k be the set of max variables assigned to cluster c_k; note that $\{\pi_k \colon k \in \mathcal{B}\}$ forms a non-overlapping partition over the variable nodes B, that is, $\cup_{k \in \mathcal{V}} \pi_k = B$ and $\pi_k \cap \pi_l =$

$\emptyset, \forall k, l \in \mathcal{V}$. We can approximate the joint entropy $H(\boldsymbol{x}; \boldsymbol{\tau})$ and marginal entropy $H(\boldsymbol{x}_B; \boldsymbol{\tau})$, respectively, by

$$H(\boldsymbol{x}; \boldsymbol{\tau}) \approx \sum_{k \in \mathcal{V}} H(\boldsymbol{x}_{c_k}; \boldsymbol{\tau}) - \sum_{(kl) \in \mathcal{E}} H(\boldsymbol{x}_{s_{kl}}; \boldsymbol{\tau}), \qquad H(\boldsymbol{x}_B; \boldsymbol{\tau}) \approx \sum_{k \in \mathcal{B}} H_{\pi_k}(\boldsymbol{\tau}),$$

where $H(\boldsymbol{x}_{c_k}; \boldsymbol{\tau})$, $H(\boldsymbol{x}_{s_{kl}}; \boldsymbol{\tau})$ and $H_{\pi_k}(\boldsymbol{\tau})$ are the entropy of the local marginals τ_{c_k}, $\tau_{s_{kl}}$ and τ_{π_k}, respectively.

We then replace $\mathbb{M}$ with the locally consistent polytope $\mathbb{L}(\mathcal{C})$, which is the set of local marginals $\boldsymbol{\tau} = \{\tau_{c_k}, \tau_{s_{kl}} \colon k \in \mathcal{V}, (kl) \in \mathcal{E}\}$ that are consistent on the intersections of the clusters and separators, that is,

$$\mathbb{L}(\mathcal{C}) = \{\boldsymbol{\tau} \colon \sum_{x_{c_k \setminus s_{kl}}} \tau_{c_k}(x_{c_k}) = \tau(x_{s_{kl}}), \tau_{c_k}(x_{c_k}) \geq 0, \text{for } \forall\ k \in \mathcal{V}, (kl) \in \mathcal{E}\}.$$

Clearly, we have $\mathbb{M} \subseteq \mathbb{L}(\mathcal{C})$ and that $\mathbb{L}(\mathcal{C})$ is tighter than the pairwise polytope $\mathbb{L}$ we used previously.

Overall, the marginal MAP dual form in (6.3) is approximated by

$$\max_{\boldsymbol{\tau} \in \mathbb{L}(\mathcal{C})} \big\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{k \in \mathcal{A}} H(\boldsymbol{x}_{c_k}; \boldsymbol{\tau}) + \sum_{k \in \mathcal{B}} H_\epsilon(\boldsymbol{x}_{c_k} | \boldsymbol{x}_{\pi_k}; \boldsymbol{\tau}) - \sum_{(kl) \in \mathcal{E}} H(\boldsymbol{x}_{s_{kl}}; \boldsymbol{\tau}) \big\} \tag{6.26}$$

where $H_\epsilon(\boldsymbol{x}_{c_k} | \boldsymbol{x}_{\pi_k}; \boldsymbol{\tau}) = H(\boldsymbol{x}_{c_k}; \boldsymbol{\tau}) - (1 - \epsilon) H(\boldsymbol{x}_{\pi_k}; \boldsymbol{\tau})$, and ϵ is a small annealing parameter. Optimizing (6.26) using a method similar to the derivation of mixed-product BP in Algorithm 6.2, and taking $\epsilon \to 0^+$, we obtain a "mixed-product" junction graph belief propagation, given in Algorithm 6.7.

Remarks. (1) The mixed-product junction graph BP in Algorithm 6.7 differs in several key ways from the mixed BP algorithms for pairwise models and factor graphs presented earlier. In particular, Algorithm 6.7 requires a special step that assigns each max node to

Algorithm 6.7 Mixed-product junction graph BP for marginal MAP

Input: A marginal MAP problem on a graphical model $p(\boldsymbol{x}) \propto \prod_{c_k \in \mathcal{C}} \psi_{c_k}(\boldsymbol{x}_{c_k})$ and its junction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with clusters $\mathcal{C} = \{c_k\}$ and separators $\mathcal{S} = \{s_{kl}\}$.
Output: An approximate marginal MAP solution $\boldsymbol{x}^*_B$.

1. Define an max/sum partition on the clusters: Assign each max node $i \in B$ to a unique max cluster $c_k \in \mathcal{B}$ that includes i. Let $\mathcal{A} = \mathcal{V} \setminus \mathcal{B}$ be the (remaining) sum clusters. Denote by $\pi_k, k \in \mathcal{B}$ the set of max nodes assigned to cluster c_k.
2. Passing messages between clusters on the junction graph until convergence:

$$\begin{aligned}
&\mathcal{A} \to \mathcal{A} \cup \mathcal{B}\text{:} \\
&\text{(sum-product)} \qquad m_{k\to l}(x_{s_{kl}}) \propto \sum_{x_{c_k \setminus s_{kl}}} \psi_{c_k}(x_{c_k}) m_{\sim k \setminus l}(x_{c_k}), \\
&\mathcal{B} \to \mathcal{A} \cup \mathcal{B}\text{:} \\
&\text{(argmax-product)} \qquad m_{k\to l}(x_{s_{kl}}) \propto \sum_{x_{c_k \setminus s_{kl}}} (\psi_{c_k}(x_{c_k}) m_{\sim k \setminus l}(x_{c_k})) \cdot \mathbf{1}[x_{\pi_k} \in \mathcal{X}^*_{\pi_k}], \\
&\text{where} \quad \mathcal{X}^*_{\pi_k} = \arg\max_{x_{\pi_k}} \sum_{x_{c_k \setminus \pi_k}} b_k(x_{c_k}), \\
&\qquad b_k(x_{c_k}) = \psi_{c_k}(x_{c_k}) \prod_{k' \in \partial(k)} m_{k' \to k}(x_{s_{k'k}}) \quad \text{and} \quad m_{\sim k \setminus l}(x_{c_k}) = \prod_{k' \in \partial(k) \setminus \{l\}} m_{k' \to k}(x_{s_{k'k}}).
\end{aligned}$$

3. Decoding: $\boldsymbol{x}^*_{\pi_k} = \arg\max_{x_{\pi_k}} \sum_{x_{c_k \setminus \pi_k}} b_k(x_{c_k})$ for $\forall k \in \mathcal{B}$.

a unique cluster to form the max- / sum- cluster partition in the junction graph, and does not reduce to the other mixed-product BP algorithms (for example, it consists of only two types of message, rather than three). These differences can be viewed as a consequence of the "simpler" entropy approximation to $H(\boldsymbol{x}_B; \boldsymbol{\tau})$ selected for join graphs, which simplifies the derivation.

(2) On the other hand, Algorithm 6.7 also admits an intuitive reparameterization interpretation and a strong local optimality guarantee similar to the pairwise mixed-product BP in Algorithm 6.2. In fact, Algorithm 6.7 is a special case of Algorithm 7.1 for solving maximum expected utility tasks in decision networks, which we develop and study in detail in Chapter 7. For this reason, we defer a more in-depth study of the algorithm to the next chapter; see Chapter 7 for more details.

6.8 Related Work

Expectation-maximization (EM) or variational EM provide one straightforward approach for marginal MAP, by viewing the sum nodes as hidden variables and the max nodes as parameters to be estimated; however, EM is prone to getting stuck at sub-optimal configurations. We show that EM can be treated as a special case of our framework when a mean field-like approximation is applied. Other classical state-of-the-art approaches include local search methods [e.g., Park and Darwiche, 2004], Markov chain Monte Carlo methods [e.g., Doucet et al., 2002, Yuan et al., 2004], and variable elimination-based methods [e.g., Dechter and Rish, 2003, Mauá and de Campos, 2012]. Jiang et al. [2011] proposed a hybrid message passing algorithm that has a similar form to our mixed-product BP algorithm, but without theoretical guarantees; we show in Section 6.3.3 that Jiang et al. [2011] can be viewed in some sense as an approximation of the marginal MAP problem that exchanges the order of sum and max operators. Another message-passing-style algorithm was proposed in Altarelli et al. [2011] for general multi-stage stochastic optimization problems based on survey propagation; compared to our approach, this algorithm has a relatively complex form (making it more difficult to interpret) and also does not provide any known optimality guarantees. Finally, Ibrahimi et al. [2011] introduces a robust max-product belief propagation for solving a related worst-case robust optimization problem, in which the hidden variables are minimized instead of marginalized. To the best of our knowledge, our work is the first general variational framework for marginal MAP, and provides the first strong optimality guarantees.

6.9 Experiments

We illustrate our algorithms' performance on both simulated models and more realistic diagnostic Bayesian network models taken from the UAI'08 inference challenge. We show that our Bethe approximation algorithms perform best among all the tested algorithms,

including Jiang et al. [2011]'s hybrid message passing and a state-of-the-art local search algorithm [Park and Darwiche, 2004].

6.9.1 Experiment Settings

We implement our mixed-product BP in Algorithm 6.2 with Bethe weights (`mix-product (Bethe)`), the regular sum-product BP (`sum-product`), max-product BP (`max-product`) and Jiang et al. [2011]'s hybrid message passing (with Bethe weights) in Algorithm 6.3 (`Jiang's method`), where the solutions are all extracted by maximizing the singleton marginals of the max nodes. For all these algorithms, we run a maximum of 50 iterations; if they fail to converge, we run 100 additional iterations with a damping coefficient of $\beta = 0.1$ (see equation (3.19)). We initialize all these algorithms with 5 random initializations and pick the best solution; for `mix-product (Bethe)` and `Jiang's method`, we run an additional trial initialized using the sum-product messages, which was reported to perform well in Park and Darwiche [2004] and Jiang et al. [2011]. We also run the proximal point algorithm with both $H(\boldsymbol{x}_A|\boldsymbol{x}_B;\boldsymbol{\tau})$ and $H(\boldsymbol{x}_B;\boldsymbol{\tau})$ approximated by Bethe approximations (`Proximal (Bethe)`), which is Algorithm 6.5 with $\rho_{ij} = 1,\ \forall (ij) \in E$ and $\lambda^t = 1,\ \forall t$.

We also implement the TRW approximation, but only using the convergent proximal point algorithm, because the TRW upper bounds are valid only when the algorithms converge. The TRW weights of $\hat{H}_{A|B}$ are constructed by first (randomly) selecting spanning trees of G_A, and then augmenting each spanning tree with one uniformly selected edge in ∂_{AB}. The TRW weights of $\hat{H}_B(\boldsymbol{\tau})$ are constructed to be provably convex, by constructing monotonic chains using a greedy method in Kolmogorov [2006, Section 4]: in short, we select a chain-structured spanning tree of G_B along a predefined node ordering such that it is not possible to further extend the chain, after which we remove the edges of the given chain from the graph and repeat the procedure until no edges are left; this constructs a set of chains that cover the graph. We then assign each chain a uniform probability and calculate the corresponding

edge appearance probability. We run all the proximal point algorithms for a maximum of 100 iterations, with a maximum of 5 iterations of weighted message passing updates (6.11)-(6.12) for the inner loops (with 5 additional damped updates with 0.1 damping coefficient).

In addition, we compare our algorithms with SamIam, which is a state-of-the-art implementation of the local search algorithm for marginal MAP [Park and Darwiche, 2004]; we use its default Taboo search method with a maximum of 500 searching steps, and report the best results among 5 trials with random initializations, and one additional trial initialized by its default method (which sequentially initializes x_i by maximizing $p(x_i|x_{\mathrm{pa}_i})$ along some predefined order).

We also implement an EM algorithm, whose expectation and maximization steps are approximated by sum-product and max-product BP, respectively. We run EM with 5 random initializations and one initialization by sum-product marginals, and pick the best solution.

6.9.2 Simulated Models

We consider pairwise models over discrete random variables taking values in $\{-1, 0, +1\}^n$,

$$p(\boldsymbol{x}) \propto \exp\big[\sum_i \theta_i(x_i) + \sum_{(ij)\in E} \theta_{ij}(x_i, x_j)\big].$$

The value tables of θ_i and θ_{ij} are randomly generated from normal distribution, $\theta_i(k) \sim \mathrm{Normal}(0, 0.01)$, $\theta_{ij}(k, l) \sim \mathrm{Normal}(0, \sigma^2)$, where σ controls the strength of coupling. Our results are averaged on 1000 randomly generated sets of parameters.

We consider different choices of graph structures and max / sum node patterns:

1. *Hidden Markov chains* with 20 nodes (see Figure 6.6(a)), whose intractability was illustrated in Figure 3.2.
2. *Latent tree models.* We generate random trees of size 50, by finding the minimum span-

ning trees of random symmetric matrices with elements drawn from Uniform($[0, 1]$). We take the leaf nodes to be max nodes, and the non-leaf nodes to be sum nodes. See Figure 6.7(a) for a typical example.

3. 10×10 *Grid* with max and sum nodes distributed in two opposite chess board patterns shown in Figure 6.8(a) and Figure 6.9(a), respectively. In Figure 6.8(a), the sum part is a loopy graph, and the max part is a (fully disconnected) tree; in Figure 6.9(a), the max and sum parts are flipped.

The results on the hidden Markov chain are shown in Figure 6.6, where we plot in panel (b) their relative energy errors defined by $Q(\hat{\boldsymbol{x}}_B; \boldsymbol{\theta}) - Q(\boldsymbol{x}_B^*; \boldsymbol{\theta})$, where $\hat{\boldsymbol{x}}_B$ is the solution returned by the algorithms, and $\boldsymbol{x}_B^*$ is the true optimum, and in panel (c) different algorithms' percentages of obtaining the globally optimal solutions among 1000 random trials.

The results of the latent tree models and the two types of 2D grids are shown in Figure 6.7, Figure 6.8 and Figure 6.9, respectively. On these problems, although it is tractable to evaluate $Q(\hat{\boldsymbol{x}}_B; \boldsymbol{\theta})$ for fixed $\hat{\boldsymbol{x}}_B$ (since G is a tree), it is intractable to calculate the globally optimal solution $\boldsymbol{x}_B^*$ due to the large A-B constrained induced width. Therefore, we report the approximate relative error defined by $Q(\hat{\boldsymbol{x}}_B; \boldsymbol{\theta}) - Q(\tilde{\boldsymbol{x}}_B; \boldsymbol{\theta})$, where $\tilde{\boldsymbol{x}}_B$ is the best solution we found across all algorithms.

6.9.3 Diagnostic Bayesian Networks

We also test our algorithms on two diagnostic Bayesian networks taken from the UAI08 Inference Challenge, where we construct marginal MAP problems by randomly selecting varying percentages of nodes to be max nodes. Since these models are not pairwise, we implement the junction graph versions of `mix-product (Bethe)` and `proximal (Bethe)` shown in Section 6.7. Figure 6.10 shows the approximate relative errors of our algorithms and `local search (SamIam)` as the percentage of the max nodes varies.

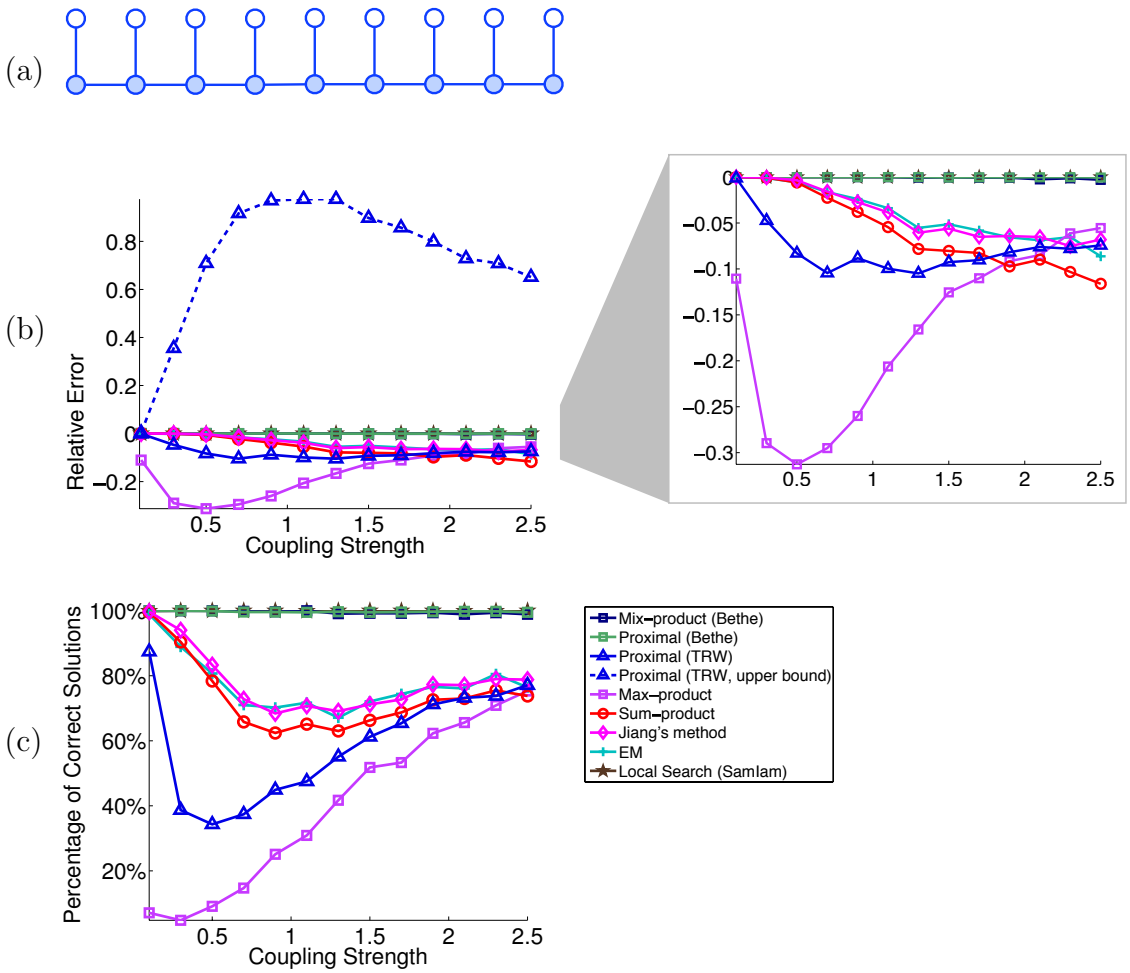
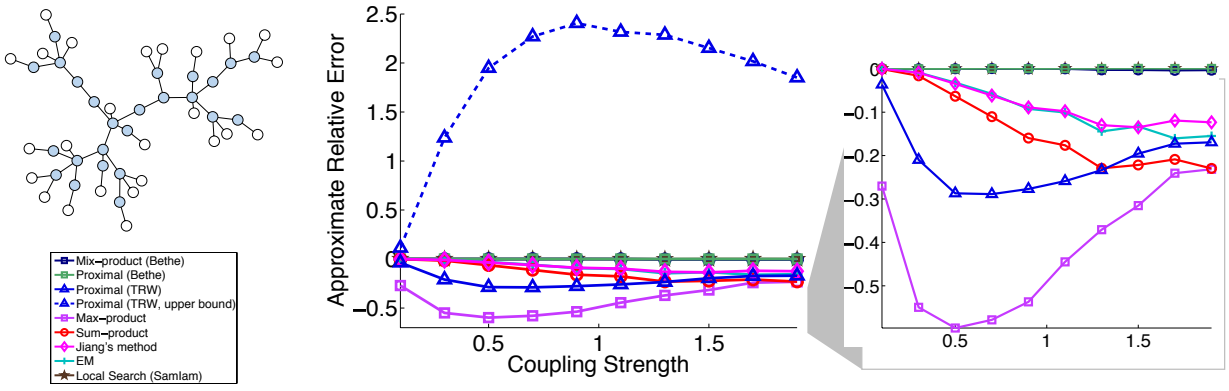


Figure 6.6: (a) A marginal MAP problem defined on a hidden Markov chain with 20 nodes, similar to Figure 3.2. (b) The relative energy errors of the different algorithms, and the upper bounds obtained by `Proximal (TRW)` as a function of coupling strength σ. (c) Different algorithms' probabilities of obtaining the globally optimal solution among 1000 random trials. `Mix-product (Bethe)`, `Proximal (Bethe)` and `Local Search (SamIam)` almost always (with probability $\geq 99\%$) found the optimal solution. (Figure best viewed in color.)

6.9.4 Insights

Across all the experiments, we find that `mix-product (Bethe)`, `proximal (Bethe)` and `local search (SamIam)` significantly outperform all the other algorithms, while `proximal (Bethe)` outperforms the two others in some circumstances. In the hidden Markov chain example in Figure 6.6, these three algorithms almost always (with probability $\geq 99\%$) find the globally optimal solutions. However, the performance of SamIam tends to degrade when



(a) (b)

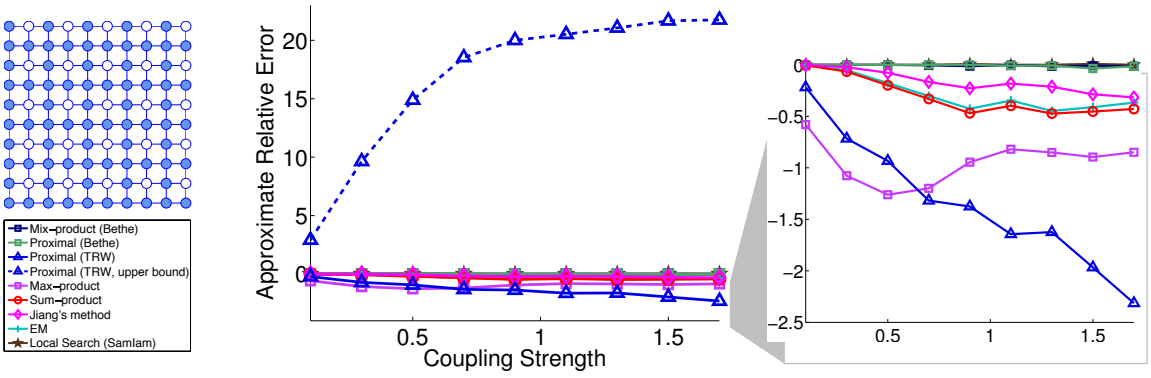
Figure 6.7: (a) A typical latent tree model, whose leaf nodes are taken to be max nodes (white) and non-leaf nodes to be sum nodes (shaded). (b) The approximate relative energy errors of different algorithms, and the upper bound obtained by `Proximal (TRW)` as a function of coupling strength σ.

the max part has loopy dependence structures (see Figure 6.9), or when the number of max nodes is large (see Figure 6.10), both of which make it difficult to explore the solution space by local search. On the other hand, `mix-product (Bethe)` tends to deteriorate as the coupling strength σ increases (see Figure 6.9), probably because its convergence behavior gets worse as σ increases.

We note that our TRW approximation gives much less accurate solutions than the other algorithms, but is able to provide an upper bound on the optimal energy. Similar phenomena have been observed for TRW-BP in standard max- and sum- inference.

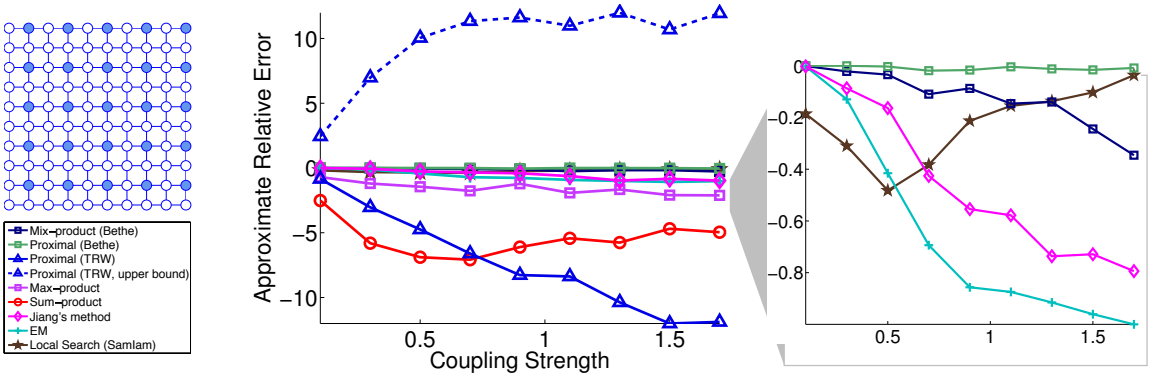
The hybrid message passing of Jiang et al. [2011] is significantly worse than `mix-product (Bethe)`, `proximal (Bethe)` and `local search (SamIam)`, but is otherwise the best among the remaining algorithms. EM performs similarly to (or sometimes worse than) Jiang's method.

The regular max-product BP and sum-product BP are among the worst of the tested algorithms, indicating the danger of approximating mixed-inference by pure max- or sum-inference. Interestingly, the performances of max-product BP and sum-product BP have



(a) (b)

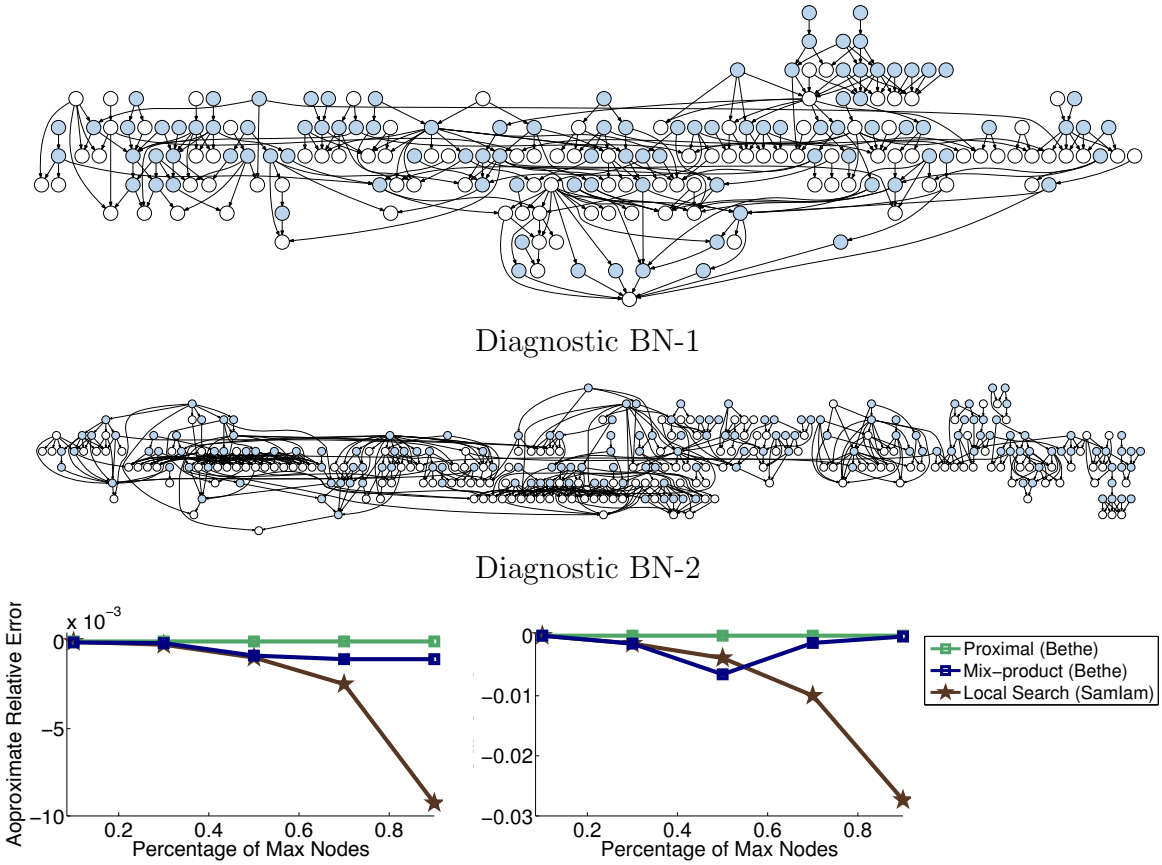
Figure 6.8: (a) A marginal MAP problem defined on a 10×10 Ising grid, with shaded sum nodes and unshaded max nodes; note that the sum part is a loopy graph, while max part is fully disconnected. (b) The approximate relative errors of different algorithms and the upper bound obtained by Proximal (TRW) as a function of coupling strength σ.



(a) (b)

Figure 6.9: (a) A marginal MAP problem defined on a 10×10 Ising grid, but with max / sum part exactly opposite to that in Figure 6.8; note that the max part is loopy, while the sum part is fully disconnected in this case. (b) The approximate relative errors of different algorithms and the upper bound obtained by Proximal (TRW) as a function of coupling strength σ.

opposite trends: In Figure 6.6, Figure 6.7 and Figure 6.8, where the max parts are fully disconnected and the sum parts are connected and loopy, max-product BP usually performs worse than sum-product BP, but gets better as the coupling strength σ increases; sum-product BP, on the other hand, tends to degrade as σ increases. In Figure 6.9, where the max / sum pattern is reversed (resulting in a larger, loopier max subgraph), max-product BP performs better than sum-product BP.



(a) Diagnostic BN-1 (b) Diagnostic BN-2

Figure 6.10: The results on two diagnostic Bayesian networks (BNs) in the UAI08 inference challenge. Top: The network structures of Diagnostic BN-1 and BN-2, each with 50% randomly selected sum nodes shaded. Bottom: The performances of algorithms on the two BNs as a function of the percentage of max nodes. The local search method tends to degrade when the number of max nodes is large, making it difficult to search over the solution space. Results are averaged over 100 random trials.

6.10 Conclusions and Future Directions

We leverage the variational representation of marginal MAP to develop a series of efficient message passing algorithms for marginal MAP. In particular, we show that our proposed "mixed-product" BP admits appealing theoretical properties and performs well in practice. Our general methodology makes it possible to extend most or all existing variational approximation and optimization techniques to marginal MAP, opening new doors to develop even more efficient algorithms.

Future Directions. This work opens many potential directions, in terms of both improving marginal MAP algorithms and applying them to real world problems with uncertain hidden variables. For example, mixed-product BP tends to have poor convergence properties, perhaps due to its special argmax update; advanced scheduling strategies, such residual BP, may be applicable to improve the convergence. Example 6.1 showed that mixed-product BP is equivalent to a coordinate descent method on a simple chain model. It is a straightforward exercise to extend this result to general trees, and this may also shed light on how to improve the convergence of mixed-product BP. To this end, it may also be worthwhile to explore connections to the sequential tree reweighted BP algorithm [Kolmogorov, 2006]. On the application side, models and problems with hidden variables or missing information are increasingly common, and marginal MAP provides a general tool for dealing with the uncertainty in these cases; for example, our work in Ping et al. [2014] leveraged mixed product BP to improve structured prediction with hidden variables and partially labelled data. In addition, one view of recently popular "deep networks" is to treat them as graphical models with large numbers of hidden variables; it would be interesting to see if the marginal MAP perspective can be applied in these cases, e.g., by explicitly training (approximate) marginal MAP predictors to minimize the empirical loss function.

Chapter 7

Variational Message Passing For Structured Decision Making

Chapter 6 has demonstrated some of the significant advantages of using variational approaches to derive novel, efficient algorithms for marginal MAP. In this chapter, we adopt a similar framework to solve the even more general and challenging decision making problem for influence diagrams (IDs), based on the variational representation in Theorem 4.3. As in Chapter 6, we develop an array of algorithms, including standard message passing and convergent proximal point versions, and analyze their optimality properties theoretically. We demonstrate that our approach is surprisingly powerful for solving decision making problems, even for decision networks with imperfect recall, in which the variational optimization can be shown to be non-convex in general. In particular, we find that our MEU-BP algorithms are superior to the standard single policy update method, both empirically and theoretically.

The chapter is organized as follows. Section 7.1 reviews some necessary background and sets up notation. We then develop an MEU-BP algorithm in Section 7.2, and then a proximal point algorithm in Section 7.3. Section 7.5 discusses some related work. Finally, we present numerical experiments in Section 7.6 and conclude the chapter in Section 7.7. Proofs and additional details can be found in Appendix C.

7.1 Background

We briefly review some background on decision networks, the MEU task, and its variational representation; see also Section 2.3 and Chapter 4 for more comprehensive introductions.

An influence diagram (ID), also known as a decision network, includes a set of decision variables $\boldsymbol{x}_D$ and random variables $\boldsymbol{x}_R$, on which we define a conditional probability $p(\boldsymbol{x}_R|\boldsymbol{x}_D)$ and a utility function $u(\boldsymbol{x}_R, \boldsymbol{x}_D)$; the product $q_u(\boldsymbol{x}) := p(\boldsymbol{x}_R|\boldsymbol{x}_D)u(\boldsymbol{x}_R, \boldsymbol{x}_D)$ is called the utility-augmented distribution. Each decision variable x_i, $i \in D$ has a parent set $\mathrm{pa}(i)$, which indicates the set of observable variables when making decisions on x_i; therefore, the decision policy on x_i can be written as a conditional table $\delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$. The maximum expected utility (MEU) task is to find an optimal decision strategy $\boldsymbol{\delta} = \{\delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \colon i \in D\}$ to maximize the expectation of the utility functions; letting $\theta(\boldsymbol{x}) = \log q_u(\boldsymbol{x})$, the MEU can be written

$$\mathrm{MEU}(\boldsymbol{\theta}) = \max_{\boldsymbol{\delta} \in \Delta} \sum_{\boldsymbol{x}} \exp(\theta(\boldsymbol{x})) \prod_{i \in D} \delta(x_i|x_{\mathrm{pa}(i)}), \tag{7.1}$$

Theorem 4.3 showed an equivalent variational representation for the MEU task,

$$\log \mathrm{MEU}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau} \in \mathbb{M}} \big\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}; \boldsymbol{\tau}) - \sum_{i \in D} H(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}; \boldsymbol{\tau}) \big\}. \tag{7.2}$$

This is in general a non-convex optimization on $\boldsymbol{\tau}$ for IDs with imperfect recall (LIMIDs), but is guaranteed to be convex for IDs with perfect recall. The subsequent sections focus on developing BP type algorithms by approximating and optimizing this variational form.

For the purpose of developing variational algorithms, we assume $q_u(\boldsymbol{x})$ has a factorized form, i.e., $q_u(\boldsymbol{x}) = \prod_{\alpha \in \mathcal{C}} \psi_\alpha(\boldsymbol{x}_\alpha) = \exp(\sum_{\alpha \in \mathcal{I}} \theta_\alpha(\boldsymbol{x}_\alpha))$, where $\mathcal{C}$ is a set of cliques on $\boldsymbol{x}$. This assumption is naturally satisfied by multiplicative utility functions, but may be problematic for additive utilities and requires additional treatment; we discuss this issue in Section 7.4.

7.2 A Belief Propagation Algorithm for MEU

We start by extending the junction graph framework for representing decision making problems; this is done by introducing special "decision clusters" that represent the decision component. Let $(\mathcal{G}, \mathcal{C}, \mathcal{S})$ be a junction graph for the augmented distribution $q_u(\boldsymbol{x}) \propto \exp(\theta(\boldsymbol{x}))$. For each decision node $i \in D$, we associate it with exactly one cluster $k \in \mathcal{C}$ satisfying $\{i, \mathrm{pa}(i)\} \subseteq c_k$; we call such a cluster a *decision cluster*. The clusters $\mathcal{C}$ are thus partitioned into decision clusters $\mathcal{D}$ and the other (normal) clusters $\mathcal{R} = \mathcal{C} \setminus \mathcal{D}$. We will assume each decision cluster $k \in \mathcal{D}$ is associated with exactly one decision node, denoted by d_k, and hence there is an one-to-one mapping from the decision clusters $\mathcal{D}$ to the decision nodes D; this is easy to ensure during the junction graph construction.

For convenience in deriving message passing type algorithms, we consider the *annealed* version of the MEU variational form

$$\log \mathrm{MEU}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau} \in \mathbb{M}} \big\{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}; \boldsymbol{\tau}) - (1-\epsilon) \sum_{i \in D} H(x_i | \boldsymbol{x}_{\mathrm{pa}(i)}; \boldsymbol{\tau}) \big\}, \tag{7.8}$$

where ϵ is a small positive temperature parameter. Following the junction graph framework in Section 3.2.3, we approximate the annealed variational form (7.8) by

$$\max_{\boldsymbol{\tau} \in \mathbb{L}(\mathcal{G})} \{ \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{k \in \mathcal{R}} H(\boldsymbol{x}_{c_k}; \boldsymbol{\tau}) + \sum_{k \in \mathcal{D}} H^{\epsilon}(\boldsymbol{x}_{c_k}; \boldsymbol{\tau}) - \sum_{(kl) \in \mathcal{E}} H(\boldsymbol{x}_{s_{kl}}; \boldsymbol{\tau}) \}, \tag{7.9}$$

$$\text{where} \quad H^{\epsilon}(\boldsymbol{x}_{c_k}; \boldsymbol{\tau}) = H(\boldsymbol{x}_{c_k}; \boldsymbol{\tau}) - (1-\epsilon) H(x_{d_k} | \boldsymbol{x}_{\mathrm{pa}(d_k)}; \boldsymbol{\tau}), \quad \forall k \in \mathcal{D}.$$

Note that (7.9) is similar to the objective of the regular sum-product junction graph BP in (3.37), except that the regular entropy term $H(\boldsymbol{x}_{c_k}; \boldsymbol{\tau})$ is replaced by $H^{\epsilon}(\boldsymbol{x}_{c_k}; \boldsymbol{\tau})$ on the decision clusters. Using a Lagrange multiplier method similar to Yedidia et al. [2005], one can derive a belief propagation algorithm for maximizing (7.9), which we call MEU-BP, shown

Algorithm 7.1 Belief propagation for MEU (at temperature ϵ)

Input: An influence diagram with a junction graph; an annealing parameter ϵ.
Output: An optimal solution $\boldsymbol{\tau}$ of variational form (7.9).

1. Until convergence, pass messages between clusters on the junction graph:

   Sum-product messages (from normal clusters $k \in \mathcal{R}$):

$$m_{k\to l}(\boldsymbol{x}_{s_{kl}}) \propto \sum_{\boldsymbol{x}_{c_k \setminus s_{kl}}} \psi_{c_k}(\boldsymbol{x}_{c_k}) m_{\sim k\setminus l}(\boldsymbol{x}_{c_k}), \tag{7.3}$$

   MEU-product messages (from decision clusters $k \in \mathcal{D}$):

$$m_{k\to l}(\boldsymbol{x}_{s_{kl}}) \propto \sum_{\boldsymbol{x}_{c_k \setminus s_{kl}}} \psi_{c_k}(\boldsymbol{x}_{c_k}) m_{\sim k\setminus l}(\boldsymbol{x}_{c_k}) \cdot \delta_\epsilon(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(c_k)})^{1-\epsilon}, \tag{7.4}$$

   where $\psi_{c_k}(\boldsymbol{x}_{c_k}) = \exp(\theta_{c_k}(\boldsymbol{x}_{c_k}))$, and $m_{\sim k\setminus l}(\boldsymbol{x}_{c_k}) = \prod_{k'\in\mathcal{N}(k)\setminus\{l\}} m_{k'\to k}(\boldsymbol{x}_{s_{k'k}})$ is the product of messages sent into cluster k not including that from cluster l. The $\delta_\epsilon(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(d_k)})$ is an "annealed" optimal policy defined based on the current messages:

$$\delta_\epsilon(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(d_k)}) = \frac{b_{c_k}(\boldsymbol{x}_{d_k\cup\mathrm{pa}(d_k)})^{1/\epsilon}}{\sum_{x_{d_k}} b_{c_k}(\boldsymbol{x}_{d_k\cup\mathrm{pa}(d_k)})^{1/\epsilon}}, \qquad b_{c_k}(\boldsymbol{x}_{d_k\cup\mathrm{pa}(d_k)}) = \sum_{\boldsymbol{x}_{c_k\setminus\{d_k\cup\mathrm{pa}(d_k)\}}} \psi_{c_k}(\boldsymbol{x}_{c_k}) m_{\sim k}(\boldsymbol{x}_{c_k}),$$

   where $m_{\sim k}(\boldsymbol{x}_{c_k}) = \prod_{k'\in\mathcal{N}(k)} m_{k'\to k}(\boldsymbol{x}_{s_{k'k}})$, the product of all the messages into cluster k.

2. Decode the optimal solution $\boldsymbol{\tau}$:

$$\tau_{c_k}(\boldsymbol{x}_{c_k}) \propto \psi_{c_k}(\boldsymbol{x}_{c_k}) m_{\sim k}(\boldsymbol{x}_{c_k}), \qquad \textit{for normal clusters } (k \in \mathcal{C}); \tag{7.5}$$
$$\tau_{c_k}(\boldsymbol{x}_{c_k}) \propto \psi_{c_k}(\boldsymbol{x}_{c_k}) m_{\sim k}(\boldsymbol{x}_{c_k}) \cdot \delta_\epsilon(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(c_k)})^{1-\epsilon}, \qquad \textit{for decision clusters } (k \in \mathcal{D}); \tag{7.6}$$
$$\tau_{s_{kl}}(\boldsymbol{x}_{s_{kl}}) \propto m_{k\to l}(\boldsymbol{x}_{s_{kl}}) m_{l\to k}(\boldsymbol{x}_{s_{kl}}), \qquad \textit{for all the separators } ((kl) \in \mathcal{E}). \tag{7.7}$$

in Algorithm 7.1 (see Appendix C for the detailed derivation). Algorithm 7.1 includes a temperature parameter ϵ: one can either adopt a deterministic annealing approach [Rose, 1998] by gradually decreasing ϵ, e.g., taking $\epsilon^t = 1/t$ at iteration t, or directly take the limit $\epsilon \to 0^+$, leading to the simpler message updates shown in Algorithm 7.2.

Our MEU-BP algorithm consists of a hybrid of two types of message updates: a regular sum-product message in (7.3), and a special MEU-related message in (7.4). Both message updates admit an intuitive interpretation: the sum-product message (7.3) corresponds to calculating the expected utility given fixed policies by marginalizing over the chance nodes, while the

Algorithm 7.2 Mixed belief propagation for MEU (at zero temperature)

Input: An influence diagram with a junction graph.
Output: An approximately optimal decision policy $\boldsymbol{\delta}^*$.

1. Until convergence, pass messages between clusters on the junction graph:

   Sum messages (*from normal clusters* $k \in \mathcal{C}$):

   $$m_{k\to l}(\boldsymbol{x}_{s_{kl}}) \propto \sum_{\boldsymbol{x}_{c_k \setminus s_{kl}}} \psi_{c_k}(\boldsymbol{x}_{c_k}) m_{\sim k\setminus l}(\boldsymbol{x}_{c_k}),$$

   MEU messages (*from decision clusters* $k \in \mathcal{C}$):

   $$m_{k\to l}(\boldsymbol{x}_{s_{kl}}) \propto \sum_{\boldsymbol{x}_{c_k \setminus s_{kl}}} \psi_{c_k}(\boldsymbol{x}_{c_k}) m_{\sim k\setminus l}(\boldsymbol{x}_{c_k}) \cdot \mathbf{1}[\boldsymbol{x}_{d_k} \in \arg\max b_{c_k}(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(d_k)})],$$

   where $m_{\sim k\setminus l}(\boldsymbol{x}_{c_k}) = \prod_{k'\in\mathcal{N}(k)\setminus\{l\}} m_{k'\to k}(\boldsymbol{x}_{s_{k'k}})$, the product of messages sent into cluster k not including that from cluster l, and $b_{c_k}(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(d_k)})$ is the conditional distribution as per the current local belief $b_{c_k}(\boldsymbol{x}_{c_k}) \propto \psi_{c_k}(\boldsymbol{x}_{c_k}) \prod_{k'\in\mathcal{N}(k)} m_{k'\to k}(\boldsymbol{x}_{s_{k'k}})$.

2. Decode the strategy: $\boldsymbol{\delta}^* = \{\arg\max_{x_{d_k}} b_{c_k}(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(d_k)}) \colon k \in \mathcal{D}\}$.

Note: in case the results of arg max *are not unique, the policies are soft decision rules that chooses the local maxima with equal probabilities.*

MEU-product message (7.4) calculates an ("annealed") optimal policy $\delta_\epsilon(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(c_k)})$ based on the incoming messages, and then passes a message in order to evaluate the resulting expected utility.

Compared to traditional algorithms such as SPU or local search methods, which treat the utility evaluation (marginalization) steps as inner loops of the strategy improving (MEU) steps, our algorithm simultaneously takes steps both on the marginalization and MEU subproblems. This has three important advantages. First, it avoids the potentially wasteful high cost of the inner loops, allowing policies to be optimized as the messages needed to compute expectations are still being propagated. Second, and perhaps more importantly, it becomes trapped at local optima much less often than SPU: the "soft" message updates allow the algorithm to see how changes in one policy may affect others, leading to better local optima in practice (see discussion in Section 7.3 and experiments in Section 7.6). Finally, the explicit

message-passing form of BP methods is well-suited to creating distributed implementations, empowering large-scale computation.

7.2.1 Reparameterization Properties and Optimality Certificates

Both sum-product BP and max-product BP can usefully be interpreted as reparameterization operators, with fixed points satisfying some consistency property yet leaving the joint distribution unchanged [e.g., Wainwright et al., 2003a, Weiss et al., 2007, Liu and Ihler, 2013]. In this section, we study the reparameterization properties of our MEU-BP, based on which some optimality guarantees will be presented.

We start by defining a set of beliefs $\boldsymbol{b} = \{b_{c_k}(\boldsymbol{x}_{c_k}), b_{s_{kl}}(\boldsymbol{x}_{s_{kl}}) \colon k \in \mathcal{C}, (kl) \in \mathcal{S}\}$ as functions of the messages (in the same way as regular sum-product BP),

$$\begin{aligned} b_{c_k}(\boldsymbol{x}_{c_k}) &\propto \psi_{c_k}(\boldsymbol{x}_{c_k}) \prod_{l \in \mathcal{N}(k)} m_{l \to k}(\boldsymbol{x}_{s_{kl}}), \quad \text{for all clusters } k \in \mathcal{C}, \\ b_{s_{kl}}(\boldsymbol{x}_{s_{kl}}) &\propto m_{k \to l}(\boldsymbol{x}_{s_{kl}}) m_{l \to k}(\boldsymbol{x}_{s_{kl}}), \qquad \text{for all separators } (kl) \in \mathcal{E}, \end{aligned} \tag{7.10}$$

Note that the "beliefs" $\boldsymbol{b}$ are different from the "marginals" $\boldsymbol{\tau}$ defined in (7.5)-(7.7).

Lemma 7.1. *(1). At each iteration of MEU-BP in Algorithm 7.1, the MEU-beliefs* $\boldsymbol{b}$ *as defined in* (7.10) *satisfy*

$$\textbf{Reparameterization:} \qquad q_u(\boldsymbol{x}) \propto \frac{\prod_{k \in \mathcal{V}} b_{c_k}(\boldsymbol{x}_{c_k})}{\prod_{(kl) \in \mathcal{E}} b_{s_{kl}}(\boldsymbol{x}_{s_{kl}})}, \tag{7.11}$$

where $q_u(\boldsymbol{x})$ *is the augmented distribution of the influence diagram.*

(2). At any fixed point of the MEU-BP in Algorithm 7.1, we have

Sum-consistency :

$$b_{s_{kl}}(\boldsymbol{x}_{s_{kl}}) = \sum_{\boldsymbol{x}_{c_k \setminus s_{kl}}} b_{c_k}(\boldsymbol{x}_{c_k}), \quad \forall \text{ normal clusters } k \in \mathcal{R}, \tag{7.12}$$

MEU-consistency :

$$b_{s_{kl}}(\boldsymbol{x}_{s_{kl}}) = \sum_{\boldsymbol{x}_{c_k \setminus s_{kl}}} b_{c_k}(\boldsymbol{x}_{c_k}) \delta_\epsilon(x_{d_k} | \boldsymbol{x}_{\mathrm{pa}(d_k)})^{1-\epsilon}, \quad \forall \text{ decision clusters } k \in \mathcal{D}. \tag{7.13}$$

Proof. (1). By simple algebraic substitution,

$$\frac{\prod_{k \in \mathcal{V}} b(x_{c_k})}{\prod_{(kl) \in \mathcal{E}} b(x_{s_{kl}})} \propto \frac{\prod_{k \in \mathcal{V}} \psi_{c_k}(\boldsymbol{x}_{c_k}) \prod_{l \in \mathcal{N}(k)} m_{l \to k}(\boldsymbol{x}_{s_{kl}})}{\prod_{(kl) \in \mathcal{E}} m_{k \to l}(\boldsymbol{x}_{s_{kl}}) m_{l \to k}(\boldsymbol{x}_{s_{kl}})} = \prod_{c_k \in \mathcal{C}} \psi_{c_k}(x_{c_k}) \propto q_u(\boldsymbol{x}).$$

(2). Simply substitute the definition (7.10) of $\boldsymbol{b}$ into the message updates in (7.3)-(7.4). □

The reparameterization property can be used to derive optimality certificates for MEU-BP. To this end, we need some notation. A partial ordering on the cluster nodes $\mathcal{V}$ is called a *tree-order* of the junction tree if we have that $k \preceq l$ iff the unique path from a special cluster (called root) to l passes through k. For any cluster $k \in \mathcal{V}$, its unique neighbor on the path to the root is called its parent, denoted by $\pi(k)$. Given a subset of decision nodes $D' \subseteq D$ in the influence diagram, a junction tree is said to be *consistent* on D' if there exists a tree-order satisfying $s_{k,\pi(k)} \subseteq \mathrm{pa}(d_k)$ for any $d_k \in D'$, where d_k is the decision node associated with (decision) cluster k.

Theorem 7.1. *Let $(\mathcal{G}, \mathcal{C}, \mathcal{S})$ be a junction tree consistent on a subset of decision nodes D'. If Algorithm 7.2 (MEU-BP with $\epsilon \to 0^+$) converges, then the decoded $\boldsymbol{\delta}^*$ is a locally optimal strategy in the sense that $\mathrm{EU}(\{\boldsymbol{\delta}_{D'}, \boldsymbol{\delta}^*_{D \setminus D'}\}) \leq \mathrm{EU}(\boldsymbol{\delta}^*)$ for any $\boldsymbol{\delta}_{D'}$.*

Proof. See the appendix. □

A junction tree is said to be *globally* consistent if it is consistent for the whole set D of decision nodes, in which case Theorem 7.1 ensures a globally optimal strategy; this notion of *global consistency* is similar to *strong junction trees* in Jensen et al. [1994]. For IDs with perfect recall, a globally consistent junction tree can be constructed by a standard procedure which triangulates the DAG of the ID along reverse temporal order. For IDs without perfect recall, it is usually not possible to construct a globally consistent junction tree.

However, the following theorem shows that for general IDs with arbitrary junction trees, Theorem 7.1 implies regular policy-by-policy optimality, indicating that MEU-BP is at least as "optimal" as SPU.

Theorem 7.2. *Let $(\mathcal{G}, \mathcal{C}, \mathcal{S})$ be an arbitrary junction tree, and $\boldsymbol{\delta}^*$ the strategy given by MEU-BP in Algorithm 7.2 at its convergence. Then $\boldsymbol{\delta}^*$ is a policy-by-policy optimal strategy in the sense that $\mathrm{EU}(\{\boldsymbol{\delta}_i, \boldsymbol{\delta}^*_{D\setminus i}\}) \leq \mathrm{EU}(\boldsymbol{\delta}^*)$ for any $i \in D$ and $\boldsymbol{\delta}_i$.*

Proof. This follows since any junction tree is consistent for any single decision node $i \in D$, by taking a tree-ordering rooted at i's decision cluster. □

7.3 Proximal Point Algorithms for MEU

One disadvantage of MEU-BP is that it has no guarantees on convergence. In this section, we present a class of convergent algorithms based on the proximal point approach [e.g., Martinet, 1970, Rockafellar, 1976]. This section is an direct generalization of the proximal point techniques for marginal MAP in Section 6.4.

To review briefly, the proximal point algorithm solves an optimization $\min_{\boldsymbol{\tau} \in \mathbb{M}} f(\boldsymbol{\tau})$, where $f(\boldsymbol{\tau})$ is our variational objective function, by iteratively solving a sequence of "proximal"

Algorithm 7.3 Proximal point algorithm for MEU

Input: An influence diagram.
Output: An optimal decision policy $\boldsymbol{\delta}^*$.
Define the proximal coefficients $\{\lambda^t\}$, e.g., $w^t = 1$, or $\lambda^t = 1/t$.
Initialize the local marginals $\boldsymbol{\tau}^0$.
for iteration t **do**

$$\theta^t(\boldsymbol{x}) = \theta(\boldsymbol{x}) + \lambda^t \sum_{i \in D} \log \tau^t(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}), \tag{7.15}$$

$$\boldsymbol{\tau}^{t+1} = \arg\max_{\boldsymbol{\tau} \in \mathbb{M}} \{\langle \boldsymbol{\theta}^t, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}\ ;\ \boldsymbol{\tau}) - (1-\lambda^t) H(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}\ ;\ \boldsymbol{\tau})\}, \tag{7.16}$$

end for
Decode the strategy: $\boldsymbol{\delta}^* = \{\tau^t(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \colon i \in D\}$.

problems whose solutions converge to the solution of the target problem,

$$\boldsymbol{\tau}^{t+1} = \arg\min_{\boldsymbol{\tau} \in \mathbb{M}} \{f(\boldsymbol{\tau}) + \lambda^t D(\boldsymbol{\tau}||\boldsymbol{\tau}^t)\}, \tag{7.14}$$

where $\boldsymbol{\tau}^t$ is the solution at iteration t and λ^t is a positive number called the proximal coefficient; $D(\cdot||\cdot)$ is the proximal function. See Section 6.4 for more details on the proximal point method.

As in Chapter 6, we use an entropic proximal function that naturally fits the MEU problem:

$$D(\boldsymbol{\tau}||\boldsymbol{\tau}') = \sum_{i \in D} \sum_{\boldsymbol{x}} \tau(\boldsymbol{x}) \log \left[\frac{\tau(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})}{\tau'(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})} \right],$$

which is a sum of conditional KL-divergences between the policies over the decision nodes. Algorithm 7.3 shows the corresponding proximal point algorithm when applied to the MEU dual (7.2). Note that the inner proximal update (7.16) has the same form as the annealed approximation in (7.9), and hence can be readily solved by MEU-BP in Algorithm 7.1. However, unlike the annealing scheme, the proximal algorithm updates use different natural parameter values $\boldsymbol{\theta}^t$ at each iteration, and it does not require λ^t to approach zero.

We use two choices of proximal coefficients $\{\lambda^t\}$: (1) $w^t = 1$ (*constant*), and (2) $\lambda^t = 1/t$

(*harmonic*). The choice $\lambda^t = 1$ is especially interesting because the proximal update reduces to a standard *marginalization* dual form $\boldsymbol{\tau}^{t+1} = \arg\max_{\boldsymbol{\tau}\in\mathbb{M}}\{\langle\boldsymbol{\theta}^t, \boldsymbol{\tau}\rangle + H(\boldsymbol{x};\boldsymbol{\tau})\}$, with solution $\tau^{t+1}(\boldsymbol{x}) \propto \exp(\theta^{t+1})$; one can simply perform marginalization on $\boldsymbol{\tau}(\boldsymbol{x})$ to get its marginals $\{\tau(x_i, x_{\mathrm{pa}(i)})\}$ by standard tools such as sum-product BP or bucket elimination, without considering the MEU's temporal elimination order restrictions. Concretely, one can show that the proximal update in this case reduces to

$$\tau_i^{t+1}(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \propto \tau_i^t(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})\mathbb{E}(u(\boldsymbol{x})|\boldsymbol{x}_{\{i\}\cup\mathrm{pa}(i)}\,;\,\boldsymbol{\delta}_{\neg i}^n),$$

with $\mathbb{E}(u(\boldsymbol{x})|\boldsymbol{x}_{\{i\}\cup\mathrm{pa}(i)}\,;\,\boldsymbol{\delta}_{\neg i}^n)$ as defined in (2.5). This proximal update has an elegant intuition as a "soft" and "parallel" version of the greedy SPU update (2.5). However, it is worth noting that convergence with $\lambda^t = 1$ may be slow; using $\lambda^t = 1/t$ takes larger steps but the proximal update (7.16) is no longer a standard marginalization.

Remark. We can gain some intuition about the advantages of our approach by considering the difference between the proximal update with $\lambda^t = 1$ and the greedy SPU update. SPU updates a single decision node, selecting a new, deterministic policy (a "hard threshold") to optimize the expected utility. The proximal update, on the other hand, updates all policies simultaneously; multiplying by the expectation (instead of selecting its largest entry) has the effect of moving the policies towards, but not all the way to, what that expectation currently indicates as their best deterministic policy. Intuitively, the soft update makes it possible for the effects of the updated policies to be measured at the other decision nodes, before becoming "locked in", allowing them to correct for early mistakes and make cooperative movements. The hard update, on the other hand, can quickly become stuck at local optima; this is analogous to the behavior of EM discussed in Section 6.5, which ensures deterministic solutions at the risk of becoming stuck easily.

7.4 Dealing with Additively Decomposable Utilities

The algorithms we described require the augmented distribution $q_u(\boldsymbol{x})$ to be multiplicatively factored, or have low (constrained) tree-width. However, this can easily fail to be the case if using a direct representation of additively decomposable utilities. To explain, recall that the augmented distribution is of the form $q_u(\boldsymbol{x}) \propto p(\boldsymbol{x}_R|\boldsymbol{x}_D)u(\boldsymbol{x})$, where $p(\boldsymbol{x}_R|\boldsymbol{x}_D) = \prod_{i\in C} p(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$ is the probability of the chance nodes, and $u(\boldsymbol{x}) = \sum_{j\in U} u_j(\boldsymbol{x}_{\beta_j})$ is the additive utility. In this case, the utility $u(\boldsymbol{x})$ creates a large factor with variable scope $\cup_j \beta_j$, and can easily destroy the multiplicative factorization structure of $q_u(\boldsymbol{x})$. Unfortunately, the naïve method of calculating the expectation node by node, or the commonly used general variable elimination procedures [e.g., Jensen et al., 1994] do not appear suitable for our variational framework. To address this problem, we introduce an artificial multiplicative structure into the utility function by augmenting the model with a latent "selector" variable, similar to that used for the "complete likelihood" in mixture models. Let y_0 be an auxiliary random variable taking values in the utility index set U, so that the joint (unormalized) distribution of $\{\boldsymbol{x}, y_0\}$ is

$$\tilde{q}_u(\boldsymbol{x}, y_0) = p(\boldsymbol{x}_R|\boldsymbol{x}_D) \prod_j \tilde{u}_j(\boldsymbol{x}_{\beta_j}, y_0), \tag{7.17}$$

$$\text{where} \qquad \tilde{u}_j(\boldsymbol{x}_{\beta_j}, y_0) = \begin{cases} u_j(\boldsymbol{x}_{\beta_j}), & \text{if } y_0 = j, \\ 1, & \text{if } y_0 \neq j. \end{cases}$$

It is easy to verify that the marginal distribution of $\tilde{q}_u(\boldsymbol{x}, y_0)$ on $\boldsymbol{x}$ is $q_u(\boldsymbol{x})$, that is, $q_u(\boldsymbol{x}) = \sum_{y_0\in U} \tilde{q}_u(\boldsymbol{x}, y_0)$. The tree-width of $\tilde{q}_u(\boldsymbol{x}, y_0)$ is no larger than one plus the tree-width of the DAG (with utility nodes included) of the ID, which is typically much smaller than that of $q_u(\boldsymbol{x})$ when the additive utility $u(\boldsymbol{x})$ is included directly. A derivation similar to that in Theorem 4.3 shows that we can replace $\theta(\boldsymbol{x}) = \log q_u(\boldsymbol{x})$ with $\tilde{\theta}(\boldsymbol{x}) = \log \tilde{q}_u(\boldsymbol{x})$ in (7.2)

without changing the results:

Corollary 7.1. *Let* $\tilde{q}_u(\boldsymbol{x}, y_0)$ *be defined in* (7.17)*, and* $\tilde{\theta}(\boldsymbol{x}, y_0) = \log \tilde{q}_u(\boldsymbol{x}, y_0)$ *its natural parameter. Then for the log MEU of the original ID, we have*

$$\log \mathrm{MEU} = \max_{\boldsymbol{\tau} \in \tilde{\mathbb{M}}} \{\langle \tilde{\boldsymbol{\theta}}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}; \boldsymbol{\tau}) - \sum_{i \in D} H(x_i | \boldsymbol{x}_{\mathrm{pa}(i)}; \boldsymbol{\tau})\},$$

where $\tilde{\mathbb{M}}$ *is the marginal polytope over* $\boldsymbol{x} \cup y_0$*. If* $\boldsymbol{\tau}^*$ *is an optimal solution, then*

$$\boldsymbol{\delta}^* = \{\boldsymbol{\tau}^*(x_i | \boldsymbol{x}_{\mathrm{pa}(i)}) \colon i \in D\}$$

is an optimal strategy of the original ID.

Proof. We can directly apply Theorem 4.3, but with the utility-augmented distribution $p(\boldsymbol{x}_R | \boldsymbol{x}_D) u(\boldsymbol{x})$ replaced by $\tilde{q}_u(\boldsymbol{x}, y_0)$. □

Therefore, we can apply the algorithms we developed on $\tilde{q}_u(\boldsymbol{x}, y_0)$, instead of $q_u(\boldsymbol{x})$. The complexity of this method may be further improved by exploiting the context-specific independence of $\tilde{q}_u(\boldsymbol{x}, y_0)$, i.e., that $\tilde{q}_u(\boldsymbol{x} | y_0)$ has a different dependency structure for different values of y_0, but this is left for future work.

7.5 Related Work

There exists a large body of work on solving influence diagrams, mostly considering exact algorithms for IDs with perfect recall; see Koller and Friedman [2009] for a recent review. The exact version of our algorithm in the case of perfect recall is most closely connected to the early work of Jensen et al. [1994], who compile an ID into a junction tree structure on which a special message passing algorithm is performed; their notion of *strong junction trees* is related

to the notion of *global consistency* we develop in Section 7.2.1. However, their framework requires the perfect recall assumption and it is unclear how to extend it to approximate inference. A somewhat different approach transforms the decision problem into a sequence of standard Bayesian network inference problems [e.g., Cooper, 1988, Shachter and Peot, 1992, Zhang, 1998], where each subroutine is a standard inference problem, and can be solved using standard inference algorithms, either exactly or approximately; again, however, their method only works within the perfect recall assumption. Several other approximation algorithms are also based on separately approximating individual components of exact algorithms, e.g., Sabbadin et al. [2011] and Sallans [2003] approximate policy update methods by mean field methods; Nath and Domingos [2010] use adaptive belief propagation to approximate the inner loop of greedy search algorithms. Dechter [2000a,b] apply bucket elimination and mini-bucket approximation algorithms to IDs with perfect recall. Watthayu [2008] proposes a loopy BP algorithm, but without theoretical justification. To the best of our knoweldge, our method is the first to "directly" approximate both the evaluation of the expectations and the policy optimization in a single, joint framework.

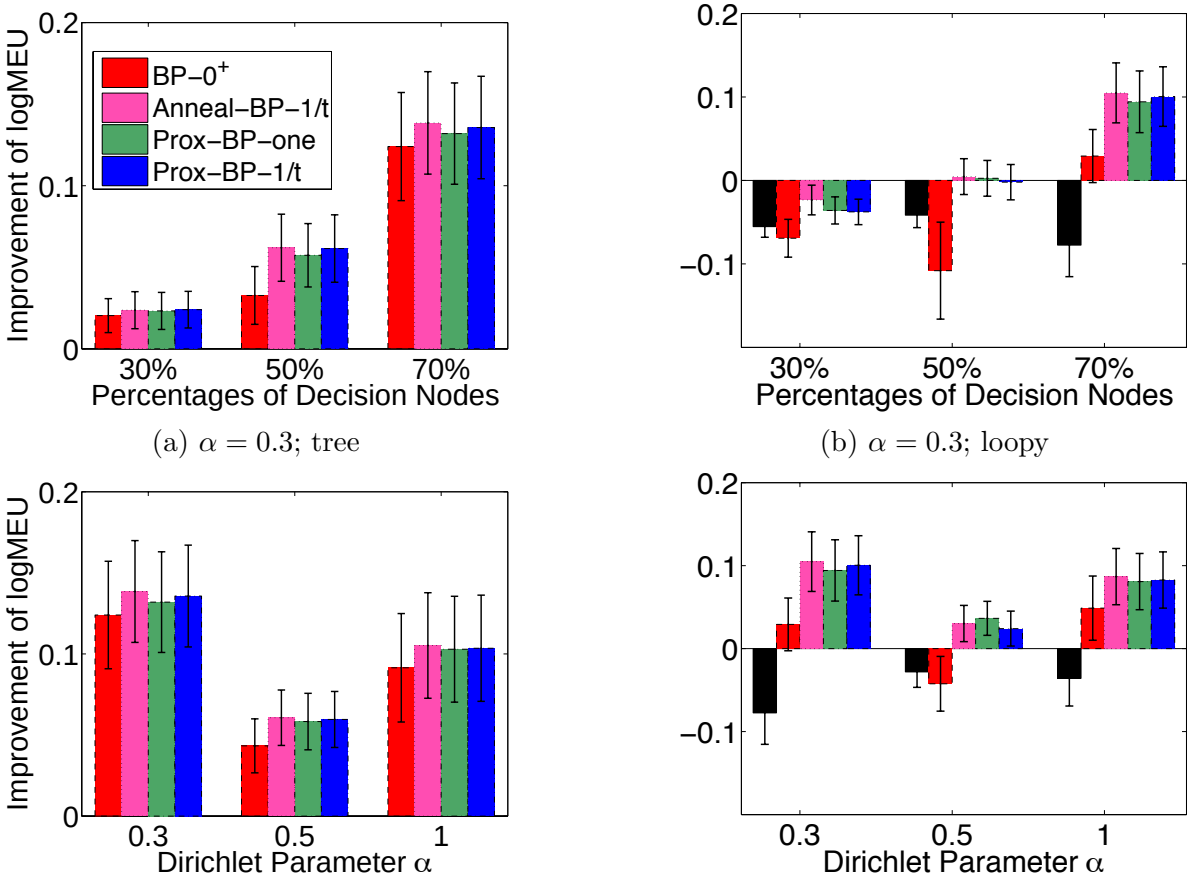
For IDs without perfect recall (LIMID), backward induction based methods do not apply. Most existing algorithms work by optimizing the decision rules node-by-node or group-by-group [e.g., Lauritzen and Nilsson, 2001, Madsen and Nilsson, 2001, Koller and Milch, 2003]; these methods reduce to exact backward induction (hence guaranteeing global optimality) when applied to IDs with perfect recall and updated backwards along the temporal ordering. However, they only guarantee local person-by-person optimality (or Nash equilibrium if treating the team decision problem as a cooperative game) for general LIMIDs, which can be weaker than the optimality guaranteed by our BP-like methods. Other styles of approaches, such as Monte Carlo methods [e.g., Bielza et al., 1999, Cano et al., 2006, Charnes and Shenoy, 2004, Garcia-Sanchez and Druzdzel, 2004] and search-based methods [e.g., Luque et al., 2008, Qi and Poole, 1995, Yuan and Wu, 2010, Marinescu, 2010] have also been proposed. Recently,

Maua and de Campos [2011] proposed a method for finding the globally optimal strategies of LIMIDs by iteratively pruning non-maximal policies. Finally, some variational inference ideas have been applied to the related problems of reinforcement learning or solving Markov decision processes [e.g., Sallans and Hinton, 2001, Furmston and Barber, 2010, Yoshimoto and Ishii, 2004].

7.6 Experiments

We demonstrate our algorithms on several influence diagrams, including randomly generated IDs, large scale IDs constructed from problems in the UAI08 inference challenge, and finally practically motivated IDs for decentralized detection in wireless sensor networks. We find that our algorithms typically find better solutions than SPU with comparable or better time complexity; for large scale problems with many decision nodes, our algorithms are usually more computationally efficient than SPU because one step of SPU requires re-computing (2.5) (a global expectation) for each decision node's update.

In all experiments, we test single policy updating (`SPU`), our MEU-BP running directly at zero temperature (`BP-0`$^+$), annealed BP with temperature $\epsilon^t = 1/t$ (`Anneal-BP-1/t`), and the proximal versions with $\lambda^t = 1$ (`Prox-BP-one`) and $\lambda^t = 1/t$ (`Prox-BP-1/t`). For the BP-based algorithms, we use two constructions of junction graphs: a standard junction tree by triangulating the DAG in backwards topological order, and a loopy junction graph following [Mateescu et al., 2010] that corresponds to Pearl's loopy BP; for SPU, we use the same junction graphs to calculate the inner update (2.5). The junction trees ensure the inner updates of `SPU` and `Prox-BP-one` are performed exactly, and have the optimality guarantees given in Theorem 7.1, but may be computationally more expensive than using loopy junction graphs with smaller cliques. For the proximal versions, we set a maximum of 5 iterations in the inner loop; changing this value did not seem to lead to significantly different results.



(a) $\alpha = 0.3$; tree

(b) $\alpha = 0.3$; loopy

(c) 70% decision nodes; tree

(d) 70% decision nodes; loopy

Figure 7.1: Results on random IDs of size 20. The y-axes show the log MEU of each algorithm compared to SPU on a junction tree. The left panels correspond to running the algorithms on junction trees, and right panels on loopy junction graphs. (a) & (b) show MEUs as the percentage of decision nodes changes. (c) & (d) show MEUs vs. the Dirichlet parameter α. The results are averaged on 20 random models.

The BP-based algorithms may return non-deterministic strategies; we round to deterministic strategies by taking the largest values.

7.6.1 Random Bayesian Networks

We test our algorithms on randomly constructed IDs with additive utilities. We first generate a set of random DAGs of size $n = 20$ with maximum parent size of $\omega = 3$, by sequentially assigning each node a randomly chosen parent set of size ω', with ω' drawn uniformly in $[1 : \omega]$, along a predefined node ordering. To create IDs, we take the leaf nodes to be utility

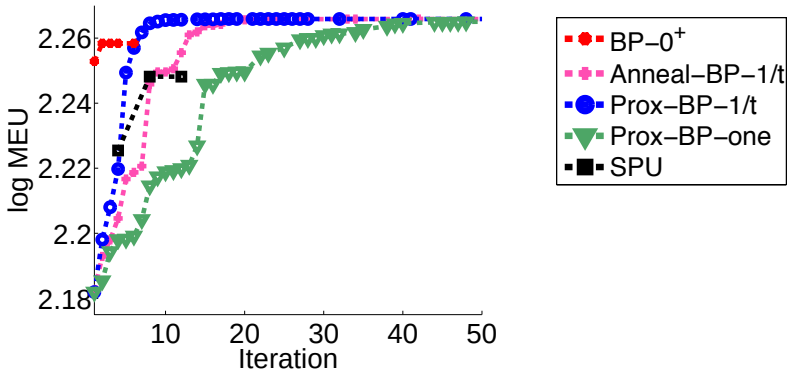
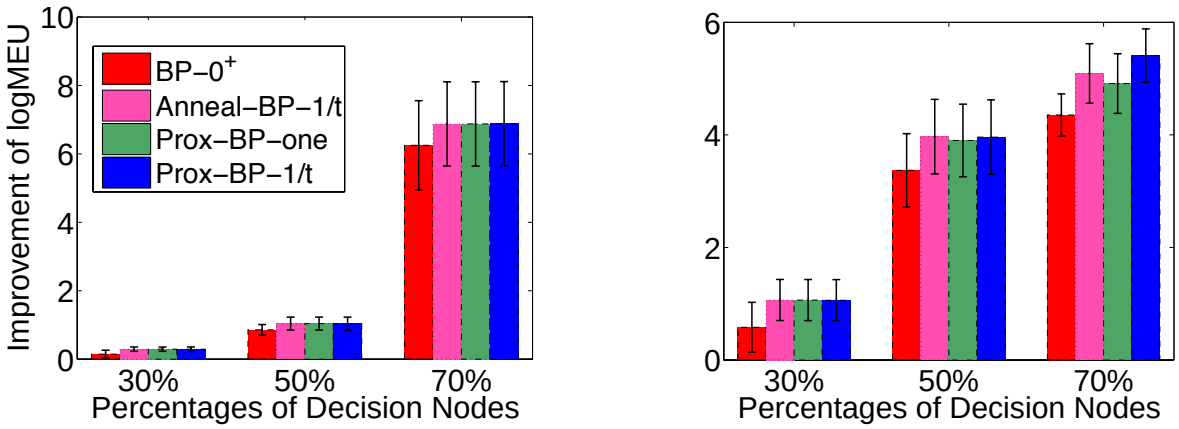


Figure 7.2: A typical trajectory of MEU (of the rounded deterministic strategies) vs. iterations for the random IDs in Figure 7.1. One iteration of the BP-like methods denotes a forward-backward induction on the junction graph; one step of SPU requires $|D|$ (number of decision nodes) inductions. `SPU` and `BP-0`$^+$ are stuck at a local mode after the 2nd iteration.

nodes, and among non-leaf nodes we randomly select a fixed percentage to be decision nodes, with the others being chance nodes. We assume the chance and decision variables are discrete with 4 states. The conditional probability tables of the chance nodes are randomly drawn from a symmetric Dirichlet distribution $\mathrm{Dir}(\alpha)$, and the entries of the utility function from Gamma distribution $\Gamma(\alpha, 1)$.

The relative improvement in log MEU compared to `SPU` with a junction tree are reported in Figure 7.1. We find that when using junction trees, all our BP-based methods dominate `SPU`; for loopy junction graphs, `BP-0`$^+$ occasionally performs worse than `SPU`, but all the annealed and proximal algorithms outperform `SPU` with the same loopy junction graph, and often even `SPU` with a junction tree. As the percentage of decision nodes increases, the improvement of the BP-based methods on `SPU` generally increases. Figure 7.2 shows a typical trajectory of the algorithms across iterations, in which we can see that both `SPU` and `BP-0`$^+$ become stuck at a local optimum after the second iteration, while the annealed and proximal algorithms continue to improve, providing better solutions at convergence. Here the algorithms were initialized uniformly; random initializations behaved similarly and have been omitted.



(a). Diagnostic BN 1 (b). Diagnostic BN 2

Figure 7.3: Results on IDs constructed from two diagnostic Bayes nets (BNs) from the UAI08 challenge. Here all the algorithms used the loopy junction graph and are initialized uniformly. (a)-(b) the logMEU of algorithms normalized to that of SPU. Averaged on 10 random trails.

7.6.2 Diagnostic Bayesian Networks

We construct larger scale IDs based on two diagnostic Bayes nets taken from the UAI08 inference challenge, with 200-300 nodes and 300-600 edges, respectively. To create influence diagrams, we took the leaf nodes to be utility nodes, and define the utility functions by the conditional probabilities when clamped to a randomly chosen state, and total utility as the product of the local utility functions (multiplicatively decomposable). The set of decision nodes is again randomly selected among the non-leaf nodes with a fixed percentage. Since the network sizes are large, we only run the algorithms on the loopy junction graphs. Again, our algorithms significantly improve on SPU; see Figure 7.3.

7.6.3 Decentralized Sensor Network

In this section, we test an influence diagram constructed for decentralized detection in wireless sensor networks [e.g., Viswanathan and Varshney, 1997, Kreidl and Willsky, 2006]. The task is to detect the states of a hidden process $p(h)$ (specified as a pairwise MRF) using a set of distributed sensors; each sensor provides a noisy measurement v_i of the local state h_i,

and overall performance is boosted by allowing the sensor to transmit small (1-bit) signals s_i along an directional path, to help the predictions of their downstream neighboring sensors. The utility function includes rewards for correct prediction and a cost for sending signals. This MEU problem here is a highly challenging team decision problem, which requires significant cooperation between the sensors: for example, a sensor should only send a signal to another sensor if the receiver can decode and leverage the signal properly. We construct an ID as sketched in Figure 7.5(a) for addressing the *offline* policy design task of finding optimal policies for how to predict the states based on the local measurement and received signals, and policies for whether and how to pass signals to downstream nodes.

Setting. In detail, we assume that the distribution $p(h)$ of the hidden variables is an attractive pairwise MRF on a graph $G_h = (V_h, E_h)$,

$$p(h) = \frac{1}{Z} \exp\Big[\sum_{(ij)\in G_h} \theta_{ij}(h_i, h_j)\Big], \tag{7.18}$$

where the h_i are discrete variables with p_h states (we take $p_h = 5$); we set $\theta_{ij}(k,k) = 0$ and randomly draw $\theta_{ij}(k,l)$ $(k \neq l)$ from the negative absolute values of a standard Gaussian variable $\mathcal{N}(0,1)$. Each sensor gives a noisy measurement v_i of the local variable h_i with probability of error e_i, that is, $p(v_i|h_i) = 1 - e_i$ for $v_i = h_i$ and $p(v_i|h_i) = e_i/(p_h - 1)$ (uniformly) for $v_i \neq h_i$.

Let G_s be a DAG that defines the path on which the sensors are allowed to broadcast signals; all the downstream sensors receive the same signal, and we assume the channels are noise free. Each sensor is associated with two decision variables: $s_i \in \{0, \pm 1\}$ represents the signal sent from sensor i, where ± 1 represents a one-bit signal with cost η and 0 represents "off" with no cost; and d_i represents the prediction of h_i based on v_i and the signals $s_{\mathrm{pa}_s(i)}$ received from i's upperstream sensors; a correct prediction $(d_i = h_i)$ yields a reward γ (we set $\gamma = \ln 2$). Hence, two types of utility functions are involved, the signal cost utilities u_η, with

$u_\eta(s_i = \pm 1) = -\eta$ and $u_\eta(s_i = 0) = 0$; the prediction reward utilities u_γ with $u_\gamma(d_i, h_i) = \gamma$ if $d_i = h_i$ and $u_\gamma(d_i, h_i) = 0$ otherwise. The total utility function is constructed multiplicatively via $u = \exp[\sum_i u_\gamma(d_i, h_i) + u_\eta(s_i)]$.

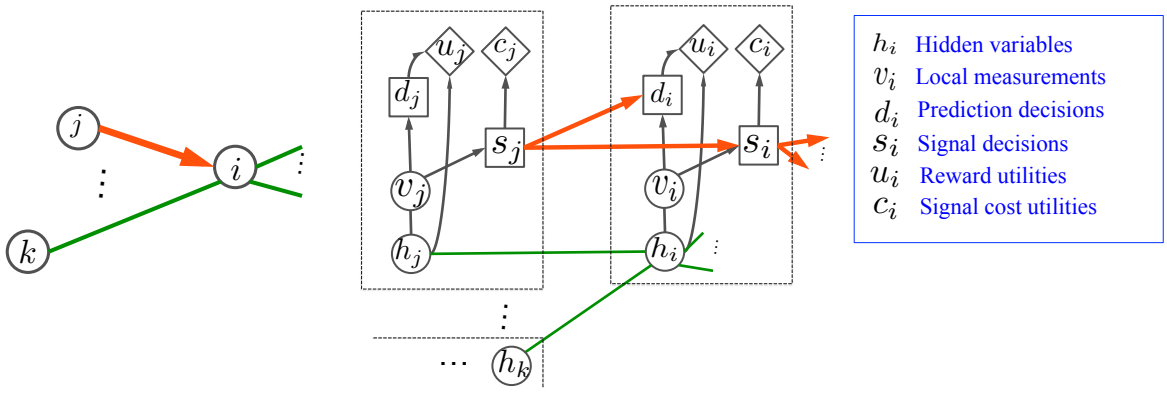
We also create two qualities of sensors: "good" sensors for whose error rate e_i is drawn from $\mathcal{U}([0, .1])$ and "bad" sensors ($e_i \sim \mathcal{U}([.7, .8])$), where $\mathcal{U}$ is the uniform distribution. Generally speaking, the optimal strategies should pass signals from the good sensors to bad sensors to improve their predictive power. See Figure 7.4 for the actual influence diagram.

Note that the definition of the ID here is not a standard one, since $p(h)$ is not specified as a Bayesian network; one could convert $p(h)$ to an equivalent Bayesian network by the standard triangulation procedure, with some loss in representational efficiency. The normalization constant Z in (7.18) only changes the expected utility function by a constant and so does not need to be calculated for the purpose of the MEU task.

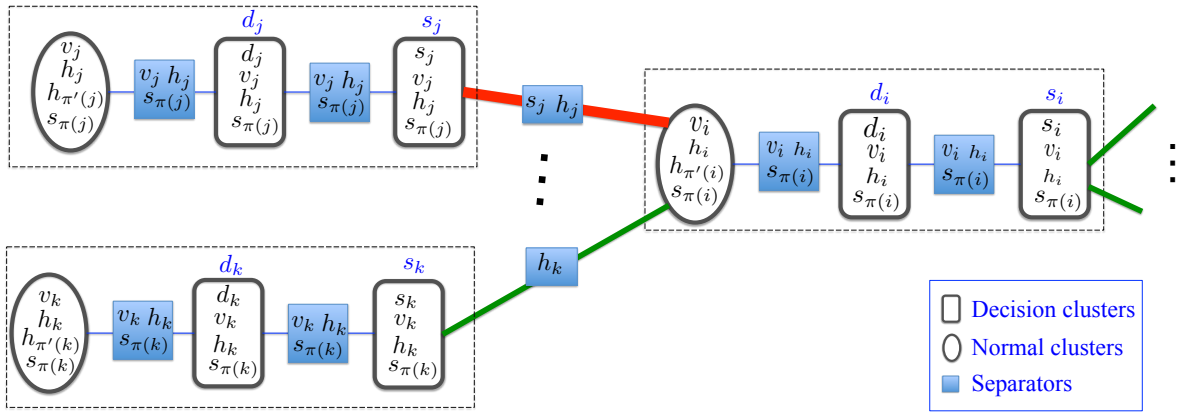
Specifying Junction Graphs. For our experiments, when computationally feasible we use an exact junction tree constructed by standard triangulation, backwards along ordering o,

$$o = [h_1, v_1, d_1, s_1 \ ; \ \ldots \ ; \ h_{n_h}, v_{n_h}, d_{n_h}, s_{n_h}],$$

which is consistent with the signal path G_h. We also use a loopy junction graph, which can be made computationally tractable by definition. A fully automated construction of a loopy junction graph is non-trivial; we show in Figure 7.4(c) the one we use in our experiments. It is constructed such that the decision structure inside each sensor node is preserved exactly, while at a higher level (among sensors), a standard loopy junction graph similar to that introduced in Mateescu et al. [2010] (corresponding to Pearl's loopy BP) captures the correlation between the sensor nodes. One can shown that this junction graph reduces to a junction tree when the MRF G_h is a tree and the signal path G_s is an oriented tree.



(a) A sensor network node (b) The influence diagram for sensor network node in (a)



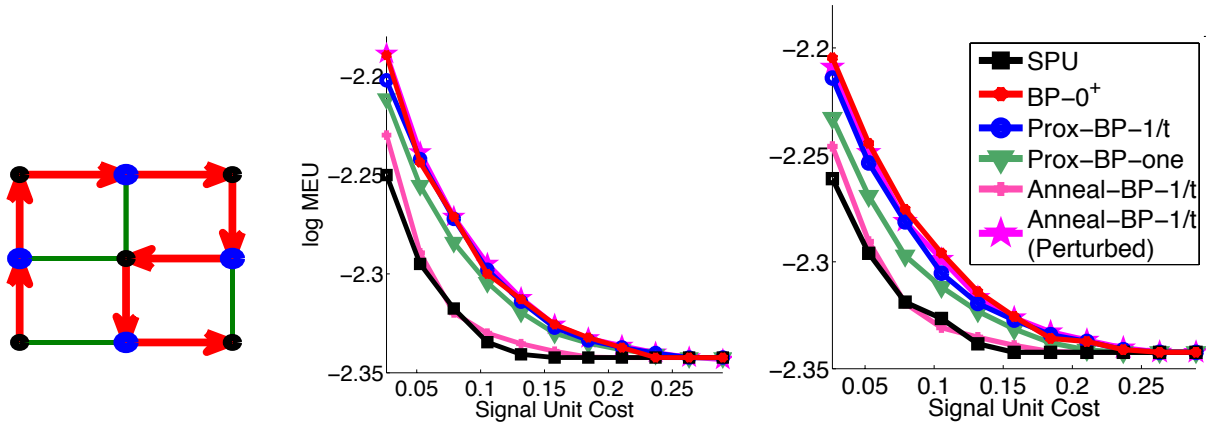
(c) A loopy junction graph for the ID in (b)

Figure 7.4: (a) A node of the sensor network structure: the nodes i, j, k represent the sensors, and the green lines denote the MRF edges, on some of which (red arrows) signals are allowed to pass. (b) The influence diagram constructed for the sensor network in (a). (c) A junction graph for the ID in (b); $\pi(i)$ denotes the parent set of sensor i in terms of the signal path G_s, and $\pi'(i)$ denotes the parent set in terms of the hidden process $p(h)$ (when $p(h)$ is transformed into a Bayesian network by triangulating reversely along order o). The decision clusters (black rectangles) are labeled with their corresponding decision variables on their top.

Results. We first test on a sensor network defined on a small, 3×3 grid, where the algorithms are run on both the exact junction tree and the loopy junction graph in Figure 7.4. To escape the "all-zero" fixed point that sends no signals at all, we initialize the proximal algorithms and SPU with 5 random policies, and `BP-0`$^+$ and `Anneal-BP-1/t` with 5 random messages. Figure 7.5(b)-(c) show the performance of the different algorithms as the signal unit cost is varied; we observe two different trends as the signal unit cost changes:

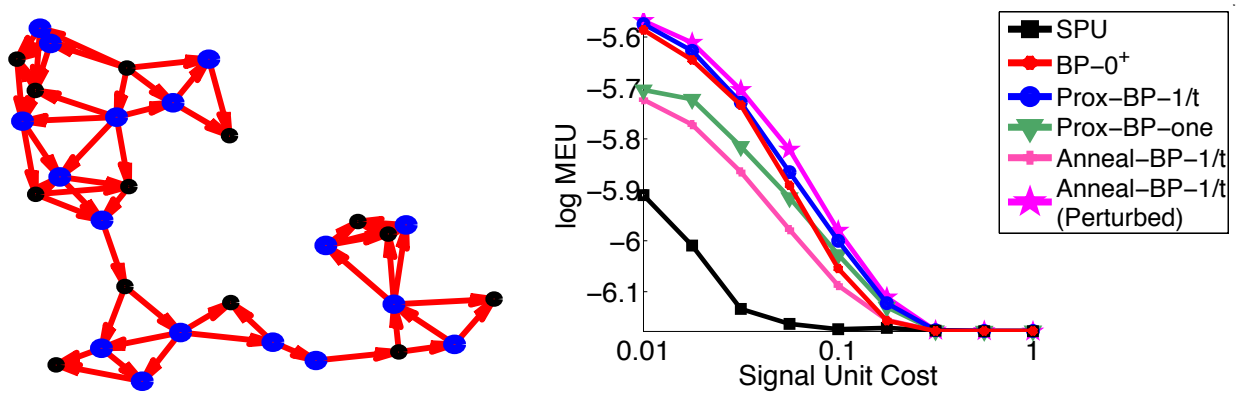
1. When the signal cost is high (e.g., > 0.2), the optimal strategy is to send no signals and simply make independent local decisions; the problem is relatively easy in this case, and all the algorithms returns the same communication-free strategy and perform the same.
2. The much more interesting region is when the signal cost is low (e.g., < 0.2), and communication becomes beneficial. In this case, the optimal strategy should cause the sensors to send the "right" number of signals, which depends on their own confidence and their neighbors' needs, and to correctly decode and combine the signals from the other sensors. This makes the problem a very challenging collaborative decision problem, and the various algorithms tend to behave very differently. In particular, the two proximal BPs and `BP-0`$^+$ perform significantly better than the other algorithms, indicating that they are able to find better strategies and enable the sensors to collaborate efficiently.

Interestingly, `Anneal-BP-1/t` performs relatively poorly here, with no significant improvement over the baseline `SPU`. This is because annealed BP always starts by solving a sum-product BP (since $\lambda^t = 1$ for $t = 1$), whose result is relatively insensitive to the initialization, making it unable to benefit significantly from multiple random initializations. This problem can be fixed by a "perturbed" annealed method that injects a random perturbation into the model, and gradually decreases the perturbation level across iterations (`Anneal-BP-1/t (Perturbed)`); to explain briefly, we take a randomly drawn policy $\delta^0 = \{\delta^0(x_i|\boldsymbol{x}_{\mathrm{pa}(x_i)}) : i \in$



(a) Toy sensor network (b) Junction tree (c) Loopy junction graph

Figure 7.5: Experiments on sensor network detection (3×3 grid). (a) A toy sensor network on 3×3 grid; green lines denote the MRF edges of the hidden process $p(h)$, on some of which (red arrows) signals are allowed to pass; each sensor may be accurate (blue) or noisy (black). Optimal strategies should pass signals from accurate sensors to noisy ones but not the reverse. (b)-(c) The log MEU of algorithms running on (b) a exact junction tree and (c) the loopy junction graph shown in Fig. 7.4(c). As the signal cost increases, all algorithms converge to the communication-free strategy; for lower signal costs, our proposed BP algorithms perform best. Results averaged on 10 random trials.



(a) Sensor network (b) Loopy junction graph

Figure 7.6: Experiments on sensor network detection (random graph). (a) A larger sensor network, defined on a random graph with 30 nodes; the MRF edges overlap with the signal paths. (b) The log MEU of the various algorithms (best of 5 initializations), run on the loopy junction graph shown in Fig. 7.4(c). Again, as the signal cost increases, all algorithms converge to the communication-free strategy; for lower signal costs, the proposed BP algorithms significantly dominate SPU. Results averaged over 5 random trials.

$D\}$, and replace the factor of each decision cluster $\psi_{c_k}(x_{c_k})$, $\forall k \in \mathcal{D}$ in MEU-message update (7.4) with $\hat{\psi}_{c_k}(\boldsymbol{x}_{c_k}) = \psi_{c_k}(\boldsymbol{x}_{c_k})\delta^0(x_i|\boldsymbol{x}_{\mathrm{pa}(x_i)})^{1/t}$ at iteration t of annealed BP, so that $\hat{\psi}_{c_k}$ is perturbed from the true ψ_{c_k} initially, but gradually reduces back to ψ_{c_k} as the iteration number t increases. We repeat (`Anneal-BP-1/t (Perturbed)`) with 5 random δ^0, so the δ^0 play a role similar to the random initial policies in the proximal point methods.

A similar experiment, but with only the loopy junction graph, is performed on a larger randomly generated graph in Figure 7.6; the algorithm performances follow a trend similar to that of Figure 7.5. SPU performs even worse in this case (much worse, even, than `Anneal-BP-1/t`). Investigating, we find that this is mostly because it appears to over-send signals when multiple "good" sensors connect to one "bad" sensor.

7.7 Conclusions and Future Directions

In this chapter we derive a general variational framework for influence diagrams, for both the "convex" centralized decisions with perfect recall and "non-convex" decentralized decisions. We derive several BP-type algorithms, but equally importantly open the door for many others that can be applied within our framework. Since our algorithms *decompose* the global problems into local ones, they also open the possibility of efficient, distributed algorithms.

Future Directions. Structured decision problems will only become more important in the future, as data, systems, and models become increasingly complex. Our general framework opens many new opportunities for future research, including developing even more efficient algorithms by using more advanced (distributed) optimization techniques, as well as extensions to related problems such as infinite horizon decision problems and multiplayer games (e.g., graphical games). See Cheng et al. [2013] for a recent example where we adapted our techniques to solve graph-based Markov decision problems (MDPs).

Chapter 8

Conclusions and Future Directions

In this thesis, we derived a set of general variational representations for various inference and decision making problems, based on which we developed a spectrum of novel and efficient approximation algorithms, including weighted mini-bucket elimination and a set of efficient mixed-product belief propagation algorithms for marginal MAP estimation and MEU policy optimization in decision networks.

Viewed at a high level, our contributions include:

- We derive a unified perspective and variational forms for hybrid inference problems, allowing variational approximation techniques to be applied to many new and important problem domains, and resulting in several powerful new algorithms for these problems.
- Our weighted mini-bucket technique highlights the connections between elimination-based and variational approximation algorithms, allowing the advantages of each to be flexibly balanced and resulting in significant improvements over either in practice.
- By allowing positive or negative weights, weighted mini-bucket also includes upper bounds, related to convex variational methods such as tree-reweighted BP, and lower bounds, related to structured mean field, demonstrating additional connections between these methods.

Our work opens many potential directions for future research, both in terms of improving

our current algorithms, and applying our results and techniques to solve other challenging problems such as learning with intractable likelihoods or hidden variables, graph-based games and partially observable Markov decision processes. While each chapter discusses some open questions and future directions for that component of the thesis, here we discuss some of the larger open directions for research.

Improvements to Our Methods

There are several potential directions to improve our algorithms to give even tighter bounds, or more efficient algorithms.

Improving Weighted Mini-bucket. The quality of weighted mini-bucket can be greatly influenced by the mini-bucket partition policy, which decides the variable scopes of the mini-buckets and the Markov structure of the augmented model. We used the simple scope-based heuristic in our algorithm, but it is worth to study the possibility of developing more efficient partition heuristics, perhaps based on our moment and entropy matching conditions, as well as the potential for iterative "re-partitioning" of the buckets (in addition to the factor reallocation). Additional research in this direction could provide significant improvements in the resulting bounds.

Another interesting problem relates to the increased difficulty in tightening the lower bounds. Although tightening the upper bounds reduces to a convex optimization, optimizing the lower bounds results in a significantly more challenging problem, including searching over an exponential number of disconnected subdomains and non-convex local search inside each subdomain. Although we derived a local optimization algorithm within subdomains, we still lack an efficient algorithm or heuristic to "jump" between the subdomains, which could achieve significantly better results. One possibility would be to extend the heuristic used in negative TRBP, where a similar problem arises; see Liu and Ihler [2010] for further discussion.

Improving Marginal MAP. Related to marginal MAP, one problem of mixed-product BP is that it can (and often does) fail to converge in practice. Although we proposed proximal point algorithms with convergence guarantees, they are relatively slow when the inner loops are expensive, and do not fully exploit the advantage of our variational framework. An important direction would be to derive efficient, *single-loop* algorithms, with better or guaranteed convergence. Various convergence techniques that have been developed for max- and sum- inference may be applied to address this problem, including convergent convex sum-product/max-product BPs [e.g., Meltzer et al., 2009, Hazan and Shashua, 2010], dual decomposition methods [e.g., Sontag et al., 2011, Komodakis et al., 2011], and and more recent augmented Lagrangian methods [e.g., Meshi and Globerson, 2011, Aguiar et al., 2011, Forouzan and Ihler, 2013].

Improving Decision Making. One critical issue in using our variational framework for MEU in influence diagrams is its difficulty when applied on additive utility functions. Although we gave a method to address this problem, additive utilities are so common and important in practice that further improvement, or alterative approaches may be useful to develop.

Learning Graphical Models From Data

Although this thesis focuses on the *inference* and *decision making* problems of *given* graphical models, the problem of constructing, or *learning* graphical models from empirical data is also an important research area. Our techniques and results open several promising directions on improving or deriving new learning algorithms.

Learning with Weighted Mini-bucket. First, the likelihood-based learning methods for graphical models are generally intractable due to the difficulty of calculating the log partition function. Variational approximation methods such as loopy BP and TRBP have been widely

applied to approximate the intractable likelihood, yielding *surrogate likelihood* methods. Our weighted MBE bound can be readily applied for this purpose, and can produce an efficient algorithm due to its simple, closed form representation. In addition, by increasing the *ibound*, weighted MBE provides a sequence of increasingly accurate bounds at the cost of increasing computational effort, providing a flexible way to trade off between model correctness and computational cost, or alternatively, between bias and variance in learning. An interesting recent study is performed in Gelfand et al. [2013], where they demonstrate a "free lunch" scenario, where a cheaper but less accurate approximate inference (weighted MBE with lower *ibound*) performs better than the more expensive and accurate approximation when used for learning in small datasets, as the former exhibits smaller variance and is more robust to model mis-specification.

Marginal MAP and Learning with Hidden Variables. Our results on marginal MAP also imply an array of possibilities for improving learning associated with hidden variables or missing information. An example of this is exploited in our recent work [Ping et al., 2014], where we use mixed-product BP to derive a novel marginal structured SVM algorithm for predicting structured objectives with hidden variables. Many additional directions are also possible. For example, the marginal MAP problem itself can be treated as a learning problem that finds the optimal "parameter" (the max nodes), maximizing a likelihood objective that marginalizes the hidden variables (the sum nodes). The superiority of our algorithm compared to EM and variational EM that we demonstrated in Section 6.5 suggests a possibility of deriving mixed-product BP analogues for learning with hidden variables that find better local optima than EM. A key challenge to address is that most model parameters are continuous variables, and hence require new versions of mixed-product BP that work for continuous-valued max variables.

In addition, the recently popular *deep networks*, such as the deep Boltzmann machine, can be treated as graphical models with large numbers of hidden variables. Currently, marginaliza-

tion over these hidden variables is mostly performed approximately by Markov chain Monte Carlo (MCMC) or mean field type methods [Salakhutdinov and Hinton, 2012]. It would be interesting to see if the marginal MAP perspective can be applied in these cases, e.g., by explicitly training (approximate) marginal MAP predictors to minimize the empirical loss.

☐ Graph-based Games and Markov Decision Processes

While this thesis studied several decision-making systems, there is substantial room to extend our methods to non-cooperative multi-agent settings and infinite-horizon planning.

Graph-based Games and Adversarial Learning. The hybrid tasks we considered in the thesis involve combinations of sum and max operators, both of which correspond to powered sums with positive weights ($w = 1$, and $w \to 0^+$, respectively). However, many important practical problems, especially these associated with games or adversarial learning, also involve min operators, corresponding to a negative weight ($w \to 0^-$). For example, an important extension of influence diagrams is the multi-agent influence diagram representation [Koller and Milch, 2003], which, in the zero-sum, two-players setting, involves both min and max for the two players' decisions, and sum for averaging over random variables. Other related examples are in robust or adversarial learning, which require us to maximize a worse-case objective against an adversary that wishes to degrade (min) the results [e.g., Ibrahimi et al., 2011]. Our Theorem 4.2 derived a general variational representation for both positive and negative weights, providing an opportunity to develop new and efficient algorithms in these settings.

Graph-based Markov Decision Processes. An influence diagram provides a general representation for partially observable Markov decision processes with finite, discrete time horizons. But many practical control and planing problems are defined on infinite, sometimes continuous time horizons. Another, largely open direction is to extend our variational

methodology to these more general problems. An initial approach is presented in our recent work [Cheng et al., 2013], but more theoretical and algorithmic developments along these directions could be greatly in demand.

Bibliography

J. Jiang, P. Rai, and H. Daumé III. Message-passing for approximate MAP inference with latent variables. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

A. Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge University Press, 2009.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

M.J. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

R. Dechter and I. Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM (JACM)*, 50(2):107–153, 2003.

L. Saul and M. Jordan. Exploiting tractable substructures in intractable networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 486–492. MIT Press, 1995.

S. L. Lauritzen. *Graphical Models*. Clarendon Press, 1996.

S.Z. Li and S. Singh. *Markov random field modeling in image analysis*, volume 26. Springer, 2009.

J. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *Journal of Artificial Intelligence Research*, 21:101–133, 2004.

W. Ping, Q. Liu, and A. Ihler. Marginal structured SVM with hidden variables. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 190–198, 2014.

J. Naradowsky, S. Riedel, and D. A. Smith. Improving NLP through marginalization of hidden syntactic structure. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 810–820. Association for Computational Linguistics, 2012.

P. Pletscher and C. S. Ong. Part & clamp: Efficient structured output learning. In *In Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 877–885, 2012.

C.P. de Campos. New complexity results for MAP in Bayesian networks. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2100–2106, 2011.

D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. *Optimization for Machine Learning*, 1:219–254, 2011.

D.M. Greig, B.T. Porteous, and A.H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 271–279, 1989.

M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22(5):1087–1116, 1993.

R.A. Howard and J.E. Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*. Strategic Decisions Group, 1985.

R.D. Shachter. Probabilistic inference and influence diagrams. *Operations Research*, 36(4): 589–604, 1988.

R.D. Shachter. Evaluating influence diagrams. *Operations research*, 34(6):871–882, 1986.

N.L. Zhang, R. Qi, and D. Poole. A computational theory of decision networks. *International Journal of Approximate Reasoning*, 11:83–158, 1994.

A. Detwarasiti and R.D. Shachter. Influence diagrams for team decision analysis. *Decision Analysis*, 2, Dec 2005.

O.P. Kreidl and A.S. Willsky. An efficient message-passing algorithm for optimizing decentralized detection networks. In *2006 IEEE Conference on Decision and Control*, Dec 2006.

S.L. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, pages 1235–1251, 2001.

D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1):181–221, 2003.

X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proceedings, Uncertainty in Artificial Intelligence*, pages 33–42, 1998.

Y. Wexler and C. Meek. MAS: a multiplicative approximation scheme for probabilistic inference. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1761–1768, 2009.

D.D. Mauá and C.P. de Campos. Anytime marginal maximum a posteriori inference. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1996.

R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1):41–85, 1999.

N. L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.

R. Dechter. Reasoning with probabilistic and deterministic graphical models: Exact algorithms. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 7(3):1–191, 2013.

R. Mateescu, K. Kask, V. Gogate, and R. Dechter. Join-graph propagation algorithms. *Journal of Artificial Intelligence Research*, 37(1):279–328, 2010.

A. Ihler, N. Flerova, R. Dechter, and L. Otten. Join-graph based cost-shifting schemes. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 397–406. AUAI Press, Corvallis, Oregon, August 2012.

Q. Liu and A. Ihler. Bounding the partition function using Hölder's inequality. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, New York, NY, USA, 2011.

J. Yarkony, C. Fowlkes, and A. Ihler. Covering trees and lower bounds on quadratic assignment. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

E. Rollon and R. Dechter. New mini-bucket partitioning heuristics for bounding the probability of evidence. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI)*, 2010.

A. Ihler, J.W. Fisher III, and A.S. Willsky. Loopy belief propagation: Convergence and effects of message errors. In *Journal of Machine Learning Research*, pages 905–936, 2005.

G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2006.

M. Welling and Y.W. Teh. Belief optimization for binary networks: A stable alternative to loopy belief propagation. In *Proceedings of the 17th conference on Uncertainty in artificial intelligence*, pages 554–561. Morgan Kaufmann Publishers Inc., 2001.

A.L. Yuille. CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14:2002, 2002.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2009.

E.T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, May 1957.

J.S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 51, July 2005.

F.R. Kschischang, B.J. Frey, and H.A. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

T. Heskes. Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. 2006.

Y. Weiss, C. Yanover, and T. Meltzer. MAP estimation, linear programming and belief propagation with convex free energies. In *Proceedings of the23rd Conference Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. A new class of upper bounds on the log partition function. *Information Theory, IEEE Transactions on*, 51(7):2313–2335, July 2005.

J. Jancsary and G. Matz. Convergent decomposition solvers for tree-reweighted free energies. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, pages 388–398, 2011.

W. Wiegerinck. Approximations with reweighted generalized belief propagation. In *In Proceedings of the 9th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2005.

A. Globerson and T. Jaakkola. Approximate inference using conditional entropy decompositions. In *In Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007a.

T.M. Cover and J.A. Thomas. *Elements of Information Theory*. John Wiley & Sons, INC., 2nd edition, 2006.

E.P. Xing, M.I. Jordan, and S. Russell. A generalized mean field algorithm for variational inference in exponential families. In *Proceedings of the 19th conference on Uncertainty in Artificial Intelligence (UAI)*, pages 583–591, 2002.

Q. Liu and A. Ihler. Negative tree reweighted belief propagation. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010.

A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21, 2007b.

N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):531–552, 2011.

O. Meshi and A. Globerson. An alternating direction method for dual MAP LP relaxation. In *Machine Learning and Knowledge Discovery in Databases*, pages 470–483. Springer, 2011.

P. Aguiar, E.P. Xing, M. Figueiredo, N.A. Smith, and A. Martins. An augmented Lagrangian approach to constrained MAP inference. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 169–176, 2011.

S. Forouzan and A. Ihler. Linear approximation to ADMM for MAP inference. In *Proceedings of the 5th Asian Conference on Machine Learning (ACML)*, pages 48–61, 2013.

D. Wolpert. Information theory – the bridge connecting bounded rational game theory and statistical physics. *Complex Engineered Systems*, pages 262–290, 2006.

G.H. Hardy, J.E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1988.

C. Sutton and A. McCallum. *Machine Learning*, 77(2-3):165–194, 2009.

M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *Information Theory, IEEE Transactions on*, 49(5):1120–1146, 2003a.

J. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173, August 2010.

M. Schmidt. UGM matlab toolbox, 2007. URL http://people.cs.ubc.ca/~schmidtm/Software/UGM.html.

A. Gelfand, R. Dechter, and A. Ihler. Does better inference mean better learning? In *NIPS Workshop on Perturbations, Optimization, and Statistics (POS)*, Dec 2013.

T. Werner. A linear programming approach to max-sum problem: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(7):1165–1179, 2007.

T. Werner. Revisiting the linear programming relaxation approach to Gibbs energy minimization and weighted constraint satisfaction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(8):1474–1488, 2010.

M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. MAP estimation via agreement on (hyper) trees: Message-passing and linear programming approaches. *IEEE Trans. Info. Theory*, 51(11):3697 – 3717, Nov 2003b.

B. Martinet. Régularisation d'inéquations variationnelles par approximations successives. *Revue Française dInformatique et de Recherche Opérationelle*, 4:154–158, 1970.

R.T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877, 1976.

P. Ravikumar, A. Agarwal, and M.J. Wainwright. Message-passing for graph-structured linear programs: Proximal projections, convergence, and rounding schemes. *Journal of Machine Learning Research*, 11:1043–1080, Mar 2010.

M. Teboulle. Entropic proximal mappings with applications to nonlinear programming. *Mathematics of Operations Research*, 17(3):pp. 670–690, 1992.

A. Iusem and M. Teboulle. On the convergence rate of entropic proximal optimization methods. *Computational and Applied Mathematics*, 12:153–168, 1993.

P. Tseng and D.P. Bertsekas. On the convergence of the exponential multiplier method for convex programming. *Mathematical Programming*, 60(1):1–19, 1993.

D.R. Hunter and K. Lange. A tutorial on MM algorithms. *The American Statistician*, 1 (58), February 2004.

T. Meltzer, A. Globerson, and Y. Weiss. Convergent message passing algorithms: a unifying view. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

T. Hazan and A. Shashua. Norm-product belief propagation: Primal-dual message-passing for approximate inference. *Information Theory, IEEE Transactions on*, 56(12):6294–6316, 2010.

V. Jojic, S. Gould, and D. Koller. Accelerated dual decomposition for MAP inference. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

B. Savchynskyy, S. Schmidt, J.H. Kappes, and C. Schnörr. Efficient MRF energy minimization via adaptive diminishing smoothing. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010.

R. Neal and G.E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer, 1998.

A. Doucet, S.J. Godsill, and C.P. Robert. Marginal maximum a posteriori estimation using Markov chain Monte Carlo. *Statistics and Computing*, 12(1), January 2002.

C. Yuan, T.C. Lu, and M.J. Druzdzel. Annealed MAP. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 628–635, 2004.

F. Altarelli, A. Braunstein, A. Ramezanpour, and R. Zecchina. Stochastic optimization by message passing. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(11): P11009, 2011.

M. Ibrahimi, A. Javanmard, Y. Kanoria, and A. Montanari. Robust max-product belief propagation. In *Asilomar Conference on Signals, Systems and Computers*, pages 43–49. IEEE, 2011.

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1568–1583, 2006.

K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210 –2239, Nov 1998.

Q. Liu and A. Ihler. Variational algorithms for marginal MAP. *The Journal of Machine Learning Research*, 14:3073–3108, 2013.

F. Jensen, F.V. Jensen, and S.L. Dittmer. From influence diagrams to junction trees. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 367–373. Morgan Kaufmann, 1994.

G.F. Cooper. A method for using belief networks as influence diagrams. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 55–63, 1988.

R.D. Shachter and M.A. Peot. Decision making using probabilistic inference methods. In *Proceedings of the 8th conference on Uncertainty in Artificial Intelligence (UAI)*, pages 276–283, 1992.

N.L. Zhang. Probabilistic inference in influence diagrams. In *Computational Intelligence*, pages 514–522, 1998.

R. Sabbadin, N. Peyrard, and N. Forsell. A framework and a mean-field algorithm for the local control of spatial processes. *International Journal of Approximate Reasoning*, 2011.

B. Sallans. Variational action selection for influence diagrams. Technical Report OEFAI-TR-2003-29, Austrian Research Institute for Artificial Intelligence, 2003.

A. Nath and P. Domingos. Efficient belief propagation for utility maximization and repeated inference. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI)*, 2010.

R. Dechter. A new perspective on algorithms for optimizing policies under uncertainty. In *International Conference on Artificial Intelligence Planning Systems (AIPS)*, 2000a.

R. Dechter. An anytime approximation for optimizing policies under uncertainty. In *International Conference on Artificial Intelligence Planning Systems (AIPS)*, 2000b.

W. Watthayu. Representing and solving influence diagram in multi-criteria decision making: A loopy belief propagation method. In *International Symposium on Computer Science and its Applications (CSA)*, pages 118 –125, Oct. 2008.

A.L. Madsen and D. Nilsson. Solving influence diagrams using HUGIN, Shafer-Shenoy and lazy propagation. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 17, pages 337–345, 2001.

C. Bielza, P. Muller, and D.R. Insua. Decision analysis by augmented probability simulation. *Management Science*, 45(7):995–1007, 1999.

A. Cano, M. Gómez, and S. Moral. A forward-backward Monte Carlo method for solving influence diagrams. *International Journal of Approximate Reasoning*, 42(1-2):119–135, 2006.

J.M. Charnes and P.P. Shenoy. Multistage Monte Carlo method for solving influence diagrams using local computation. *Management Science*, pages 405–418, 2004.

D. Garcia-Sanchez and M.J. Druzdzel. An efficient sampling algorithm for influence diagrams. In *Proceedings of the Second European Workshop on Probabilistic Graphical Models (PGM–04), Leiden, Netherlands*, pages 97–104, 2004.

M. Luque, T.D. Nielsen, and F.V. Jensen. An anytime algorithm for evaluating unconstrained influence diagrams. In *Proc. PGM*, pages 177–184, 2008.

R. Qi and D. Poole. A new method for influence diagram evaluation. *Computational Intelligence*, 11(3):498–528, 1995.

C. Yuan and X. Wu. Solving influence diagrams using heuristic search. In *Proc. Symp. AI& Math*, 2010.

R. Marinescu. A new approach to influence diagrams evaluation. In *Research and Development in Intelligent Systems XXVI*, pages 107–120. Springer London, 2010.

D.D. Maua and C.P. de Campos. solving decision problems with limited information. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

B. Sallans and G.E. Hinton. Using free energies to represent q-values in a multiagent reinforcement learning task. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1075–1081, 2001.

T. Furmston and D. Barber. Variational methods for reinforcement learning. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 241–248, 2010.

J. Yoshimoto and S. Ishii. A solving method for mdps by minimizing variational free energy. In *International Joint Conference on Neural Networks*, volume 3, pages 1817–1822. IEEE, 2004.

R. Viswanathan and P.K. Varshney. Distributed detection with multiple sensors: part I – fundamentals. *Proc. IEEE*, 85(1):54 –63, Jan 1997.

Q. Cheng, Q. Liu, F. Chen, and A. Ihler. Variational planning for graph-based mdps. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2976–2984, 2013.

R. Salakhutdinov and G. Hinton. An efficient learning procedure for deep boltzmann machines. *Neural computation*, 24(8):1967–2006, 2012.

Appendix A

Derivations and Proofs for Weighted Mini-bucket

A.1 Derivation of the Entropy Matching Condition

We derive the entropy matching condition (5.20) as a necessary condition for optimal weight values in weighted mini-bucket, both when minimizing the upper bound or maximizing the lower bound:

$$\text{Upper bound:} \qquad \min_{\bar{\boldsymbol{w}}} \bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) \quad \text{s.t.} \quad \bar{\boldsymbol{w}} \in \overline{\mathcal{W}}^{+}, \tag{A.1}$$

$$\text{Lower bound:} \qquad \max_{\bar{\boldsymbol{w}}} \bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) \quad \text{s.t.} \quad \bar{\boldsymbol{w}} \in \overline{\mathcal{W}}^{-} := \cup \overline{\mathcal{W}}^{-}_{[\kappa_1 \cdots \kappa_n]}. \tag{A.2}$$

To represent the optimization in $\overline{\mathcal{W}}^{+}$ and $\overline{\mathcal{W}}^{-}$ in a unified way, we denote by $\bar{s}_{i^r}$ the sign of $\bar{w}_{i^r}$ enforced by $\overline{\mathcal{W}}^{+}$ or each $\overline{\mathcal{W}}^{-}_{[\kappa_1 \cdots \kappa_n]}$. That is, we set $\bar{s}_{i^r} = 1$ for all i^r for optimization over $\mathcal{W}^{+}$, and $\bar{s}_{i^{\kappa_i}} = 1$, $\bar{s}_{i^r} = -1$ for all $r \neq \kappa_i$ for optimization over $\overline{\mathcal{W}}^{-}_{[\kappa_1 \cdots \kappa_n]}$. Then, the optimization of weights in $\overline{\mathcal{W}}^{+}$ and each $\overline{\mathcal{W}}^{-}_{[\kappa_1 \cdots \kappa_n]}$ can be both written as

$$\min_{\{\bar{w}_{i^r}\}} \bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) \qquad \text{s.t.} \qquad \sum_r \bar{w}_{i^r} = 1, \quad \bar{s}_{i^r} \bar{w}_{i^r} \geq 0, \quad \forall i^r. \tag{A.3}$$

Using the derivative result in Theorem 5.1, the KKT condition of $\{\bar{w}_{ir}\}$ in (A.3) gives

$$-H_{ir|\prec} = -\mu_{ir}\bar{s}_{ir} + \lambda_i \qquad \text{(Stationarity)}, \tag{A.4}$$

$$\mu_{ir}(-\bar{s}_{ir}\bar{w}_{ir}) = 0 \qquad \text{(Complementary Slackness)}, \tag{A.5}$$

$$\sum_r \bar{w}_{ir} = 1 \qquad \text{(Primal Feasibility)}, \tag{A.6}$$

$$\mu_{ir} \geq 0 \qquad \text{(Dual Feasibility)}, \tag{A.7}$$

where μ_{ir} and λ_i are the KKT multipliers. Then, by multiplying (A.4) by $\bar{w}_{ir}$ and summing over r, we get

$$-\sum_r \bar{w}_{ir} H_{ir|\prec} = -\sum_r \mu_{ir}\bar{s}_{ir}\bar{w}_{ir} + \lambda_i(\sum_r \bar{w}_{ir}).$$

Plugging (A.5) and (A.6) into the above gives

$$-\sum_r \bar{w}_{ir} H_{ir|\prec} = \lambda_i,$$

and hence

$$\mu_{ir}\bar{s}_{ir} = H_{ir|\prec} - \sum_r \bar{w}_{ir} H_{ir|\prec}.$$

Plugging this into (A.5), we get the entropy matching condition,

$$\bar{w}_{ir}[H_{ir|\prec} - H_{i|\prec}] = 0,$$

where $H_{i|\prec} = \sum_r \bar{w}_{ir} H_{ir|\prec}$ is the average entropy over the replicates. Therefore, the conditional entropies $H_{ir|\prec}$ of the replicates should be matched to each other, unless the weight $\bar{w}_{ir}$ equals zero.

A.2 Proof of Theorem 5.2(2)

Theorem 5.2. (2). *For fixed $\bar{\boldsymbol{w}} > 0$, we have*

$$\min_{\bar{\boldsymbol{\theta}} \in \overline{\Theta}} \bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) = \max_{\tau \in \mathbb{L}(\mathcal{G})} \big\{ \langle \tau, \theta \rangle + \sum_{k=1}^{\bar{n}} \bar{w}_k H(x_{\Gamma(k)} | \bar{x}_{\Gamma(c_k \setminus \{k\})}, \tau) \big\}. \tag{A.8}$$

i.e., the optimal choice of $\bar{\boldsymbol{\theta}}$ has a dual form representable in terms of the polytope, marginals, and conditional entropies defined on the cliques of the junction graph $\mathcal{G}$.

Proof. Using the the result of Theorem 5.2(2), we have

$$\begin{aligned}
\min_{\bar{\boldsymbol{\theta}} \in \overline{\Theta}} \bar{\Phi}_w(\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{w}}) &= \min_{\bar{\boldsymbol{\theta}} \in \overline{\Theta}} \max_{\bar{\tau} \in \mathbb{L}(\bar{\mathcal{G}})} \big\{ \langle \bar{\tau}, \bar{\boldsymbol{\theta}} \rangle + \sum_{k=1}^{\bar{n}} \bar{w}_k H_w(\bar{x}_k | \bar{x}_{c_k \setminus \{k\}}, \bar{\tau}) \big\} \\
&= \max_{\bar{\tau} \in \mathbb{L}(\bar{\mathcal{G}})} \min_{\bar{\boldsymbol{\theta}} \in \overline{\Theta}} \big\{ \langle \bar{\tau}, \bar{\boldsymbol{\theta}} \rangle + \sum_{k=1}^{\bar{n}} \bar{w}_k H_w(\bar{x}_k | \bar{x}_{c_k \setminus \{k\}}, \bar{\tau}) \big\} \\
&= \max_{\bar{\tau} \in \mathbb{L}(\bar{\mathcal{G}})} \big\{ \min_{\bar{\boldsymbol{\theta}} \in \overline{\Theta}} \langle \bar{\tau}, \bar{\boldsymbol{\theta}} \rangle + \sum_{k=1}^{\bar{n}} \bar{w}_k H_w(\bar{x}_k | \bar{x}_{c_k \setminus \{k\}}, \bar{\tau}) \big\}.
\end{aligned} \tag{A.9}$$

For $\forall \alpha \in \mathcal{I}$, let $\{\alpha^r\}$ be the collection of replicates of α included in the augmented cliques $\bar{\mathcal{I}}$. Given an initial $\bar{\boldsymbol{\theta}}^0 \in \overline{\Theta}$, we write an arbitrary $\bar{\boldsymbol{\theta}} \in \overline{\Theta}$ as $\bar{\boldsymbol{\theta}} = \bar{\boldsymbol{\theta}}^0 + \sum_\alpha \sum_r \vartheta_{\alpha^r}$, where $\vartheta \in \{\vartheta : \sum_r \vartheta_{\alpha^r}(x_\alpha) = 0, \forall \alpha \in \mathcal{I}, x_\alpha \in \mathcal{X}_\alpha\}$. Here, ϑ_{α^r} can be considered as a reallocation of factors among the replicates. Noting that the inner min of $\bar{\boldsymbol{\theta}}$ in (A.9) involves only a linear form, we have

$$\min_{\bar{\boldsymbol{\theta}} \in \overline{\Theta}} \langle \bar{\tau}, \bar{\boldsymbol{\theta}} \rangle = \langle \bar{\tau}, \bar{\boldsymbol{\theta}}^0 \rangle + \min_{\vartheta} \sum_\alpha \sum_r \langle \tau_{\alpha^r}, \vartheta_{\alpha^r} \rangle$$

where τ_{α^r} are the marginals of $\bar{\tau}$ over x_{α^r}. The minimum of the linear form is negative infinity unless the τ_{α^r} are equal for all the replicates of α, that is, $\bar{\tau}$ should be in a set $\mathcal{D}_\tau = \{\bar{\tau} : \forall \alpha \in \mathcal{I}, \exists \text{ some } \tau_\alpha, \text{ s.t. } \tau_{\alpha^r} = \tau_\alpha \text{ for all } r\}$. Therefore, the optimization in (A.9)

is equivalent to

$$\max_{\bar{\tau} \in \mathbb{L}(\bar{\mathcal{G}}) \cap \mathcal{D}_\tau} \langle \bar{\tau}, \bar{\boldsymbol{\theta}} \rangle + \sum_{k=1}^{\bar{n}} \bar{w}_k H(\bar{x}_k | \bar{x}_{c_k \setminus \{k\}};\ \bar{\tau})$$

Equivalently, this optimization can be collapsed to an optimization over the original model,

$$\max_{\tau \in \mathbb{L}(\mathcal{G})} \langle \tau, \theta \rangle + \sum_{k=1}^{\bar{n}} \bar{w}_k H(\bar{x}_k | \bar{x}_{c_k^0}, \tau)$$

Note that τ is now defined in the space of the marginals on the original model $p(\boldsymbol{x})$. $\square$

Appendix B

Derivations and Proofs for Marginal MAP

B.1 Proof of Proposition 6.2

Proposition 6.2. *Assuming w_i and w_{ij} are strictly positive, the stationary points of* (6.9) *satisfy the fixed point condition of the following message passing update,*

$$\textit{Message Update:} \qquad m_{i\to j}(x_j) \leftarrow \Big[\sum_{x_i} (\psi_i(x_i) m_{\sim i}(x_i))^{\frac{1}{w_i}} \Big(\frac{\psi_{ij}(x_i,x_j)}{m_{j\to i}(x_i)}\Big)^{\frac{1}{w_{ij}}}\Big]^{w_{ij}}, \tag{B.1}$$

Marginal Decoding:

$$\tau_i(x_i) \propto \big[\psi_i(x_i) m_{\sim i}(x_i)\big]^{\frac{1}{w_i}}, \qquad \tau_{ij}(x_i,x_j) \propto \tau_i(x_i)\tau_j(x_j)\left[\frac{\psi_{ij}(x_i,x_j)}{m_{i\to j}(x_j) m_{j\to i}(x_i)}\right]^{\frac{1}{w_{ij}}}, \tag{B.2}$$

where $m_{\sim i}(x_i) := \prod_{k\in\partial_i} m_{k\to i}(x_i)$ is the product of messages sent into node i, and ∂_i is the set of neighboring nodes of i.

Proof. The proof parallels the proof of Theorem 3.1. The Lagrangian of (6.9) w.r.t. the local

consistency constraint $\sum_{x_i} \tau_{ij}(x_i, x_j) = \tau_j(x_j)$ is

$$\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{i \in V} [w_i H_i(\boldsymbol{\tau}) + \lambda_i^0 \sum_{x_i} \tau_i(x_i)] - \sum_{(ij) \in E} [w_{ij} I_{ij}(\boldsymbol{\tau}) + \sum_{x_j} \lambda_{i \to j}(x_j)(\sum_{x_i} \tau_{ij}(x_i, x_j) - \tau_j(x_j))],$$

where $\{\lambda_i^0 \colon i \in V\}$ and $\{\lambda_{j \to i}(x_i) \colon (ij) \in E, x_i \in \mathcal{X}_i\}$ are the Lagrange multipliers. Then, recall that

$$\begin{aligned}
\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle &= \sum_{i \in V} \theta_i(x_i) \tau_i(x_i) + \sum_{(ij) \in E} \theta_{ij}(x_i, x_j) \tau_{ij}(x_i, x_j), \\
H_i(\boldsymbol{\tau}) &= -\sum_{x_i} \tau_i(x_i) \log \tau_i(x_i), \\
I_{ij}(\boldsymbol{\tau}) &= \sum_{x_i, x_j} \tau_{ij}(x_i, x_j) \log \frac{\tau_{ij}(x_i, x_j)}{\sum_{x_i} \tau_{ij}(x_i, x_j) \sum_{x_j} \tau_{ij}(x_i, x_j)}.
\end{aligned}$$

Taking the derivative of the Lagrangian w.r.t. $\tau_i(x_i)$ and $\tau_{ij}(x_i, x_j)$, we have

$$\theta_i(x_i) - w_i \log \tau_i(x_i) + \sum_{j \in \partial_i} \lambda_{j \to i}(x_i) = const, \tag{B.3}$$

$$\theta_{ij}(x_i, x_j) - w_{ij} \log \frac{\tau_{ij}(x_i, x_j)}{\tau_i(x_i) \tau_j(x_j)} - \lambda_{i \to j}(x_j) - \lambda_{j \to i}(x_i) = const, \tag{B.4}$$

where we used the local consistency condition that $\sum_{x_j} \tau_{ij}(x_i, x_j) = \tau_i(x_i)$. By defining $m_{i \to j}(x_j) = \exp(\lambda_{i \to j}(x_j))$, we obtain (B.2) directly from (B.3)-(B.4).
Plugging (B.2) into the constraint that $\sum_{x_j} \tau_{ij}(x_i, x_j) = \tau_i(x_i)$ gives (B.1). □

B.2 Proof of Theorem 6.1

Theorem 6.1. *Suppose the sum part G_A is a tree, and we approximate $\Phi_{AB}(\boldsymbol{\theta})$ using $\Phi_{tree}(\boldsymbol{\theta})$ as defined in* (6.7)*. Assume that* (6.7) *is* globally *optimized; then:*

1. *We have $\Phi_{tree}(\boldsymbol{\theta}) \geq \Phi_{AB}(\boldsymbol{\theta})$. If there exists $\boldsymbol{x}_B^*$ such that $Q(\boldsymbol{x}_B^*; \boldsymbol{\theta}) = \Phi_{tree}(\boldsymbol{\theta})$, we have*

$\Phi_{tree}(\boldsymbol{\theta}) = \Phi_{AB}(\boldsymbol{\theta})$, *and* $\boldsymbol{x}_B^*$ *is a globally optimal marginal MAP solution.*

2. *Suppose* $\boldsymbol{\tau}^*$ *is a* global *maximum of* (6.7)*, and* $\{\tau_i^*(x_i)\colon i \in B\}$ *have integer values, i.e.,* $\tau_i^*(x_i) = 0$ *or* 1*. Then,* $\{x_i^* = \arg\max_{x_i} \tau_i^*(x_i)\colon i \in B\}$ *is a globally optimal solution of the marginal MAP problem* (6.1)*.*

Proof. (1). For $\boldsymbol{\tau} \in \mathbb{M}^o$ (as defined in Corollary 6.1), the objective function in (6.7) equals

$$\begin{aligned} F_{tree}(\boldsymbol{\tau}, \boldsymbol{\theta}) &= \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle \; + \; \sum_{i \in V} H_i(\tau_i) - \sum_{(ij) \in E_A} I_{ij}(\tau_{ij}) - \sum_{(ij) \in \partial_{AB}} \rho_{ij} I_{ij}(\tau_{ij}) \\ &= \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle \; + \; \sum_{i \in V} H_i(\tau_i) - \sum_{(ij) \in E_A} I_{ij}(\tau_{ij}) \qquad \text{(B.5)} \\ &= \langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle \; + H_{A|B}(\boldsymbol{\tau}) \qquad \text{(B.6)} \\ &= F_{mix}(\boldsymbol{\tau}, \boldsymbol{\theta}), \end{aligned}$$

where the equality in (B.5) is because $I_{ij}(\tau_{ij}) = 0$ if $\forall (ij) \in \partial_{AB}$, and the equality in (B.6) is because the sum part G_A is a tree and we have the tree decomposition $H_{A|B} = \sum_{i \in V} H_i(\tau_i) - \sum_{(ij) \in E_A} I_{ij}(\tau_{ij})$. Therefore we have

$$\Phi_{tree}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau} \in \mathbb{L}} F_{tree}(\boldsymbol{\tau}, \boldsymbol{\theta}) \geq \max_{\boldsymbol{\tau} \in \mathbb{M}^o} F_{tree}(\boldsymbol{\tau}, \boldsymbol{\theta}) = \max_{\boldsymbol{\tau} \in \mathbb{M}^o} F_{mix}(\boldsymbol{\tau}, \boldsymbol{\theta}) = \Phi_{AB}(\boldsymbol{\theta}), \qquad \text{(B.7)}$$

where the inequality is because $\mathbb{M}^o \subset \mathbb{M} \subset \mathbb{L}$.

If there exists $\boldsymbol{x}_B^*$ such that $Q(\boldsymbol{x}_B^*; \boldsymbol{\theta}) = \Phi_{tree}(\boldsymbol{\theta})$, then we have

$$Q(\boldsymbol{x}_B^*; \boldsymbol{\theta}) = \Phi_{tree}(\boldsymbol{\theta}) \geq \Phi_{AB}(\boldsymbol{\theta}) = \max_{\boldsymbol{x}_B} Q(\boldsymbol{x}_B; \boldsymbol{\theta}).$$

This proves that $\boldsymbol{x}_B^*$ is a globally optimal marginal MAP solution.

(2). Because $\tau_i^*(x_i)$ for $\forall i \in B$ are deterministic, and the sum part G_A is a tree, we have that $\boldsymbol{\tau}^* \in \mathbb{M}^o$. Therefore the inequality in (B.7) is tight, and we can conclude the proof by

using Corollary 6.1. □

B.3 Proof of Theorem 6.3

Theorem 6.3. *Suppose C is a subset of B such that $G_{C\cup A}$ is a semi-A-B subtree, and the weights $\{\rho_{ij}\}$ satisfy*

1. *$\rho_{ij} = 1$ for $(ij) \in E_A$;*
2. *$0 \le \rho_{ij} \le 1$ for $(ij) \in E_{C\cup A} \cap \partial_{AB}$;*
3. *$\{\rho_{ij} \colon (ij) \in E_{C\cup A} \cap E_B\}$ is provably convex.*

At the fixed point of mixed-product BP in Algorithm 6.2, if the mixed-beliefs on the max nodes $\{b_i, b_{ij} \colon i, j \in B\}$ defined in (6.16) all have unique maxima, then there exists a B-configuration $\boldsymbol{x}_B^*$ satisfying $x_i^* = \arg\max b_i(x_i)$ for $\forall i \in B$ and $(x_i^*, x_j^*) = \arg\max b_{ij}(x_i, x_j)$ for $\forall (ij) \in E_B$, and $\boldsymbol{x}_B^*$ is locally optimal in the sense that $Q(\boldsymbol{x}_B^*; \boldsymbol{\theta})$ is not smaller than any B-configuration that differs from $\boldsymbol{x}_B^*$ only on C, that is, $Q(\boldsymbol{x}_B^*; \boldsymbol{\theta}) = \max_{\boldsymbol{x}_C} Q([\boldsymbol{x}_C, x_{B\setminus C}^*]; \boldsymbol{\theta})$.

Proof. By Theorem 6.2, the beliefs $\{b_i, b_{ij}\}$ should satisfy the reparameterization property in (6.17) and the consistency conditions in (6.18)-(6.20). Without loss of generality, we assume $\{b_i, b_{ij}\}$ are normalized such that $\sum_{x_i} b_i(x_i) = 1$ for $i \in A$ and $\max_{x_i} b_i(x_i) = 1$ for $i \in B$.

I) For simplicity, we first prove the case of $C = B$, when $G = G_{C\cup A}$ itself is a semi A-B tree, and the theorem implies that $\boldsymbol{x}_B^*$ is a global optimum. By the reparameterization condition, we have that

$$p(\boldsymbol{x}) = \hat{p}_B(\boldsymbol{x}_B)\hat{p}_{A|B}(\boldsymbol{x}),$$

where

$$\hat{p}_B(\boldsymbol{x}_B) = \prod_{i \in B} b_i(x_i) \prod_{(ij) \in E_B} \left[\frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)} \right]^{\rho_{ij}}, \tag{B.8}$$

$$\hat{p}_{A|B}(\boldsymbol{x}) = \prod_{i \in A} b_i(x_i) \prod_{(ij) \in E_A} \left[\frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)} \right]^{\rho_{ij}} \prod_{(ij) \in \partial_{AB}} \left[\frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)} \right]^{\rho_{ij}}. \tag{B.9}$$

Note we have

$$p(\boldsymbol{x}_B) = \sum_{\boldsymbol{x}_A} p(\boldsymbol{x}) = \sum_{\boldsymbol{x}_A} \hat{p}_B(\boldsymbol{x}_B) \hat{p}_{A|B}(\boldsymbol{x}) = \hat{p}_B(\boldsymbol{x}_B) \sum_{\boldsymbol{x}_A} \hat{p}_{A|B}(\boldsymbol{x}).$$

We just need to show that $\boldsymbol{x}_B^*$ maximizes $\hat{p}_B(\boldsymbol{x}_B)$ and $\sum_{\boldsymbol{x}_A} \hat{p}_{A|B}(\boldsymbol{x})$, respectively.

First, since $\hat{p}_B(\boldsymbol{x}_B)$ involves only the max nodes, a standard MAP analysis applies. Because the max part of the beliefs, $\{b_i, b_{ij} \colon (ij) \in E_B\}$, satisfy the standard max-consistency conditions, and the corresponding TRW weights $\{\rho_{ij} \colon (ij) \in E_B\}$ are provably convex by assumption, we establish that $\boldsymbol{x}_B^*$ is the MAP solution of $\hat{p}_B(\boldsymbol{x}_B)$ by Theorem 1 of Weiss et al. [2007].

Secondly, to show that $\boldsymbol{x}_B^*$ also maximizes $\hat{p}_{A|B}(\boldsymbol{x})$ requires the combination of the mixed-consistency and sum-consistency conditions. Since G is a semi A-B tree, we denote by π_i the unique parent node of i ($\pi_i = \emptyset$ if i is a root). In addition, let ∂_A be the subset of A whose parent nodes are in B, that is, $\partial_A = \{i \in A \colon \pi_i \in B\}$. Equation (B.9) can be rewritten as

$$\hat{p}_{A|B}(\boldsymbol{x}) = \prod_{i \in A \setminus \partial_A} \frac{b_{i,\pi_i}(x_i, x_{\pi_i})}{b_{\pi_i}(x_{\pi_i})} \prod_{i \in \partial_A} \left[\frac{b_{i,\pi_i}(x_i, x_{\pi_i})}{b_{\pi_i}(x_{\pi_i})} \right]^{\rho_{i,\pi_i}} \left[b_i(x_i) \right]^{1 - \rho_{i,\pi_i}},$$

where we use the fact that $\rho_{ij} = 1$ for $(ij) \in E_A$. Therefore, we have for any $\boldsymbol{x}_B \in \mathcal{X}_B$,

$$\begin{aligned}
\sum_{\boldsymbol{x}_A} \hat{p}_{A|B}(\boldsymbol{x}) &= \sum_{\boldsymbol{x}_A} \left\{ \prod_{i \in A \setminus \partial_A} \frac{b_{i,\pi_i}(x_i, x_{\pi_i})}{b_{\pi_i}(x_{\pi_i})} \prod_{i \in \partial_A} \left[\frac{b_{i,\pi_i}(x_i, x_{\pi_i})}{b_{\pi_i}(x_{\pi_i})} \right]^{\rho_{i,\pi_i}} \left[b_i(x_i) \right]^{1-\rho_{i,\pi_i}} \right\} \\
&= \prod_{i \in \partial_A} \sum_{x_i} \left[\frac{b_{i,\pi_i}(x_i, x_{\pi_i})}{b_{\pi_i}(x_{\pi_i})} \right]^{\rho_{i,\pi_i}} \left[b_i(x_i) \right]^{1-\rho_{i,\pi_i}} \qquad \text{(B.10)} \\
&\leq \prod_{i \in \partial_A} \left[\sum_{x_i} \frac{b_{i,\pi_i}(x_i, x_{\pi_i})}{b_{\pi_i}(x_{\pi_i})} \right]^{\rho_{i,\pi_i}} \left[\sum_{x_i} b_i(x_i) \right]^{1-\rho_{i,\pi_i}} \qquad \text{(B.11)} \\
&= 1, \qquad \text{(B.12)}
\end{aligned}$$

where the equality in (B.10) eliminates (by summation) all the interior nodes in A. The inequality in (B.11) follows from Hölder's inequality. Finally, the equality in (B.12) holds because the sum part of beliefs $\{b_i, b_{ij} \colon (ij) \in E_A\}$ satisfies the sum-consistency (6.18).

On the other hand, for any $(i, \pi_i) \in \partial_{AB}$, because $x^*_{\pi_i} = \arg\max_{x_{\pi_i}} b_{\pi_i}(x_{\pi_i})$, we have $b_{i,\pi_i}(x_i, x^*_{\pi_i}) = b_i(x_i)$ by the mixed-consistency condition (6.20). Therefore,

$$\begin{aligned}
\sum_{\boldsymbol{x}_A} \hat{p}_{A|B}([\boldsymbol{x}_A, \boldsymbol{x}^*_B]) &= \prod_{i \in \partial_A} \sum_{x_i} \left[\frac{b_{i,\pi_i}(x_i, x^*_{\pi_i})}{b_{\pi_i}(x^*_{\pi_i})} \right]^{\rho_{i,\pi_i}} \left[b_i(x_i) \right]^{1-\rho_{i,\pi_i}} \\
&= \prod_{i \in \partial_A} \left[\frac{1}{b_{\pi_i}(x^*_{\pi_i})} \right]^{\rho_{i,\pi_i}} \sum_{x_i} b_i(x_i) \\
&= 1. \qquad \text{(B.13)}
\end{aligned}$$

Combining (B.12) and (B.13), we have $\sum_{\boldsymbol{x}_A} \hat{p}_{A|B}(\boldsymbol{x}) \leq \sum_{\boldsymbol{x}_A} \hat{p}_{A|B}([\boldsymbol{x}_A, \boldsymbol{x}^*_B]) = 1$ for any $\boldsymbol{x}_B \in \mathcal{X}_B$, that is, $\boldsymbol{x}^*_B$ maximizes $\sum_{\boldsymbol{x}_A} \hat{p}_{A|B}(\boldsymbol{x})$. This finishes the proof for the case $C = B$.

II) In the case of $C \neq B$, let $D = B \setminus C$. We decompose $p(\boldsymbol{x})$ into

$$p(\boldsymbol{x}) = \hat{p}_B([\boldsymbol{x}_C, \boldsymbol{x}_D]) \hat{p}_{A|C}([\boldsymbol{x}_A, \boldsymbol{x}_C]) \hat{r}_{AD}([\boldsymbol{x}_A, \boldsymbol{x}_D])$$

where $\hat{p}_B(\boldsymbol{x}_B)$ and $\hat{p}_{A|B}(\boldsymbol{x})$ are defined similarly to (B.8) and (B.9),

$$\hat{p}_B(\boldsymbol{x}_B) = \prod_{i \in B} b_i(x_i) \prod_{(ij) \in E_B} \left[\frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)} \right]^{\rho_{ij}},$$
$$\hat{p}_{A|C}([\boldsymbol{x}_A, \boldsymbol{x}_C]) = \prod_{i \in A} b_i(x_i) \prod_{(ij) \in E_A} \left[\frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)} \right]^{\rho_{ij}} \prod_{(ij) \in \partial_{AC}} \left[\frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)} \right]^{\rho_{ij}},$$

where π_i is the parent node of i in the semi A-B tree $G_{A \cup C}$ and ∂_{AC} is set of edges across A and C, that is, $\partial_{AC} = \{(ij) \in E \colon i \in A, j \in C\}$. The term $\hat{r}_{AD}(\boldsymbol{x})$ is defined as

$$\hat{r}_{AD}([\boldsymbol{x}_A, \boldsymbol{x}_D]) = \prod_{(ij) \in \partial_{AD}} \left[\frac{b_{ij}(x_i, x_j)}{b_i(x_i) b_j(x_j)} \right]^{\rho_{ij}},$$

where similarly ∂_{AD} is the set of edges across A and D.

Because $x_j^* = \arg\max_{x_j} b_j(x_j)$ for $j \in D$, we have $b_{ij}(x_i, x_j^*) = b_i(x_i)$ for $(ij) \in \partial_{AD}$, $j \in D$ by the mixed-consistency condition in (6.20). Therefore, one can show that $\hat{r}_{AD}([\boldsymbol{x}_A, \boldsymbol{x}_D^*]) = 1$, and hence

$$p([\boldsymbol{x}_A, \boldsymbol{x}_C, \boldsymbol{x}_D^*]) = \hat{p}_B([\boldsymbol{x}_C, \boldsymbol{x}_D^*]) \hat{p}_{A|C}([\boldsymbol{x}_A, \boldsymbol{x}_C]).$$

The remainder of the proof is similar to that for the case $C = B$: by the analysis in Weiss et al. [2007], it follows that $\boldsymbol{x}_C^* \in \arg\max_{\boldsymbol{x}_C} p([\boldsymbol{x}_C, \boldsymbol{x}_D^*])$, and we have previously shown that $\boldsymbol{x}_C^* \in \arg\max_{\boldsymbol{x}_C} \sum_{\boldsymbol{x}_A} \hat{p}_{A|C}([\boldsymbol{x}_A, \boldsymbol{x}_C])$. This establishes that x_C^* maximizes

$$\sum_{\boldsymbol{x}_A} p([\boldsymbol{x}_A, \boldsymbol{x}_C, \boldsymbol{x}_D^*]) = p([\boldsymbol{x}_C, \boldsymbol{x}_D^*]) \sum_{\boldsymbol{x}_A} \hat{p}_{A|C}([\boldsymbol{x}_A, \boldsymbol{x}_C]),$$

which concludes the proof. □

B.4 Factor Graph BP for Marginal MAP

In this section, we derive the factor graph mixed-product belief propagation given in Algorithm 6.6, Section 6.6.

Define $\epsilon_i = 1$ for $i \in A$ and $\epsilon_i = \epsilon$ for $i \in B$. Using the KKT condition, one can show that the optima of (6.25) satisfy

$$\tau_i(x_i) \propto (\psi_i(x_i) \prod_{\alpha \in \partial_i} m_{\alpha \to i}(x_i))^{1/\epsilon_i}, \qquad \forall i \in A \tag{B.14}$$

$$\tau_\alpha(\boldsymbol{x}_{\alpha_A} | \boldsymbol{x}_{\alpha_B}) \tau_\alpha(\boldsymbol{x}_{\alpha_B})^\epsilon \propto \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{i \in \alpha} \frac{\tau_i(x_i)^{\epsilon_i}}{m_{\alpha \to i}(x_i)}, \qquad \forall \alpha \in \mathcal{I}, \tag{B.15}$$

where $\psi_i(x_i) = \exp(\theta_i(x_i))$, $\psi_\alpha = \exp(\theta_\alpha(\boldsymbol{x}_\alpha))$ and the *factor-to-variable* messages $m_{\alpha \to i}(x_i)$ are the exponential of the Lagrangian multipliers (via derivations similar to that of Algorithm 6.2 for pairwise models). Let us define the *variable-to-factor* messages $m_{i \to \alpha}$ via

$$m_{i \to \alpha}(x_i) \propto \prod_i \psi_i(x_i) \prod_{\alpha' \in \partial i \setminus \{\alpha\}} m_{\alpha' \to i}(x_i).$$

Combining this with (B.14) and (B.15) gives

$$\tau_i(x_i)^{\epsilon_i} \propto m_{\alpha \to i}(x_i) m_{i \to \alpha}(x_i), \qquad \forall i \in A \tag{B.16}$$

$$\tau_\alpha(\boldsymbol{x}_{\alpha_A} | \boldsymbol{x}_{\alpha_B}) \tau_\alpha(\boldsymbol{x}_{\alpha_B})^\epsilon \propto \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{i \in \alpha} m_{i \to \alpha}(x_i), \qquad \forall \alpha \in \mathcal{I}, \tag{B.17}$$

Summing over $\boldsymbol{x}_{\alpha_A}$ in (B.17) gives

$$\tau_\alpha(\boldsymbol{x}_{\alpha_B})^\epsilon = \sum_{\boldsymbol{x}_{\alpha_A}} \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{i \in \alpha} m_{i \to \alpha}(x_i) \stackrel{def}{=} b_\alpha(\boldsymbol{x}_{\alpha_B})$$

and hence

$$\begin{aligned}\tau_\alpha(\boldsymbol{x}_\alpha) = \tau_\alpha(\boldsymbol{x}_{\alpha_A}|\boldsymbol{x}_{\alpha_B})\tau_\alpha(\boldsymbol{x}_{\alpha_B}) &= \tau_\alpha(\boldsymbol{x}_{\alpha_A}|\boldsymbol{x}_{\alpha_B})\tau_\alpha(\boldsymbol{x}_{\alpha_B})^\epsilon \ \times \ \tau_\alpha(\boldsymbol{x}_{\alpha_B})^{1-\epsilon} \\ &\propto \psi_\alpha(\boldsymbol{x}_\alpha)\prod_{i\in\alpha} m_{i\to\alpha}(x_i) \ \times \ b_\alpha(\boldsymbol{x}_{\alpha_B})^{1/\epsilon-1} \end{aligned} \tag{B.18}$$

Plugging (B.18) and (B.16) into the consistency constraint $\sum_{\boldsymbol{x}_{\alpha\setminus\{i\}}} \tau_\alpha(\boldsymbol{x}_\alpha) = \tau_i(x_i)$, we get

$$(m_{\alpha\to i}(x_i)m_{i\to\alpha}(x_i))^{1/\epsilon_i} \propto \sum_{\boldsymbol{x}_{\alpha\setminus\{i\}}} \psi_\alpha(\boldsymbol{x}_\alpha)\prod_{i'\in\alpha} m_{i'\to\alpha}(x_{i'}) \times b_\alpha(\boldsymbol{x}_{\alpha_B})^{1/\epsilon-1}$$

This reduces to

$$\text{for } i\in A\text{:} \qquad m_{\alpha\to i}(x_i) \propto \sum_{\boldsymbol{x}_{\alpha\setminus\{i\}}} \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{i'\in\alpha\setminus\{i\}} m_{i'\to\alpha}(x_{i'}) \times b_\alpha(\boldsymbol{x}_{\alpha_B})^{1/\epsilon-1}, \tag{B.19}$$

$$\text{for } i\in B\text{:} \qquad m_{\alpha\to i}(x_i) \propto \Big[\sum_{\boldsymbol{x}_{\alpha\setminus\{i\}}} \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{i'\in\alpha\setminus\{i\}} m_{i'\to\alpha}(x_{i'}) \times \frac{b_\alpha(\boldsymbol{x}_{\alpha_B})^{1/\epsilon-1}}{m_{i\to\alpha}(x_i)^{1/\epsilon-1}}\Big]^\epsilon, \tag{B.20}$$

Letting $\epsilon\to 0^+$, we get the message updates in Algorithm 6.6 (via derivations similar to that of pairwise messages):

$$\text{for } i\in A\text{:} \quad m_{\alpha\to i}(x_i) \propto \sum_{\boldsymbol{x}_{\alpha\setminus\{i\}}} \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{i'\in\alpha\setminus\{i\}} m_{i'\to\alpha}(x_{i'}) \times \mathbf{1}[\boldsymbol{x}_{\alpha_B} \in \arg\max_{\boldsymbol{x}_{\alpha_B}} b_\alpha(\boldsymbol{x}_{\alpha_B})], \tag{B.21}$$

$$\text{for } i\in B\text{:} \quad m_{\alpha\to i}(x_i) \propto \max_{\boldsymbol{x}_{\alpha_B\setminus\{i\}}} \sum_{\boldsymbol{x}_{\alpha_A}} \psi_\alpha(\boldsymbol{x}_\alpha) \prod_{i'\in\alpha\setminus\{i\}} m_{i'\to\alpha}(x_{i'}), \tag{B.22}$$

where (B.21) uses the fact that

$$\lim_{\epsilon\to 0^+} \frac{1}{C^{1/\epsilon}} \sum_x g(x)f(x)^{1/\epsilon} = \sum_x g(x)\mathbf{1}[x\in\arg\max f(x)], \quad \text{where } \ C = \max_x f(x),$$

and (B.22) uses the fact that

$$\lim_{\epsilon \to 0^+} \big[\sum_x g(x) f(x)^{1/\epsilon} \big]^\epsilon = \max_x f(x),$$

for any finite positive functions $f(x)$ and $g(x)$.

Appendix C

Derivations and Proofs for Decision Making

C.1 Randomized vs. Deterministic Strategies

It is a well-known fact in decision theory that no randomized strategy can improve on the utility of the best deterministic strategy, so that:

Lemma C.1. *For any ID,* $\max_{\boldsymbol{\delta}\in\Delta} \mathrm{EU}(\boldsymbol{\delta}) = \max_{\boldsymbol{\delta}\in\Delta^o} \mathrm{EU}(\boldsymbol{\delta})$.

Proof. Since $\Delta^o \subset \Delta$, we need to show that for any randomized strategy $\boldsymbol{\delta} \in \Delta$, there exists a deterministic strategy $\boldsymbol{\delta}' \in \Delta^o$ such that $\mathrm{EU}(\boldsymbol{\delta}) \leq \mathrm{EU}(\boldsymbol{\delta}')$. Note that

$$\mathrm{EU}(\boldsymbol{\delta}) = \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \prod_{i\in D} \delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}),$$

where $q(\boldsymbol{x})$ is the augmented distribution. Thus, $\mathrm{EU}(\boldsymbol{\delta})$ is linear on $\delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$ for any $i \in D$, with all the other policies fixed; therefore, one can always replace $\delta_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$ with some deterministic $\delta'_i(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$ without decreasing $\mathrm{EU}(\boldsymbol{\delta})$. Doing so sequentially for all $i \in D$ yields to a deterministic rule $\boldsymbol{\delta}'$ with $\mathrm{EU}(\boldsymbol{\delta}) \leq \mathrm{EU}(\boldsymbol{\delta}')$. □

One can further show that any (globally) optimal randomized strategy can be represented as a convex combination of a set of optimal deterministic strategies.

C.2 Variational Representation of MEU

The following lemma is the "dual" version of Lemma C.1; it will be helpful for proving Corollary 4.2 and Corollary 4.3.

Lemma C.2. *Let $\mathbb{M}^o$ be the set of distributions $\tau(\boldsymbol{x})$ in which $\tau(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}), \forall i \in D$ are deterministic (that is, $\tau(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) = 1$ or 0). Then the optimization domain $\mathbb{M}$ in (4.19) of Theorem 4.3 can be replaced by $\mathbb{M}^o$ without changing the result, that is,*

$$\log \mathrm{MEU}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau} \in \mathbb{M}^o} \{\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}; \boldsymbol{\tau}) - \sum_{i \in D} H(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}; \boldsymbol{\tau})\}. \tag{C.1}$$

Proof. Note that $\mathbb{M}^o$ corresponds to the set of deterministic strategies Δ^o. The proof of Lemma C.1 shows that there always exists at least one optimal deterministic strategy. This implies that at least one optimal solution of (4.19) falls in $\mathbb{M}^o$. The result follows. □

Corollary 4.2. *For an influence diagram with natural parameter $\boldsymbol{\theta}$, we have*

$$\log \mathrm{MEU}(\boldsymbol{\theta}) = \max_{\boldsymbol{\tau} \in \mathbb{I}} \{\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + \sum_{i \in C} H(x_{o_i}|\boldsymbol{x}_{o_{1:i-1}}; \boldsymbol{\tau})\},$$

where $\mathbb{I} = \{\boldsymbol{\tau} \in \mathbb{M} \colon x_{o_i} \perp \boldsymbol{x}_{o_{1:i-1} \setminus \mathrm{pa}(o_i)} | \boldsymbol{x}_{\mathrm{pa}(o_i)}\}$, corresponding to the distributions respecting the imperfect recall structures (i.e., the decision variables only condition on their parent sets); "$x \perp y \,|\, z$" means that x and y are conditionally independent given z.

Proof. For any $\boldsymbol{\tau} \in \mathbb{I}$, we have $H(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}; \boldsymbol{\tau}) = H(x_i|\boldsymbol{x}_{o_{1:i-1}}; \boldsymbol{\tau})$, hence by the entropic chain rule, the objective function in (4.23) is the same as that in (4.19).

Then, for any $\boldsymbol{\tau} \in \mathbb{M}^o$, because $\tau(x_{o_i}|\boldsymbol{x}_{\mathrm{pa}(o_i)}), \forall o_i \in D$ is deterministic, we have $0 \leq H(x_{o_i}|\boldsymbol{x}_{o_{1:i-1}}) \leq H(x_{o_i}|\boldsymbol{x}_{\mathrm{pa}(o_i)}) = 0$, which implies

$$I(x_{o_i}; \boldsymbol{x}_{o_{1:i} \setminus \mathrm{pa}(o_i)}|\boldsymbol{x}_{\mathrm{pa}(o_i)}) = H(x_{o_i}|\boldsymbol{x}_{\mathrm{pa}(o_i)}) - H(x_{o_i}|\boldsymbol{x}_{o_{1:i-1}}) = 0.$$

Therefore, we have $\mathbb{M}^o \subseteq \mathbb{I} \subseteq \mathbb{M}$. We thus have that the LHS of (4.23) is no larger than (4.19), while no smaller than (C.1). The result follows since (4.19) and (C.1) equal by Lemma C.2. □

Corollary 4.3. *For any ϵ, let $\boldsymbol{\tau}^*$ be an optimum of*

$$\max_{\boldsymbol{\tau} \in \mathbb{M}} \{\langle \boldsymbol{\theta}, \boldsymbol{\tau} \rangle + H(\boldsymbol{x}) - (1-\epsilon) \sum_{i \in D} H(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})\}.$$

If $\boldsymbol{\delta}^* = \{\tau^*(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) | i \in D\}$ is a deterministic strategy, then it is an optimal strategy of the influence diagram.

Proof. First, we have $H(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}; \boldsymbol{\tau}) = 0$ for $\boldsymbol{\tau} \in \mathbb{M}^o$ and $i \in D$, since such $\tau(x_i|\boldsymbol{x}_{\mathrm{pa}(i)})$ are deterministic. Therefore, the objective functions in (4.24) and (C.1) are equivalent when the maximization domains are restricted to $\mathbb{M}^o$. The result follows by applying Lemma C.2. □

C.3 Belief Propagation for MEU

We provide the derivation of MEU-BP in Algorithm 7.1 and Algorithm 7.2, and prove the reparameterization property in Theorem 7.1 and the optimality guarantee in Theorem 7.1.

C.3.1 Derivation of MEU-BP

We now derive the MEU-BP in Algorithm 7.1 for solving (7.9) using a Lagrange multiplier method similar to Yedidia et al. [2005]. Consider the Lagrange function of (4.24),

$$\sum_{k\in\mathcal{R}\cup\mathcal{D}}\sum_{\boldsymbol{x}_{c_k}}\theta_{c_k}(\boldsymbol{x}_{c_k})\tau_{c_k}(\boldsymbol{x}_{c_k}) + \sum_{k\in\mathcal{R}}H(\boldsymbol{x}_{c_k};\boldsymbol{\tau}) + \sum_{k\in\mathcal{D}}H^{\epsilon}(\boldsymbol{x}_{c_k};\boldsymbol{\tau}) - \sum_{(kl)\in\mathcal{E}}H(\boldsymbol{x}_{s_{kl}};\boldsymbol{\tau}) + \sum_{(kl)\in\mathcal{E}}\sum_{\boldsymbol{x}_{s_{kl}}}\lambda_{s_{k\to l}}(\boldsymbol{x}_{s_{kl}})[\sum_{\boldsymbol{x}_{c_k\setminus s_{kl}}}\tau_{c_k}(\boldsymbol{x}_{c_k}) - \tau_{s_{kl}}(\boldsymbol{x}_{s_{kl}})], \tag{C.2}$$

where $\lambda_{s_{k\to l}}(\boldsymbol{x}_{s_{kl}})$ are the Lagrange multipliers for the local consistency constraints,

$$\sum_{x_{s_{kl}}}\tau_{c_k}(\boldsymbol{x}_{c_k}) = \tau_{s_{kl}}(\boldsymbol{x}_{s_{kl}}), \quad \forall k\in\mathcal{C},\ (kl)\in\mathcal{E},\ \boldsymbol{x}_{s_{kl}}\in\mathcal{X}_{s_{kl}}, \tag{C.3}$$

while the nonnegative and normalization constraints of $\boldsymbol{\tau}$ are not included and are enforced directly. Taking the gradient of (C.2) w.r.t. τ_{c_k} and $\tau_{s_{kl}}$, one can show that

$$\tau_{c_k}(\boldsymbol{x}_{c_k}) \propto \psi_{c_k}(\boldsymbol{x}_{c_k})m_{\sim k}(\boldsymbol{x}_{c_k}), \qquad \textit{for normal clusters } (k\in\mathcal{C}); \tag{C.4}$$

$$\tau_{c_k}(\boldsymbol{x}_{c_k}) \propto \psi_{c_k}(\boldsymbol{x}_{c_k})m_{\sim k}(\boldsymbol{x}_{c_k})\cdot\delta_{\epsilon}(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(c_k)})^{1-\epsilon}, \qquad \textit{for decision clusters } (k\in\mathcal{D}); \tag{C.5}$$

$$\tau_{s_{kl}}(\boldsymbol{x}_{s_{kl}}) \propto m_{k\to l}(\boldsymbol{x}_{s_{kl}})m_{l\to k}(\boldsymbol{x}_{s_{kl}}), \qquad \textit{for all the separators } ((kl)\in\mathcal{E}). \tag{C.6}$$

where $\psi_{c_k}(\boldsymbol{x}_{c_k}) = \exp(\theta_{c_k}(\boldsymbol{x}_{c_k}))$, $m_{k\to l}(\boldsymbol{x}_{s_{kl}}) = \exp(\lambda_{k\to l}(\boldsymbol{x}_{s_{kl}}))$, and $m_{\sim k}(\boldsymbol{x}_{c_k})$ is the product of messages sent into cluster c_k from its neighboring clusters $\mathcal{N}(k) := \{l\in\mathcal{C}\colon (kl)\in\mathcal{E}\}$, that is, $m_{\sim k}(\boldsymbol{x}_{c_k}) = \prod_{l\in\mathcal{N}(k)} m_{l\to k}(\boldsymbol{x}_{s_{kl}})$. Substituting the local consistency constraints in (C.3) into (7.5)-(7.7) leads to the fixed point updates in (7.3)-(7.4).

It only remains to prove (C.4)-(C.6). The proofs of (C.4) for the normal clusters and (C.6) for the separators are relatively straightforward: taking the gradient of (C.2) w.r.t. $\tau_{c_k}(\boldsymbol{x}_{c_k})$

for $\forall k \in \mathcal{R}$, and $\tau_{s_{kl}}(\boldsymbol{x}_{s_{kl}})$ for $\forall (kl) \in \mathcal{E}$, we have

$$\theta_{c_k}(\boldsymbol{x}_{c_k}) - \log \tau_{c_k}(\boldsymbol{x}_{c_k}) + \sum_{l \in \mathcal{N}(k)} \lambda_{l \to k}(\boldsymbol{x}_{s_{kl}}) + \text{const} = 0, \quad \forall k \in \mathcal{R},\ \boldsymbol{x}_{c_k} \in \mathcal{X}_{c_k},$$

$$\log \tau_{s_{kl}}(\boldsymbol{x}_{s_{kl}}) - \lambda_{l \to k}(\boldsymbol{x}_{s_{kl}}) - \lambda_{k \to l}(\boldsymbol{x}_{s_{kl}}) + \text{const} = 0, \quad \forall (kl) \in \mathcal{E},\ \boldsymbol{x}_{s_{kl}} \in \mathcal{X}_{s_{kl}},$$

which immediately leads to (C.4) and (C.6), respectively.

The proof of (C.5) for the decision clusters is more involved. Recall that

$$H^{\epsilon}(\boldsymbol{x}_{c_k}; \boldsymbol{\tau}) = H(\boldsymbol{x}_{c_k}; \boldsymbol{\tau}) - (1 - \epsilon) H(x_{d_k} | \boldsymbol{x}_{\text{pa}(d_k)}; \boldsymbol{\tau}),$$

Taking the gradient of (C.2) w.r.t. $\tau_{c_k}(\boldsymbol{x}_{c_k})$ for $\forall k \in \mathcal{D}$, we have

$$\theta_{c_k}(\boldsymbol{x}_{c_k}) - \log \tau_{c_k}(\boldsymbol{x}_{c_k}) + (1 - \epsilon) \tau_{c_k}(x_{d_k} | \boldsymbol{x}_{\text{pa}(d_k)}) + \sum_{l \in \mathcal{N}(k)} \lambda_{l \to k}(\boldsymbol{x}_{s_{kl}}) + \text{const} = 0, \tag{C.7}$$

where here and in what follows $\tau_{c_k}(\boldsymbol{x}_{c_k})$ and $\tau_{c_k}(\boldsymbol{x}_{\text{pa}(d_k)})$ are the conditional and marginalized probabilities induced by $\tau_{c_k}(\boldsymbol{x}_{c_k})$, respectively. Observe from (C.7) that

$$\tau_{c_k}(\boldsymbol{x}_{c_k}) \propto \psi_{c_k}(\boldsymbol{x}_{c_k}) m_{\sim k}(\boldsymbol{x}_{c_k}) \cdot \tau_{c_k}(x_{d_k} | \boldsymbol{x}_{\text{pa}(d_k)})^{1-\epsilon}. \tag{C.8}$$

Therefore, we only need to prove that $\tau_{c_k}(x_{d_k} | \boldsymbol{x}_{\text{pa}(d_k)}) = \delta_{\epsilon}(x_{d_k} | \boldsymbol{x}_{\text{pa}(c_k)})$. Summing over $\boldsymbol{x}_{c_k \setminus \{d_k, \text{pa}(d_k)\}}$ on both side of (C.8), we have

$$\begin{aligned} b_{c_k}(x_{d_k}, \boldsymbol{x}_{\text{pa}(d_k)}) &= \sum_{\boldsymbol{x}_{c_k \setminus \{d_k, \text{pa}(d_k)\}}} \psi_{c_k}(\boldsymbol{x}_{c_k}) m_{\sim k}(\boldsymbol{x}_{c_k}) \\ &\propto \tau_{c_k}(x_{d_k}, \boldsymbol{x}_{\text{pa}(d_k)}) \tau_{c_k}(x_{d_k} | \boldsymbol{x}_{\text{pa}(d_k)})^{\epsilon - 1} \\ &= \tau_{c_k}(\boldsymbol{x}_{\text{pa}(d_k)}) \tau_{c_k}(x_{d_k} | \boldsymbol{x}_{\text{pa}(d_k)})^{\epsilon}. \end{aligned} \tag{C.9}$$

Raising both sides of (C.9) to the power $1/\epsilon$ and then summing over x_{d_k}, we have

$$[\tau_{c_k}(\boldsymbol{x}_{\mathrm{pa}(d_k)})]^{1/\epsilon} \propto \sum_{x_{d_k}} [b_{c_k}(x_{d_k}, \boldsymbol{x}_{\mathrm{pa}(d_k)})]^{1/\epsilon}. \tag{C.10}$$

Combining (C.10) with (C.8), we obtain

$$\begin{aligned}\tau_{c_k}(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(d_k)}) &\propto \frac{b_{c_k}(x_{d_k}, \boldsymbol{x}_{\mathrm{pa}(d_k)})^{1/\epsilon}}{[\tau_{c_k}(\boldsymbol{x}_{\mathrm{pa}(d_k)})]^{1/\epsilon}} \\ &= \frac{b_{c_k}(x_{d_k}, \boldsymbol{x}_{\mathrm{pa}(d_k)})^{1/\epsilon}}{\sum_{x_{d_k}} b_{c_k}(x_{d_k}, \boldsymbol{x}_{\mathrm{pa}(d_k)})^{1/\epsilon}} = \delta_\epsilon(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(d_k)}).\end{aligned}$$

This concludes the proof of (C.5).

C.3.2 Correctness Guarantees

Theorem 7.1. *Let $(\mathcal{G}, \mathcal{C}, \mathcal{S})$ be a junction tree consistent on a subset of decision nodes D'. If the MEU-BP (with zero temperature) in Algorithm 7.2 converges, then the decoded $\boldsymbol{\delta}^*$ is a locally optimal strategy in the sense that $\mathrm{EU}(\{\boldsymbol{\delta}_{D'}, \boldsymbol{\delta}^*_{D\setminus D'}\}) \leq \mathrm{EU}(\boldsymbol{\delta}^*)$ for any $\boldsymbol{\delta}_{D'}$.*

Proof. On a junction tree, the reparameterization in (6.17) can be rewritten as

$$q(\boldsymbol{x}) = b_0 \prod_{k\in\mathcal{V}} \frac{b_{c_k}(\boldsymbol{x}_{c_k})}{b_{s_k}(\boldsymbol{x}_{s_k})},$$

where $s_k = s_{k,\pi(k)}$ ($s_k = \emptyset$ for the root node) and b_0 is the normalization constant.

For notational convenience, we only prove the case when $D' = D$, i.e., the junction tree is globally consistent. More general cases follow similarly, by noting that any decision node that is assigned a fixed decision rule can be simply treated as a chance node.

First, we can rewrite $\mathrm{EU}(\boldsymbol{\delta}^*)$ as

$$\begin{aligned}
\mathrm{EU}(\boldsymbol{\delta}^*) &= \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \prod_{i \in D} \delta_i^*(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \\
&= b_0 \sum_{\boldsymbol{x}} \prod_{k \in \mathcal{V}} \frac{b_{c_k}(\boldsymbol{x}_{c_k})}{b_{s_k}(\boldsymbol{x}_{s_k})} \prod_{i \in D} \delta_i^*(x_i|\boldsymbol{x}_{\mathrm{pa}(i)}) \\
&= b_0 \sum_{\boldsymbol{x}} \left\{ \prod_{k \in \mathcal{C}} \frac{b_{c_k}(\boldsymbol{x}_{c_k})}{b_{s_k}(\boldsymbol{x}_{s_k})} \right\} \cdot \left\{ \prod_{k \in \mathcal{D}} \frac{b_{c_k}(\boldsymbol{x}_{c_k}) \delta_i^*(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(d_k)})}{b_{s_k}(\boldsymbol{x}_{s_k})} \right\} \\
&= b_0,
\end{aligned}$$

where the last equality follows by sequentially (along the reverse of the tree-order of the junction tree) applying the sum- and MEU- consistency condition (with $\epsilon \to 0^+$) in Lemma 7.1. To complete the proof, we just need to show that $\mathrm{EU}(\boldsymbol{\delta}) \leq b_0$ for any $\boldsymbol{\delta} \in \Delta$. Note that

$$\begin{aligned}
\mathrm{EU}(\boldsymbol{\delta})/b_0 &= \sum_{\boldsymbol{x}} \left\{ \prod_{k \in \mathcal{C}} \frac{b_{c_k}(\boldsymbol{x}_{c_k})}{b_{s_k}(\boldsymbol{x}_{s_k})} \right\} \cdot \left\{ \prod_{k \in \mathcal{D}} \frac{b_{c_k}(\boldsymbol{x}_{c_k}) \delta_{d_k}(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(d_k)})}{b_{s_k}(\boldsymbol{x}_{s_k})} \right\} \\
&\leq \prod_{k \in \mathcal{D}} \max_{\boldsymbol{x}_{s_k}} \sum_{\boldsymbol{x}_{c_k \setminus s_k}} \frac{b_{c_k}(\boldsymbol{x}_{c_k}) \delta_{d_k}(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(d_k)})}{b_{s_k}(\boldsymbol{x}_{s_k})} && \text{(C.11)} \\
&= \prod_{k \in \mathcal{D}} \max_{\boldsymbol{x}_{s_k}} \sum_{\boldsymbol{x}_{\{d_k, \mathrm{pa}(d_k)\} \setminus s_k}} \frac{b_{c_k}(x_{d_k}, \boldsymbol{x}_{\mathrm{pa}(d_k)}) \delta_{d_k}(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(d_k)})}{b_{s_k}(\boldsymbol{x}_{s_k})} && \text{(C.12)} \\
&\leq \prod_{k \in \mathcal{D}} \max_{\boldsymbol{x}_{s_k}} \sum_{\boldsymbol{x}_{\{d_k, \mathrm{pa}(d_k)\} \setminus s_k}} \frac{b_{c_k}(x_{d_k}, \boldsymbol{x}_{\mathrm{pa}(d_k)}) \mathbf{1}[x_{d_k} \in \arg\max b_{c_k}(x_{d_k}|\boldsymbol{x}_{\mathrm{pa}(d_k)})]}{b_{s_k}(\boldsymbol{x}_{s_k})} && \text{(C.13)} \\
&= 1, && \text{(C.14)}
\end{aligned}$$

where the inequality (C.11) is obtained by sequentially applying the sum-consistency condition in Lemma 7.1, and the equality (C.12) holds because $s_k \subseteq \mathrm{pa}(d_k)$. The inequality in (C.13) can be verified by simple algebra. Finally, the equality in (C.14) follows from the MEU-consistency condition. This completes the proof. □

Based to Theorem 7.1, we can easily establish person-by-person optimality of BP on an arbitrary junction tree.

Theorem 7.2. *Let* $(\mathcal{G}, \mathcal{C}, \mathcal{S})$ *be an arbitrary junction tree, and* $\boldsymbol{\delta}^*$ *the strategy given by MEU-BP in Algorithm 7.2 at its convergence. Then* $\boldsymbol{\delta}^*$ *is a policy-by-policy optimal strategy in the sense that* $\mathrm{EU}(\{\boldsymbol{\delta}_i, \boldsymbol{\delta}^*_{D\setminus i}\}) \leq \mathrm{EU}(\boldsymbol{\delta}^*)$ *for any* $i \in D$ *and* $\boldsymbol{\delta}_i$.

Proof. Following Theorem 7.1, one need only show that any junction tree is consistent for any single decision node $i \in D$; this is easily done by choosing a tree-ordering rooted at i's decision cluster. □