SUBTREE-BASED BOUNDS AND SIMULATION-BASED ESTIMATIONS FOR THE PARTITION FUNCTION

THÈSE N^{O} 3880 (2007)

PRÉSENTÉE LE 18 OCTOBRE 2007

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Laboratoire d'algorithmique

SECTION D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Mehdi MOLKARAIE

M.Sc. in Electrical Engineering, University of Tehran, Iran
et de nationalité iranienne

acceptée sur proposition du jury:

Prof. E. Telatar, président du jury
Prof. M. A. Shokrollahi , directeur de thèse
Prof. H.-A. Loeliger, rapporteur
Dr N. Macris, rapporteur
Dr H. Pfister, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, EPFL
2007

To Emre Telatar
and his colourful presence at EPFL

O, what men dare do!
What men may do!
What men daily do,
Not knowing what they do!

MUCH ADO ABOUT NOTHING

Contents

Abstract

Many different algorithms developed in statistical physics, coding theory, signal processing, and artificial intelligence can be expressed by graphical models and solved (either exactly or approximately) with iterative message-passing algorithms on the model.

One quantity of interest in these algorithms is the partition function. In graphical models without cycle (trees), the partition function can be computed efficiently by means of message-passing algorithms, like GDL or the sum-product algorithm. In contrast, when graphical models contain cycles, the computation of the partition function is in general intractable. Our contributions in this dissertation are: deriving deterministic upper and lower bounds on partition function, and developing methods to approximate this quantity with Monte Carlo methods.

Specifically, we obtain subtree-based upper and lower bounds which lead to theoretical results on optimality properties of the minimum entropy sub-tree and finally lead to a greedy algorithm. At last, we introduce and analyze a number of estimators that use Gibbs sampling to draw samples from different target distributions to estimate the value of the partition function. In one estimator, we demonstrate a novel strategy which combines Gibbs sampling and message-passing algorithms on trees.

Keywords: Probabilistic Inference, The Graphical Models, Generalized Distributive Law, Statistical Physics, Partition Function, Deterministic Bounds, Gibbs Sampling.

Résumé

La plupart des algorithmes dans les domaines de la physique statistique, de la théorie du codage, du traitement des signaux, et de l'intelligence artificielle peuvent être représentés par des modèles graphiques et être résolus par des algorithmes itératifs, de manière exacte ou approximative.

La quantité qui nous intéresse est la fonction de partition de ces algorithmes. Dans la famille des modèles graphiques sans cycles (arbres), cette fonction de partition peut être évaluée de façon efficace par des algorithmes itératifs. Par contre, quand les modèles graphiques possèdent des cycles, l'évaluation de cette fonction est en général algorithmiquement difficile. Notre contribution dans ce travail de thèse a été double: d'une part, nous avons obtenu des bornes déterministes pour la fonction de partition; d'autre part, nous avons développé des méthodes d'approximation de cette quantité basées sur les méthodes Monte-Carlo.

Plus spécifiquement, nous avons obtenu, à partir des sous-arbres du modèle, des bornes inférieures et supérieures conduisant à des résultats théoriques d'optimalité du sous-arbre d'entropie minimale ainsi qu'à un algorithme glouton. Nous avons introduit et analysé des estimateurs basés sur l'échantillonnage de Gibbs afin d'obtenir des échantillons à partir de différentes distributions de probabilités. Nous avons enfin développé un estimateur qui combine l'échantillonnage de Gibbs avec des algorithmes itératifs sur les arbres.

Mots clés: Inférence Probabiliste, Modèles Graphiques, Loi Distributive Généralisée, Physique Statistique, Fonction de Partition, Bornes déterministes, Echantillonnage de Gibbs.

Acknowledgements

I would like to express my sincere thanks to my thesis director Amin Shokrollahi for accepting me in his lab, his confidence in me, also for carefully proofreading my dissertation. I would like to express my deepest gratitude to Hans-Andrea Loeliger. Andi kindly accepted to be my coadvisor in the last stages of my phd. I was impressed and excited to see how quickly he grasped the idea behind my research, provided me with a great insight into it, and introduced me to the beautiful world of simulation-based methods. Andi's great intuition, his intellectual energy, and his enthusiasm for research make working with him a pleasant and unique experience. Many thanks to Emre Telatar for too many reasons to list here. I would only like to mention that I believe education is a holy profession for Emre. Besides, he has the gift to convey this sense to his students.

I am particularly indebted to Payam Pakzad for providing me with the guidance and support that I needed - especially in the early stages of my phd. Payam's contribution can be seen in every page of my thesis. I am also very happy that our collaboration formed the basis of a close friendship between us.

Special thanks to Justin Dauwels. I really enjoyed our many discussions when we shared an office in ETHZ, since then he has always been a valuable resource for my questions, proofreading my dissertation and articles, and pointing me out relevant work in Statistical Physics, Graphical models, and Information Theory. I wish all the good things that may happen in life for him and Shoko.

I am grateful to the members of my thesis committee Nicolas Macris and Henry Pfister for their insightful comments. Special thanks to Natascha Fontana, Evelyn Duperrex, and Simona Bucurescu for organizing my private defense.

I would like to thank Marie Ligier (ange gardien of EPFL) and J.-P. Wassmer for their help and support.

I thank Martin Wainwright and Terence Chan for their comments on earlier versions of my articles and pointers to related research.

I wish to thank J.-C. Chappelier, Patrick Thiran, Luciano Sbaiz, and J.P. Martin-Flatin for the wonderful time I had as their teaching assistant. Many thanks to J.-C. Chappelier, Patrick Thiran, and Olivier Leveque for correcting the French resume of my thesis.

At EPFL, I have also benefited from interactions and discussions with many faculty members. Especially, I would like to thank Nicolas Macris, Pierre Bremaud, Suhas Diggavi, Christina Fragouli, Kamiar Aminian, and Pascal Fua.

A big thank you to Mahdi Cheraghchi for sharing with me his deep knowledge in algorithms, computer programming, and LaTeX.

Here is my *Thank You* list to my family, friends, and colleagues. If I have forgotten your name, please accept my apologies and know that you are written in my mind. And if you ever meet any of these people, invite them for a drink! Adeline Maerten, Alejandra Garcia Martinez, Alex Alahi, Ali Kafi (for the many lessons of life and his passion for Persian language), Ali and Maryam Darbandi, Ali Salehi, Ali Shahrokni, Ali Elahi, Amin Karbasi, Amir F. Dana, Amir Esfandiari (le joli coeur, stay in peace!), Aoife Hegarty, Arash Abadi, Arash and Marjan Salarian, Arash Hooshmand, Audrey Robert, Azadeh Faridi, Azadeh Farshchi, Azad and Azin Koliji (the lovely couple), Babak Arsalanpoor (a friend who is my brother and a brother who is my friend), Babak Sedghi, Bahram Moobed, Basira Salehi, Bel Garcia, Benoit Le Callennec, Bertrand Ndzanandzana, Cecilia and Simon Jolliet, Christian Schlatter, Christine Neuberg, Chantal Michoud, Charlotte Vandenberghe, Damir Laurenzi (computer wizard), Daniel Hoesli, David Hoover, Deanna Lund, Delphine Claret, Dinkar Vausudevan, Elitza Maneva, Emad Hajimiri, Emad Zand, Emilio Casanova, Etienne Perron, Frederique Oggier, Francesco Zilliani, Gerald Maze, Giovanni Cangiani (computer wizard), Golmehr Ashrafpoor, Hamid

Molkaraie, Hareesh P. Veettil, Helena Grillon, Hooman Dejnabadi, Hooman and Faranak Kasehchi, Hung and Tam Nguyen, Isabelle Taverner (and the good old days, je dois dire), Irina O. Pakzad, Ivana Podnar, Jerome Poulin, Jian Young, Jamshid Nazi and Alireza Molkaraie, Jim Pugh, Jurgen Ehrensberger, Kamran Lynda and Daris Hooshagpour (the lovely famiy), Kamiyar Kazemi, Karin Ramseier, Lorenz Minder (a mathematician of great wit), Luca Vacchetti, Mahkameh and Hassan Nasrollahzadeh, Mana Mahaseni, Manfred Housewirth, Marcelo Fernandez, Margarita Miranda Lopez, Mariam Momenzadeh, Marisa Marciano Wynn, Maryam Mostaedi, Melanie Althaus, Michele Wigger, Mila Tevez, Mireille Clavien, Mona and Yalda Nasroli, Matthias Frey, Minoo Fakharpour, Naghmeh Ghasemzadeh, Nasim Mozaffari, Nastaran Fatemi (and her hot chocolates), Natalia Miliou, Natalie Meystre, Natalie Sautter, Neda and Ali Movaghar, Negar Kiavash, Nicolas and Sylvie Elsig, Nima and Marjan Nilipoor, Nona Ziaei, Omid Etesami (the sharpest guy I know), Paola Bonfanti, Pablo and Ivana de Heras Ciechomski, Parisa Siamak Kiana and Kourosh Okhovat, Pegah Shokati, Philippe Cudre-Mauroux, Prakash Mathur, Rana Rezakhaniha, Regis, Renault John, Reza and Helia Kasravi, Renaud Ott, Robin Evans, Roman Schmidt, Ronan Boulic, Sahar Hosseinian, Saleh Tabandeh, Sam Kalantari, Samuel Sonderegger, Sandrine Allenbach, Sandro Saitta, Sanket Dusad, Satish Babu Korada, Shahnaz and Masoud Motaedi, Shrinivas Kudekar, Sibi Raj B., Soheil Mohajer, Sorayya Sadeghi, Stefan Moser (and his great sense of humour), Stephan Tinguely, Thierry Michellod (a real gentleman), Tim Van Pelt (and our adventures in Hollandhouse), Tobias Koch, Tooraj and Ehsan Arvajeh, Vera Batutina, Vincent Lepetit, Vishwambhar Rathi, Wei Wang, Wendy Gauthier, Wojciech Galuba, Xuan Zhang, Yongluan Zhou, and Zerrin Celebi.

I am grateful to my mother Fereshteh, my father Nezam, my sister Moojan, and my aunt Farah for their love and support throughout my life.

Chapter 1

Introduction

1.1 Motivation

The central topic in this dissertation is Z, the **partition function**. The most obvious role of this quantity is normalizing a distribution. The partition function is a quantity of interest in a wide variety of algorithms developed in different fields, including statistical physics, approximate inference, Bayesian model comparison, large deviations bounds, and combinatorial enumeration. In statistical physics Z typically carries information about all the thermodynamic properties of a system. In Bayesian model comparison Z is a crucial quantity and can be interpreted as the likelihood of observing a set of data given the model. In the context of large deviations, Z specifies (exponential) error rates.

Graphical models provide an attractive framework to formulate problems common to all these algorithms and to solve them with efficient iterative message-passig algorithms described on such models. For graphical models without cycles, Z can be computed efficiently with message-passing algorithms. However, when the graphical model contains cycles, the computation of the partition function is in general intractable. Therefore, approximating and obtaining low-complexity bounds on this quantity is important.

Our contributions in this dissertation are twofold

1. Subtree-based deterministic upper and lower bounds on Z.

2. Simulation-based estimation of Z with Monte Carlo methods.

The methods we use are also twofold. In the first part, we use iterative message-passing algorithms on sub-trees of a given graphical model to derive deterministic

upper and lower bounds on the partition function. In the second part, we use Monte Carlo methods to estimate the partition function and propose various estimators. In one estimator, we demonstrate how to combine Monte Carlo methods and message-passing algorithms on trees.

Next we present an overview of this dissertation.

1.2 Overview

In Chapter 2 we give a brief introduction of a probabilistic inference problem, graphical models, and the generalized distributive law. A probabilistic inference problem is concerned with computing the partition function and the marginals of a global function of many variables. The global function often factors as a product of local functions, each of which depends on a subset of the variables.

Consider a set $\{X_1, X_2, \ldots, X_N\}$ of N discrete random variables. Suppose x_i represents the possible realizations of X_i and $\mathbf{x}$ represents $\{x_1, x_2, \ldots, x_N\}$. Suppose $R_1, R_2, \ldots, R_M$ are subsets of $\{1, 2, \ldots, N\}$ and $\mathcal{R} = \{R_1, R_2, \ldots, R_M\}$ is a collection of subsets of the indices of the random variables. Also suppose $p(\mathbf{x})$, the global probability function, factors into a product of non-negative and finite local kernels as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{R \in \mathcal{R}} \alpha_R(\mathbf{x}_R), \tag{1.1}$$

where each local kernel $\alpha_R(\mathbf{x}_R)$ is a function of the variables whose indices appear in R and Z is the partition function. In a probabilistic inference problem we are interested in computing the partition function Z, defined by

$$Z = \sum_{\mathbf{x}} \prod_{R \in \mathcal{R}} \alpha_R(\mathbf{x}_R), \tag{1.2}$$

and marginal densities $p_R(\mathbf{x}_R)$, defined by

$$p_R(\mathbf{x}_R) = \sum_{\mathbf{x} \setminus \mathbf{x}_R} p(\mathbf{x}). \tag{1.3}$$

The factorization in a probabilistic inference problem can be visualized with a graphical model. A graphical model is a family of probability distributions that factorize according to the structure of an underlying graph. They are used and studied in various applied statistical and computational fields such as signal processing, information theory, statistical physics, and artificial intelligence. The two

most common forms of graphical models are *directed graphical models* and *undirected graphical models*. Here we consider undirected graphical models and focus on graphical models defined in terms of junction graphs/trees. According to Theorem 1.1, for any probabilistic inference problem there is always a junction graph representation [AM01].

Theorem 1.1. *For any collection $\mathcal{R} = \{R_1, R_2, \ldots, R_M\}$ of subsets of $\{1, 2, \ldots, n\}$ there is a junction graph for $\mathcal{R}$.*

Having a junction graph representation of a probabilistic inference problem allows us to use a general and low-complexity message passing algorithm, called the generalized distributive law (GDL), to solve or approximate a probabilistic inference problem. Some of the best (approximate) decoding algorithms (i.e., for decoding Turbo codes and low-density parity-check codes) involve message passing methods on graphical models. When the problem can be organized into a junction tree, according to Theorem 2.9, GDL gives an exact message passing solution to the probabilistic inference problem .

In Chapter 3, we give a brief introduction of statistical physics and its connections to the probabilistic inference problem. We give definitions of the partition function and free energy.

Consider $\{X_1, X_2, \ldots, X_N\}$ a set of N identical particles and suppose $\mathbf{x}$ represents the state $\{x_1, x_2, \ldots, x_N\}$. Also suppose that each state has energy $E(\mathbf{x})$. In thermal equilibrium, the probability of a state is given by the Boltzmann distribution as

$$p(\mathbf{x}) = \frac{1}{Z} e^{-E(\mathbf{x})}. \tag{1.4}$$

The partition function is defined as

$$Z = \sum_{\mathbf{x}} e^{-E(\mathbf{x})}. \tag{1.5}$$

And the free energy is defined as

$$F_H = -\ln(Z). \tag{1.6}$$

We explain variational methods to compute the free energy of a system and approximations to these methods. We describe one such approximation, namely the Bethe-Kikuchi approximation to the variational free energy with respect to $\mathcal{G}$ and denote by $\tilde{H}^{\mathcal{G}}_{BK}(q)$. In Theorem 1.2 we show that for a given collection of random

variables the Bethe-Kikuchi approximation on a junction tree representation is an upper bound to the exact value of the entropy function denoted by $H(q)$.

Theorem 1.2. *Let $q(\mathbf{y})$ be an arbitrary probability distribution on $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_N\}$ and $\mathcal{T}$ any collection of subsets of the index set $\{1, 2, \cdots, N\}$ for which a junction tree exists. Then the following bound holds*

$$H(q) \leq H_{BK}^{\mathcal{T}}(q). \tag{1.7}$$

The message passing decoding algorithms on graphical models are deeply related to different energy approximations developed in statistical physics. We explain how the Bethe-Kikuchi approximation to the free energy is related to GDL on junction graphs.

In Chapter 4, we introduce deterministic bounds on the partition function and derive subtree-based upper and lower bounds. For a probabilistic inference problem defined by $\mathcal{R} = \{R_1, R_2, \ldots, R_M\}$, we suppose $\mathcal{R}_T \subset \mathcal{R}$ has a junction tree representation with probability distribution $q_T(\mathbf{x})$ and partition function Z_T. Therefore, rewrite $p(\mathbf{x})$ as

$$p(\mathbf{x}) = \frac{Z_T}{Z}\, q_T(\mathbf{x}) \prod_{R \in \mathcal{R} \setminus \mathcal{R}_T} \alpha_R(\mathbf{x}_R). \tag{1.8}$$

With this reformulation we derive the following upper and lower bounds

$$\sum_{R \in \mathcal{R} \setminus \mathcal{R}_T} \sum_{\mathbf{x}} q_T(\mathbf{x}) \ln \alpha_R(\mathbf{x}_R) \leq \ln(\frac{Z}{Z_T}) \leq \sum_{R \in \mathcal{R} \setminus \mathcal{R}_T} \sum_{\mathbf{x}_R} p(\mathbf{x}_R) \ln \alpha_R(\mathbf{x}_R). \tag{1.9}$$

We denote the lower bound computed using q_T by $\mathcal{L}_{q_T}$ and in Theorem 1.3 we present an inequality to compare lower bounds derived from different sub-junction trees according to their entropies.

Theorem 1.3. *Consider $\mathcal{R}_1$ and $\mathcal{R}_2$, subsets of $\mathcal{R}$ with junction tree representations. Suppose that $q_1(\mathbf{x})$, Z_1, $q_2(\mathbf{x})$, and Z_2 denote the global probability distributions and the partition functions over $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively. Without loss of generality suppose $H(q_1) \leq H(q_2)$, then the following inequality holds*

$$\mathcal{L}_{q_2} \leq \mathcal{L}_{q_1} + \min\big(D(q_1||\overline{q}_1) - D(q_2||q_1), D(q_1||q_2) + D(q_1||\overline{q}_2)\big), \tag{1.10}$$

where $\overline{q}_1$ and $\overline{q}_2$ denote the global probability distributions on $\mathcal{R} \setminus \mathcal{R}_1$ and $\mathcal{R} \setminus \mathcal{R}_2$ respectively.

Using the results of Theorem 1.3, we study the properties of the lower bound obtained from the minimum entropy sub-tree and show that this sub-tree has some optimality properties.

In Chapter 5, by simplifying the upper and lower bounds obtained in Chapter 3, we present new and low-complexity bounds for the partition function. In a theorem we prove that the new bounds only need to be optimized on maximal subjunction trees of a given junction graph. We propose a greedy algorithm to find such a sub-tree and compute the bounds. We study some properties of our greedy algorithm and apply it to two-dimensional grids. Simulation results are reported at the end of the chapter.

In Chapter 6, we use Monte Carlo methods to estimate the value of the partition function. Monte Carlo methods were first used by Metropolis, et al. in simulation of liquids in statistical physics [Fre04]. The term Monte Carlo was also invented by Metropolis because of the similarity of statistical simulation to games of chance.

Sampling from a high-dimensional distribution (which is known up to a normalizing constant) is the central problem in Monte Carlo methods. Here we focus on Gibbs sampling, one of the simplest and widely used Monte carlo methods, and propose various estimators that use Gibbs sampling to estimate the partition function of a given probabilistic inference problem. We propose three types of estimators.

Let $\mathbf{X}$ stand for $\{X_1, X_2, \ldots, X_N\}$ and let $\mathcal{X}$ represent the sample space, assume that $p(\mathbf{x})$ has the same factorization as in (1.1) and that the local kernels are positive.

Let us define

$$\alpha(\mathbf{x}) \triangleq \prod_{R \in \mathcal{R}} \alpha_R(\mathbf{x}_R), \tag{1.11}$$

$$\gamma \triangleq \frac{1}{Z}. \tag{1.12}$$

Then, we can rewrite (1.1) as

$$p(\mathbf{x}) = \frac{1}{Z}\alpha(\mathbf{x}) = \gamma\alpha(\mathbf{x}). \tag{1.13}$$

With these assumptions we can define Z as

$$Z = \sum_{\mathbf{x} \in \mathcal{X}} \alpha(\mathbf{x}). \tag{1.14}$$

In (6.7), we propose an estimator that uses samples drawn according to the probability distribution $p(\mathbf{x})$ and prove that

Theorem 1.4. *If $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(K)}$ are i.i.d. samples drawn with distribution $p(\mathbf{x})$ from $\mathcal{X}$, $\tilde{\gamma}$ is an unbiased estimator for γ, where*

$$\tilde{\gamma} = \frac{1}{K|\mathcal{X}|} \sum_{k=1}^{K} \frac{1}{\alpha(\mathbf{x}^{(k)})}. \tag{1.15}$$

Suppose $\mathbf{z}$ is a subset of $\mathbf{x}$. In (6.28) propose an estimator for the partition function that uses samples drawn according to $p(\mathbf{z})$ with sample space $\mathcal{Z}$.

Theorem 1.5. *If $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(K)}$ be i.i.d. samples drawn with distribution $p(\mathbf{z})$ from $\mathcal{Z}$, $\tilde{\gamma}$ is an unbiased estimator for γ, where*

$$\tilde{\gamma} = \frac{1}{K|\mathcal{Z}|} \sum_{k=1}^{K} \frac{1}{\alpha(\mathbf{z}^{(k)})}. \tag{1.16}$$

We show how a clever choice the variables in $\mathbf{z}$ enables us to use the ideas from message passing algorithms on trees for this estimator. Finally, in (6.34), we propose an estimator that uses samples drawn uniformly from $\mathcal{X}$.

Theorem 1.6. *If $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(K)}$ be i.i.d. samples drawn uniformly from $\mathcal{X}$, $\tilde{Z}$ is an unbiased estimator for Z, where*

$$\tilde{Z} = \frac{|\mathcal{X}|}{K} \sum_{k=1}^{K} \alpha(\mathbf{x}^{(k)}). \tag{1.17}$$

We estimate the value of the partition function of two-dimensional grids with different estimators. We show how to choose the variables in $\mathbf{z}$ with examples of grids with different sizes and report the simulation results.

Chapter 2

The Probabilistic Inference Problem and the GDL Algorithm

In this chapter we give an introduction to the probabilistic inference problem, graphical models, and the generalized distributive law. We discuss two main tasks in a probabilistic inference problem, namely computing the partition function and the set of desired marginals of a product function. These tasks lie at the heart of many algorithms in coding theory, statistical physics, signal processing, and machine learning. Graphical models provide a way to view all these algorithms in a common framework [Dau06], [WJ93].

We focus on graphical models defined in terms of junction graphs and describe a general procedure, called the generalized distributive law, that gives a message passing solution for a probabilistic inference problem on junction trees.

The outline of this chapter is as follows. In Section 2.1, we introduce the probabilistic inference problem. In Section 2.2, we show how graphical models, particularly junction graphs, are used to visualize a probabilistic inference problem. In Section 2.3, we review the generalized distributive law.

2.1 The Probabilistic Inference Problem

Suppose a global function defined over several random variables, e.g., a probability mass function, factors as a product of a series of non-negative and finite local kernels. Also suppose each kernel depends on a subset of the set of all random variables. The goal is to compute the partition function and the marginals of the global function according to those subsets.

More formally, consider a set $\{X_1, X_2, \ldots, X_N\}$ of N discrete random variables each of which taking their values in a finite set A, e.g. $A = \{0, 1, 2, \ldots, a-1\}$.

Let x_i represent the possible realizations of X_i and let $\mathbf{x}$ stand for $\{x_1, x_2, \ldots, x_N\}$. Suppose $R_1, R_2, \ldots, R_M$ are subsets of $\{1, 2, \ldots, N\}$ and $\mathcal{R} = \{R_1, R_2, \ldots, R_M\}$ is a collection of subsets of the indices of the random variables X_1 through X_N. Let us also suppose that $p(\mathbf{x})$, the joint probability mass function, factors into a product of non-negative and finite local kernels as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{R \in \mathcal{R}} \alpha_R(\mathbf{x}_R), \tag{2.1}$$

where each local kernel $\alpha_R(\mathbf{x_R})$ is a function of the variables whose indices appear in R and Z is the *partition function*, also known as the *global normalization constant* whose role is simply normalizing the probability distribution.

Let us define

$$\alpha(\mathbf{x}) \triangleq \prod_{R \in \mathcal{R}} \alpha_R(\mathbf{x}_R). \tag{2.2}$$

With the above definition, we can rewrite (2.1) as

$$p(\mathbf{x}) = \frac{1}{Z} \alpha(\mathbf{x}). \tag{2.3}$$

In a *probabilistic inference* problem we are interested in computing

1. The partition function Z, defined by

$$Z = \sum_{\mathbf{x}} \prod_{R \in \mathcal{R}} \alpha_R(\mathbf{x}_R) = \sum_{\mathbf{x}} \alpha(\mathbf{x}). \tag{2.4}$$

2. Marginal densities $p_R(\mathbf{x}_R)$, defined by

$$p_R(\mathbf{x}_R) = \sum_{\mathbf{x} \setminus \mathbf{x}_R} p(\mathbf{x}), \tag{2.5}$$

   where $\mathbf{x} \setminus \mathbf{x}_R$ means all variables except for $\mathbf{x}_R$

The problem of finding the marginal densities is usually called the MPF (marginalize a product function) problem.

Both computations involve a summation over all possible combinations for the values of a large number of random variables and in general grow exponentially with the number of variables.

An efficient way to solve (or find approximate solutions for) a probabilistic inference problem is to exploit the factorization of $\alpha(\mathbf{x})$ by graphical models and use message passing algorithms on the model.

2.2 Graphical Models

The graphical model framework provides a suitable tool to visualize and formulate a probabilistic inference problem. This framework allows us to visualize the factorization in (2.1) with a graph. Having such a graphical representation for a probabilistic inference problem, we can express the computations in (2.4) and (2.5) with operations on this graph [Mur01], [WJ93].

There are many graphical models in literature such as Markov random fields, factor graphs [KFL01], and Forney-style factor graphs (normal graphs) [For01]. We focus on graphical models defined in terms of junction graphs [AM00]. Our results can easily be re-expressed with other graphical models. (For example in [WF01, Appendix], the authors show how to convert junction graphs and factor graphs to Markov random fields)

2.2.1 Junction Graphs

Definition 2.1. A *junction graph* is an undirected graph $\mathcal{G} = (V, E, L)$ where each vertex $v \in V$ and each edge $e \in E$ is labeled with a subset of $\{1, 2, \ldots, n\}$. We denote the set of labels for vertex $v \in V$ and edge $e \in E$ by $L(v)$, and $L(e)$ respectively. The labels on the edges must be a subset of the labels of their corresponding vertices. In other words, if an edge e joins the vertices v_1 and v_2, we write $e = (v_1, v_2)$ and require that

$$L(e) \subseteq L(v_1) \cap L(v_2). \tag{2.6}$$

Furthermore, we require that the induced subgraph of $\mathcal{G}$ consisting only of the vertices and edges which contain a particular label, must be a tree.

Figs. 2.1-a and 2.1-b show two junction graphs.

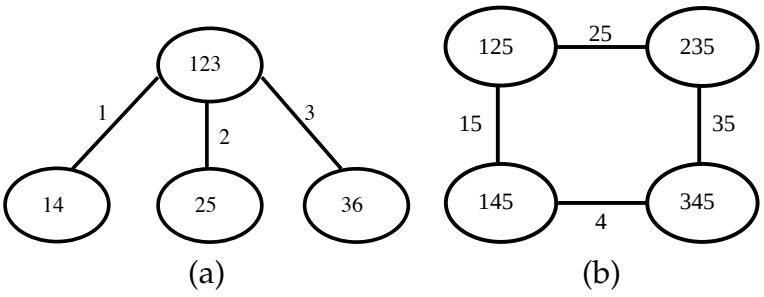

Figure 2.1: Two Junction Graphs

For a probabilistic inference problem, we say that $\mathcal{G} = (V, E, L)$ is a junction graph for $\mathcal{R}$, if $\{L(v_1), L(v_2), \ldots, L(v_M)\} = \mathcal{R}$. We denote vertex labels by $L(v_i)$ and edge labels by $L(v_i, v_j)$ Fig. 2.2 shows a junction graph for $\mathcal{R} = \big\{\{1,2\}, \{1,3\}, \{1,4\}, \{2,3,4\}\big\}$.

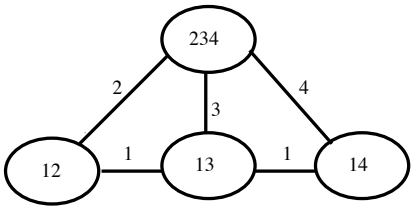

Figure 2.2: Junction Graph Representing $\mathcal{R} = \big\{\{1,2\}, \{1,3\}, \{1,4\}, \{2,3,4\}\big\}$

Theorem 2.2. *For any collection $\mathcal{R} = \{R_1, R_2, \ldots, R_M\}$ of subsets of $\{1, 2, \ldots, n\}$ there is a junction graph for $\mathcal{R}$ [AM01, Theorem 4].*

Proof: Begin with a complete graph $\mathcal{G}$ with vertex set $V = \{v_1, v_2, \ldots, v_m\}$, vertex labels $L(v_i) = R_i$ and edge labels $L(v_i, v_j) = R_i \cap R_j$. For each $k \in \{1, 2, \ldots, n\}$, let $\mathcal{G}_k = (V_k, E_k)$ be the subgraph of $\mathcal{G}$ consisting of those vertices and edges whose labels contain k. Clearly $\mathcal{G}_k$ is a complete graph, since if $k \in L(v_i) = R_i$, then $k \in L(v_i) \cap L(v_j) = R_i \cap R_j$. Now let $\mathcal{T}_k$ be any spanning tree of $\mathcal{G}_k$ and delete k from the labels of all edges in E_k except those in $\mathcal{T}_k$. The resulting labeled graph is a junction graph for $\mathcal{R}$. ∎

Using the fact that a complete graph with m vertices has exactly m^{m-2} spanning trees we can conclude that if m_i denotes the number of sets R_j such that $i \in R_j$ then the number of junction graphs for $\mathcal{R}$ is $\prod_{i=1}^{n} m_i^{m_i-2}$. Therefore, in general there might be many junction graphs representing the same inference problem.

Example 2.3. *For $\mathcal{R} = \big\{\{1,2,5\}, \{2,3,5\}, \{3,4,5\}, \{1,4,5\}\big\}$, there are in total sixteen junction graph representations. Fig. 2.3 shows four such junction graphs.*

According to Theorem 2.2, for any probabilistic inference problem there is always a junction graph representation but *not* necessarily a *junction tree* representation.

Definition 2.4. A *junction tree* is an undirected tree $\mathcal{T} = (V, E, L)$ where each vertex and each edge have labels, denoted by $L(v)$, and $L(e)$ respectively, such that for any label the induced subgraph of $\mathcal{T}$ consisting only of the vertices and edges which contain that label, is connected.

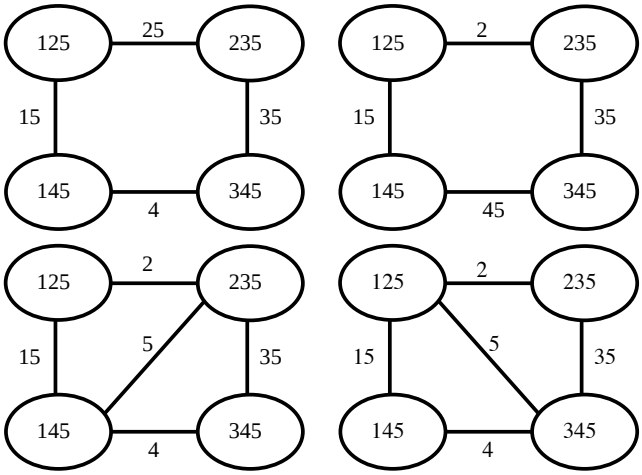

Figure 2.3: Four Junction Graphs for $\big\{\{1,2,5\},\{2,3,5\},\{3,4,5\},\{1,4,5\}\big\}$

Therefore, Fig. 2.1-a shows a junction tree for $\mathcal{R} = \big\{\{1,2,3\},\{1,4\},\{2,5\},\{3,6\}\big\}$.

It is easy to decide whether for $\mathcal{R} = \{R_1, R_2, \ldots, R_M\}$ a junction tree exists or not. The key is the *local domain graph* G_{LD}, which is a weighted complete graph with M vertices $v_1, v_2, \ldots, v_M$, one for each $R_i \in \mathcal{R}$, with the weight of the edge $e_{i,j}$ defined by $w_{i,j} = |R_i \cap R_j|$. Denote by w_{max} the weight of the maximal-weight spanning tree of G_{LD}. Also define

$$w^* = \sum_{i=1}^{M} |R_i| - N \tag{2.7}$$

where N is the number of variables (indices).

Theorem 2.5. *$w_{max} \leq w^*$, with equality if and only if there is a junction tree. If $w_{max} = w^*$, then any maximal-weight spanning tree of G_{LD} is a junction tree [AM00, Theorem 4.1].*

Example 2.6. *For $\mathcal{R} = \big\{\{1,2,3\},\{1,4\},\{2,5\},\{3,6\}\big\}$, it is easy to see that $w^* = (3+2+2+2)-6 = 3$ and the weight of a maximal spanning tree is also 3. Therefore according to Theorem 2.5, for $\mathcal{R} = \big\{\{1,2,3\},\{1,4\},\{2,5\},\{3,6\}\big\}$ a junction tree representation exists, Fig 2.1-a.*

Example 2.7. *For $\mathcal{R} = \big\{\{1,2,3\},\{2,3,4\},\{3,4,5\}\big\}$, we can see that $w^* = (3+3+3)-5 = 4$ and w_{max} is also 4. Therefore according to Theorem 2.5, a junction tree representation exists for $\mathcal{R}$.*

Example 2.8. *For* $\mathcal{R} = \big\{\{1,2\},\{1,3\},\{1,4\},\{2,3,4\}\big\}$*, we can verify that* $w_{max} = (3+2+2+2)-4 = 5$ *and the weight of a maximal spanning tree is* 3*. Therefore according to Theorem 2.5, a junction tree representation does not exist.*

We will see in Section 2.3 that if junction tree representation for a probabilistic inference problem exists, there is a message passing algorithm to compute the partition function and the marginal densities efficiently. This is the topic of the next section.

2.3 The Generalized Distributive Law (GDL)

The generalized distributive law (GDL) is an iterative message passing algorithm, described by its *messages* and *beliefs*, to solve the probabilistic inference problem on a junction graph [AM00]. In [SS90], the authors describe a general framework similar to GDL, here we prefer to follow the more convenient and recent framework presented in [AM00].

GDL is a local message passing algorithm. Roughly speaking, GDL uses the *distributive law* of the product operation over summation to reduce the complexity of the calculations.

On a junction graph $\mathcal{G}$, if vertices u and v are connected by an edge e, the message sent from u to v is a function of the variables whose indices are on e, and is denoted by $m_{u,v}(\mathbf{x}_{L(u,v)})$, where $L(u,v)$ denotes the labels on $e=(u,v)$.

When a particular message $m_{u,v}(\mathbf{x}_{L(u,v)})$ is updated, the following rule is used

$$m_{u,v}(\mathbf{x}_{L(u,v)}) = \sum_{\mathbf{x}_{L(u)\setminus L(u,v)}} \alpha_u(\mathbf{x}_{L(u)}) \prod_{u'\in N(u)\setminus v} m_{u',u}(\mathbf{x}_{L(u',u)}), \tag{2.8}$$

where $N(u)$ denotes the neighbors of u.

In words, the message from u to v is formed by the product of the local kernel at u with all the messages u has received from its neighbors other than v when marginalized properly.

Initially, all messages are defined to be identically 1 :

$$m_{u,v}(\mathbf{x}_{L(u,v)}) \equiv 1, \quad \text{for all } e \in E. \tag{2.9}$$

The beliefs on vertices and edges are denoted by $b_v(\mathbf{x}_{L(v)})$ and $b_e(\mathbf{x}_{L(e)})$, respectively. At any stage of the algorithm, the beliefs can be computed as

$$b_v(\mathbf{x}_{L(v)}) = \frac{1}{Z_v}\alpha_v(\mathbf{x}_{L(v)}) \prod_{u\in N(v)} m_{u,v}(\mathbf{x}_{L(u,v)}), \tag{2.10}$$

$$b_e(\mathbf{x}_{L(e)}) = \frac{1}{Z_e} m_{u,v}(\mathbf{x}_{L(e)}) m_{v,u}(\mathbf{x}_{L(e)}), \tag{2.11}$$

where Z_v and Z_e are the local normalizing constants, defined as

$$Z_v = \sum_{\mathbf{x}_{L(v)}} \alpha_v(\mathbf{x}_{L(v)}) \prod_{u\in N(v)} m_{u,v}(\mathbf{x}_{L(u,v)}), \tag{2.12}$$

$$Z_e = \sum_{\mathbf{x}_{L(e)}} m_{u,v}(\mathbf{x}_{L(e)}) m_{v,u}(\mathbf{x}_{L(e)}). \tag{2.13}$$

In words, when normalized properly, the belief on vertex v is the product of its local kernel with all the messages it has received from its neighbors, and the belief on edge e, the edge between v and u, is the product of the messages passed in both directions on e, i.e. from u to v and from v to u.

Theorem 2.9. *On a junction tree the beliefs in (2.10) and (2.11) converge to the exact desired local marginal probabilities after a finite number of steps, [AM00, Theorem 3.1], [Pea88].*

Therefore, the GDL algorithm gives a message passing solution to the MPF problem, stated in Section 2.1, when the sets $\{R_1, R_2, \ldots, R_M\}$ can be organized into a junction tree.

Theorem 2.10. *On a junction tree defined by (2.1) $p(\mathbf{x})$ factors as the product of local probabilities on the vertices over the product of local probabilities on the edges [Cow98]*

$$p(\mathbf{x}) = \frac{\prod_{v\in V} p_v(\mathbf{x}_{L(v)})}{\prod_{e\in E} p_e(\mathbf{x}_{L(e)})}. \tag{2.14}$$

In this case, the entropy decomposes as the sum of the entropies of the vertices minus the sum of the entropies on the edges. This gives (see Theorem 3.2).

$$H(p) = \sum_{v\in V} H(p_v(\mathbf{x}_{L(v)})) - \sum_{e\in E} H(p_e(\mathbf{x}_{L(e)})). \tag{2.15}$$

Similarly, the global normalization constant Z can be expressed in terms of the local normalization constants as follows

$$Z = \frac{\prod_{v\in V} Z_v}{\prod_{e\in E} Z_e}. \tag{2.16}$$

Therefore, if $\mathcal{G}$ is a tree, there is an efficient algorithm to compute the marginals of $p(\mathbf{x})$, the entropy of $p(\mathbf{x})$, and the partition function. If $\mathcal{G}$ is not a tree, the above algorithm is not guaranteed to give the exact solution or even to converge, although empirically it performs very well in various applications such as decoding error-correcting codes.

New theoretical results show that there is a close connection between the message passing algorithms on graphical models and certain approximations to the energy function in statistical physics, thus establishing a link between message passing algorithms and approximations from physics. This is the topic of the next chapter.

We finish this section by stating the following remarks.

Remark 2.11. Similar to GDL (but with a slightly different notation), there is a local message passing algorithm, called the sum-product algorithm, that solves the probabilistic inference problem on factor graphs, see [KFL01]. The results developed in our setting can be translated into an equivalent setting using factor graphs and be solved by the sum-product algorithm.

Remark 2.12. In order to compute the beliefs at all vertices, GDL can be scheduled in a fully parallel manner. In this case, at every iteration, every belief is updated, and every message is computed and sent simultaneously. Alternatively, the GDL can be scheduled fully serially. In this case, each message is sent only once, and each belief is computed only once. A vertex sends a message to a neighbor when, for the first time, it has received messages from all its neighbors. See [AM00, Section 3], for more details on parallel, serial, and hybrid GDL scheduling.

Remark 2.13. In Section 2.2.1, we stated that for a valid junction graph $\mathcal{G}$, it is required that the induced subgraph of $\mathcal{G}$ consisting only of the vertices and edges which contain a particular label be a tree. In [YFW05], this concept has been generalized to introduce valid *region-based* approximations and the *generalized belief propagation algorithm*. The junction graph introduced in this chapter is a special case of the *junction graph method* described in [YFW05, Appendix A].

Remark 2.14. In [PA04], the authors develop a measure-theoretic version of GDL where the desired marginals are viewed as conditional expectations of a product of random variables.

Chapter 3

Connection to Statistical Physics

In this chapter, we study the connection between the probabilistic inference problem, stated in Section 2.1, and statistical physics. The connection has been noted by several authors and in different contexts. Also many methods from statistical physics have been used for probabilistic inference, information theory, combinatorial optimization, and vice versa [MM06], [Nea93]. The recent paper by Yedidia et al [YFW05] showed that there is a close connection between message passing algorithms in probabilistic inference and certain approximations in statistical physics.

The outline of the chapter is as follows. In Section 3.1 we review some basic ideas from statistical mechanics, especially the Boltzmann distribution, partition function, and free energy. Variational free energy, in Section 3.2, suggests a method for computing the free energy. In Section 3.2 we also discuss the Bethe-Kikuchi approximation to the free energy and its connection to the probabilistic inference problem. In Section 3.3 we prove that if an arbitrary probability distribution has the same marginals as the Boltzmann distribution on a junction tree, its entropy is upper bounded by the entropy of the Boltzmann distribution.

3.1 Some Statistical Mechanics

Consider $\{X_1, X_2, \ldots, X_N\}$ a set of N identical particles each of which taking their states in a finite set A with a elements as $A = \{0, 1, 2, \ldots, a-1\}$. Let x_i represent the state of the ith particle and let the overall state $\mathbf{x}$ stand for $\{x_1, x_2, x_3, \ldots, x_N\}$. Also suppose that each state of the system has a corresponding energy $E(\mathbf{x})$.

A well-known result from statistical physics says that in thermal equilibrium, the probability of a state will be given by *Boltzmann distribution* as follows

$$p(\mathbf{x}) = \frac{1}{Z(T)} e^{-E(\mathbf{x})/T}, \tag{3.1}$$

where T is the temperature and $Z(T)$ is the corresponding partition function.

As T increases, i.e., at high temperature, the Boltzmann distribution is close to uniform, while as T decreases, i.e., at low temperature, the system is called to be in *disordered* configuration and typically finds itself in low energy states.

In our case, T is only a scale for the units in which one measures energy. We set $T = 1$ and define the partition function as

$$Z = \sum_{\mathbf{x} \in \mathcal{X}} e^{-E(\mathbf{x})}, \tag{3.2}$$

where $\mathcal{X}$ is the space of all possibles states $\mathbf{x}$ of a system.

The *free energy* of a system is defined as

$$F_H = -\ln(Z). \tag{3.3}$$

The free energy is of fundamental importance in statistical mechanics because most macroscopic thermodynamic properties of a system follow from differentiating this quantity and many problems can be recast as free energy computation.

Exact calculation of the partition function and the free energy of a system is computationally intractable due to the exponential number of terms in (3.2), therefore physicists have developed a number of techniques which give good approximations to these quantities. One is the *variational free energy* technique.

3.2 Variational Free Energy

Suppose that $p(\mathbf{x})$ is the Boltzmann distribution and $q(\mathbf{x})$ represents a *trial* probability distribution which represents the probability that the system is in state $\mathbf{x}$. The corresponding *variational free energy* (also called *Gibbs free energy*) is defined by

$$\tilde{F}(q) = \bar{E}(q) - H(q), \tag{3.4}$$

where $\bar{E}(q)$ is the *variational average energy* and $H(q)$ is the *variational entropy* defined by

$$\bar{E}(q) = \sum_{\mathbf{x}} q(x) E(x), \tag{3.5}$$

$$H(q) = -\sum_{\mathbf{x}} q(x) \ln q(x). \tag{3.6}$$

It follows from the above definitions that

$$\begin{aligned}
\tilde{F}(q) &= \sum_{\mathbf{x}} q(x)E(x) + \sum_{\mathbf{x}} q(x)\ln q(x) && (3.7)\\
&= -\ln(Z) - \sum_{\mathbf{x}} q(x)\ln p(x) + \sum_{\mathbf{x}} q(x)\ln q(x) && (3.8)\\
&= -\ln(Z) + \sum_{\mathbf{x}} q(x)\ln\frac{q(x)}{p(x)} && (3.9)\\
&= -\ln(Z) + D(q||p), && (3.10)
\end{aligned}$$

where (3.8) follows from the definition of the Boltzmann distribution in (3.1) and (3.10) follows from the definition of relative entropy.

The relative entropy between two distributions, say q and p, is always non-negative and it is zero if and only if $q(\mathbf{x}) = p(\mathbf{x})$. We see that $\tilde{F}(q) \geq F_H$, with equality when $q(\mathbf{x}) = p(\mathbf{x})$. Therefore, one exact technique for computing the free energy and recovering $p(\mathbf{x})$ is to minimize the variational free energy with respect to trial probability distributions. The minimum that we find is the free energy and the distribution that gives this minimum is the Boltzmann distribution.

$$\begin{cases} F = \min_{q(\mathbf{x})}\tilde{F}(q) \\ p(\mathbf{x}) = \operatorname{argmin}_{q(\mathbf{x})}\tilde{F}(q) \end{cases}.$$

However this method is also intractable as N becomes large. One practical method is to minimize the variational free energy not over all the probability distributions but over a restricted subspace of trial distributions. This subspace should be simple enough so that the free energy can be computed, but should also contain distributions which are able to give a good approximation to the true behavior of the system [YFW05], [MM06].

This is the idea behind the *Bethe-Kikuchi approximation* (and similar approximations) to the variational free energy.

3.2.1 Bethe-Kikuchi Approximation to the Variational Free Energy

Usually, and in fact in many cases of interest, the energy function - that so far has been arbitrary - is determined by short-range interactions and decomposes as the sum of the energies of the subsets of the particles.

More formally, suppose $R_1, R_2, \ldots, R_M$ are subsets of $\{1, 2, \ldots, N\}$ and $\mathcal{R} = \{R_1, R_2, \ldots, R_M\}$ is a collection of subsets of the indices of the particles X_1

through X_N. Let us also suppose that $E(\mathbf{x})$ decomposes into a sum of local energies as

$$E(\mathbf{x}) = \sum_{R \in \mathcal{R}} E_R(\mathbf{x}_R), \tag{3.11}$$

where each local energy $E_R(\mathbf{x}_R)$ is a function of the particles with indices in R.

With these assumptions, the Boltzmann distribution defined in (3.1) factors into the product of local kernels as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{R \in \mathcal{R}} \alpha_R(\mathbf{x}_R), \tag{3.12}$$

where $\alpha_R(\mathbf{x}_R) = e^{E_R(\mathbf{x}_R)}$

The role of the partition function is to normalize the probability distribution. Therefore calculating the partition function is very closely related to the probabilistic inference problem stated in Section 2.1.

If the energy decomposes into sums so does the average energy. In this case the variational average energy will only depend on the marginals of the probability distribution

$$\begin{aligned} \bar{E}(q) &= \sum_{R \in \mathcal{R}} \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) E_R(\mathbf{x}_R) && (3.13) \\ &= \sum_{R \in \mathcal{R}} \bar{E}_R(q_R), && (3.14) \end{aligned}$$

where $\bar{E}_R(q_R) = \sum_{\mathbf{x}_R} q_R(\mathbf{x}_R) E_R(\mathbf{x}_R)$.

The aim is to minimize the variational free energy over all distributions $q(\mathbf{x})$, but according to (3.14) the variational average energy (3.5) depends on $q(\mathbf{x})$ only through the marginals of $q(\mathbf{x})$. The remaining term is the variational entropy.

Question 3.1. *Does the variational entropy (3.6) also depend only on the marginals of $q(\mathbf{x})$?*

The following theorem partially answers this question.

Theorem 3.2. *If $\mathcal{G} = (V, E, L)$ is a junction tree for $\mathcal{R}$, at Boltzmann equilibrium, then the entropy decomposes as sum of the entropies of the vertices minus sum of the entropies of the edges, as*

$$H(p) = \sum_{v \in V} H(p_v(\mathbf{x}_v)) - \sum_{e \in E} H(p_e(\mathbf{x}_e)). \tag{3.15}$$

Proof: Start from (2.14); take logarithms of both sides, multiply both sides by $q(\mathbf{x})$, and sum over x. ∎

One can continue to use the decomposition of Theorem 3.2, as an approximation to the variational entropy, in the case of junction graphs *with* cycles. We call this approximation the *Bethe-Kikuchi approximation* to the variational entropy with respect to $\mathcal{G}$ and denote it by $\tilde{H}^{\mathcal{G}}_{BK}(q)$.

Recall that junction graphs have local tree structures and that, in a tree, the number of vertices always exceeds the number of edges by one; this property guarantees that on both sides of (3.15) each variable is counted only once. Therefore $\tilde{H}^{\mathcal{G}}_{BK}(q)$ - which is exact on trees - also seems plausible as an approximation on junction graphs with cycles.

Example 3.3. *For the junction tree in Fig. 2.1-a, $H^{\mathcal{T}}_{BK}(q)$ is exact and can be expressed as sum of the entropies of the vertices minus sum of the entropies of the edges*

$$H^{\mathcal{T}}_{BK}(q) = H(x_1,x_2,x_3)+H(x_1,x_4)+H(x_2,x_5)+H(x_3,x_6)-H(x_1)-H(x_2)-H(x_3). \tag{3.16}$$

Example 3.4. *For both junction graphs in Fig. 3.1, $\tilde{H}^{\mathcal{G}}_{BK}(q)$ is an approximation to the exact value of the entropy and can be decomposed as*

$$\begin{aligned}\tilde{H}^{\mathcal{G}}_{BK}(q) \;=\; & H(x_1,x_2,x_5)+H(x_2,x_3,x_5)+H(x_1,x_4,x_5)+H(x_3,x_4,x_5)+H(x_1,x_4,x_5)\\ & -H(x_1,x_5)-H(x_3,x_5)-H(x_2)-H(x_4)-H(x_5),\end{aligned} \tag{3.17}$$

which sounds plausible as an approximation since each variable is counted just once. For example, on the left hand side x_5 has appeared once with positive sign, and on the right hand side has appeared four times with positive sign and three times with negative sign.

Now in (3.4), we can substitute the variational average energy from (3.14) and the variational entropy from (3.15) to get an approximation to the free energy which depends on $q(\mathbf{x})$ only through its marginals. We call this Bethe-Kikuchi approximation to the free energy with respect to $\mathcal{G}$ and denote it by $\tilde{F}^{\mathcal{G}}_{BK}(q)$

$$\tilde{F}^{\mathcal{G}}_{BK}(q) = \sum_{v\in V}\bar{E}_v(q_v(\mathbf{x}_v)) - \Big(\sum_{v\in V}H(q_v(\mathbf{x}_v)) - \sum_{e\in E}H(q_e(\mathbf{x}_v))\Big). \tag{3.18}$$

Therefore, one plausible technique to approximate the free energy and recover approximate marginals of $p(\mathbf{x})$, is to minimize the Bethe-Kikuchi free energy with respect to trial marginal probability distributions. The minimum that we find is a plausible estimate of the free energy and the marginal distributions that gives this

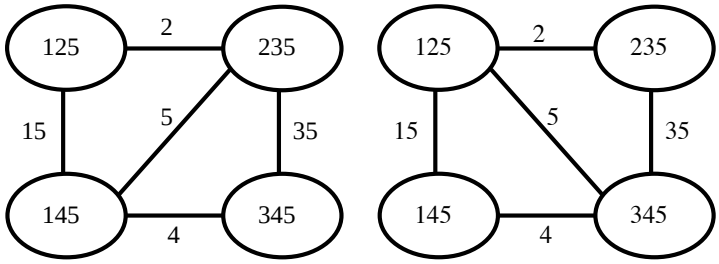

Figure 3.1: $H^{\mathcal{G}}_{BK}(q)$ on junction graphs is an approximation to H.

minimum are plausible approximates for the marginals of the Boltzmann distribution.

$$\begin{cases} \tilde{F}_{BK} = \min_{\{q_v(\mathbf{x}_v), q_e(\mathbf{x}_e)\}} \tilde{F}_{BK}(q_v(\mathbf{x}_v), q_e(\mathbf{x}_e)) \\ \{q_v(\mathbf{x}_v), q_e(\mathbf{x}_e)\} = \operatorname{argmin}_{\{q_v(\mathbf{x}_v), q_e(\mathbf{x}_e)\}} \tilde{F}_{BK}(q_v(\mathbf{x}_v), q_e(\mathbf{x}_e)) \end{cases}.$$

New theoretical results show that the stationary points of the Bethe-Kikuchi energy function (3.18) subject to some consistency constraints coincide with the fixed points of the GDL message passing algorithm with local kernels defined as $\alpha_R(\mathbf{x}_R) = e^{-E_R(\mathbf{x}_R)}$ [YFW05], [AM01], [PA05]. Therefore, establishing a link between message passing algorithms and certain approximations in statistical physics. The idea is that having plausible approximations to the energy function gives hope that the minimizing arguments are also reasonable approximations to the exact marginals.

3.3 Entropy Decomposition on Junction Trees

According to Theorem 3.2, at the Boltzmann equilibrium there is a simple decomposition for the entropy on junction trees. This decomposition is not correct in general. The following theorem shows that for a given collection of random variables the Bethe-Kikuchi approximation on *any* junction tree representation is an upper bound to the exact value of the entropy function.

Theorem 3.5. *Let $q(\mathbf{y})$ be an arbitrary probability distribution on $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_N\}$ and $\mathcal{S}$ any collection of subsets of the index set $\{1, 2, \cdots, N\}$ for which a junction tree $\mathcal{T} = (V, E, L)$ exists. Then, the following bound holds*

$$H(q) \leq H^{\mathcal{T}}_{BK}(q), \tag{3.19}$$

with equality if and only if $q(\mathbf{x})$ factors on $\mathcal{T}$ *as*

$$q(\mathbf{y}) = \frac{\prod_{v \in V} q_v(\mathbf{y}_{L(v)})}{\prod_{e \in E} q_e(\mathbf{y}_{L(e)})}. \tag{3.20}$$

Proof: The proof is by induction on M the number of vertices of the junction tree. The only inequality we will use throughout the proof is the positivity of the conditional mutual information.

For the base $M = 2$, consider $\mathcal{T}_2 = (V_2, E_2, L_2)$ a junction tree for $\mathcal{S} = \{S_1, S_2\}$. This tree consists of two vertices and one connecting edge with indices in S_1 and S_2, and $S_1 \cap S_2$ respectively. Define $S_1' = S_1 \setminus S_2$ and $S_2' = S_2 \setminus S_1$, we have

$$I(\mathbf{Y}_{S_1'}; \mathbf{Y}_{S_2'} | \mathbf{Y}_{S_1 \cap S_2}) = H(\mathbf{Y}_{S_1}) + H(\mathbf{Y}_{S_2}) - H(\mathbf{Y}_{S_1 \cap S_2}) - H(\mathbf{Y}_{S_1 \cup S_2}). \tag{3.21}$$

Hence by the positivitiy of the conditional mutual information, we have

$$H(\mathbf{Y}) \leq H(\mathbf{Y}_{S_1}) + H(\mathbf{Y}_{S_2}) - H(\mathbf{Y}_{S_1 \cap S_2}). \tag{3.22}$$

In other words, $H(q) \leq H_{BK}^{\mathcal{T}_2}(q)$ with equality if and only if $q(\mathbf{y}) = \frac{q(\mathbf{y}_{S_1}) q(\mathbf{y}_{S_2})}{q(\mathbf{y}_{S_1 \cap S_2})}$.

For the induction step, consider $\mathcal{T}_{M-1}$, the junction tree representing $\mathcal{S} = \{S_1, S_2, \ldots, S_{M-1}\}$. Induction hypothesis on $\mathcal{T}_{M-1} = (V_{M-1}, E_{M-1}, L_{M-1})$ yields

$$H(\mathbf{Y}_{\bigcup_{i=1}^{M-1} S_i}) \leq \sum_{v \in V_{M-1}} H(\mathbf{Y}_{L_{M-1}(v)}) - \sum_{e \in E_{M-1}} H(\mathbf{Y}_{L_{M-1}(e)}). \tag{3.23}$$

Build the new junction tree $\mathcal{T}_M = (V_M, E_M, L_M)$ by adding a new vertex to $\mathcal{T}_{M-1}$. Without loss of generality we assume that $S_M \cap S_1 \neq \emptyset$ and we connect the new vertex with indices in S_M to the vertex with indices in S_1. See Fig 3.2

Now if we define $S_M' = S_M \setminus \bigcup_{i=1}^{M-1} S_i$ and $S' = \bigcup_{i=1}^{M-1} S_i \setminus S_M$, positivity of $I(\mathbf{Y}_{S_M'}; \mathbf{Y}_{S'} | \mathbf{Y}_{\cap_{i=1}^{M} S_i})$ allows us to write

$$H(\mathbf{Y}_{\bigcup_{i=1}^{M} S_i}) \leq H(\mathbf{Y}_{\bigcup_{i=1}^{M-1} S_i}) + H(\mathbf{Y}_{S_M}) - H(\mathbf{Y}_{S_M \cap (\bigcup_{i=1}^{M-1} S_i)}). \tag{3.24}$$

The induction hypothesis in (3.23) together with (3.24) yield

$$H(\mathbf{Y}_{\bigcup_{i=1}^{M} S_i}) \leq \sum_{v \in V_{M-1}} H(\mathbf{Y}_{L_{M-1}(v)}) - \sum_{e \in E_{M-1}} H(\mathbf{Y}_{L_{M-1}(e)}) + H(\mathbf{Y}_{S_M}) - H(\mathbf{Y}_{S_M \cap (\bigcup_{i=1}^{M-1} S_i)}). \tag{3.25}$$

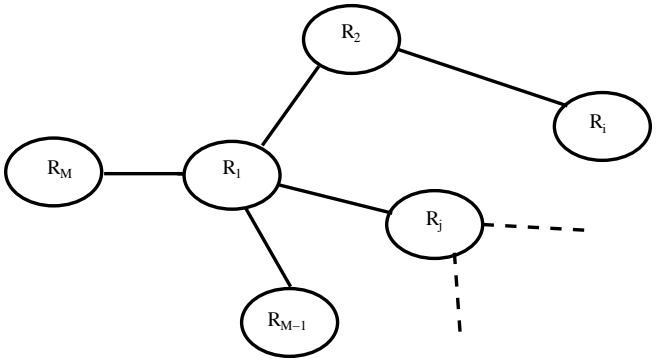

Figure 3.2: Junction Graph in Theorem 3.5

The junction tree property, stated in Section 2.2.1, guarantees that $S'_M \cap (\bigcup_{i=1}^{M-1} S_i) = \emptyset$ and the label on the edge between S_1 and S_M in $\mathcal{T}_M$ is precisely equal to $S_M \cap (\bigcup_{i=1}^{M-1} S_i)$. Thus, we can rewrite (3.25) as

$$H(\mathbf{Y}_{\bigcup_{i=1}^{M} S_i}) \leq \sum_{v \in V_M} H(\mathbf{Y}_{L_M(v)}) - \sum_{e \in E_M} H(\mathbf{Y}_{L_M(e)}). \tag{3.26}$$

Hence we have shown that $H(q) \leq H_{BK}^{\mathcal{T}_M}(q)$ with equality if and only if

$$q(\mathbf{y}) = \frac{\prod_{v \in V_M} q_v\left(\mathbf{y}_{L_M(v)}\right)}{\prod_{e \in E_M} q_e\left(\mathbf{y}_{L_M(e)}\right)}$$

. ∎

Remark 3.6. Alternative proofs for Theorem 3.5 can be found in [DLMO04], [WJW05, Appendix A], and [WJ93, Section 8.4] for its extension to hypertrees. However, our proof that first appeared in [MP05] is different.

Remark 3.7. We finish this section by stating one direct implication of Theorem 3.5. If an *arbitrary* probability distribution $q(\mathbf{x})$ has the same marginals as the Boltzmann distribution on a junction tree, its entropy is upper bounded by the entropy of the Boltzmann distribution. For the proof just consider the fact that the Bethe-Kikuchi approximation takes the same value for both distributions while it is an upper bound for $H(q)$ and is tight for the Boltzmann distribution.

Chapter 4

Subtree-Based Upper and Lower Bound on the Partition Function

For a general junction graph, i.e., with cycles, calculating the partition function in a straightforward manner, as expressed in (2.4), involves a sum with an exponential number of terms. Therefore it is desirable to have upper and lower bounds on partition function which can be obtained with low complexity.

In the context of deterministic bounds on the partition function, mean field theory provides a popular lower bound, see [YFW05], [JS93]. A general optimization framework for deriving and examining various mean field approximations is proposed in [Zha96]. In [LK01], the authors derive lower bounds that are tighter than the mean field bounds. For the special case of Ising models, [JJ96] proposes a recursive procedure to derive upper bounds on the partition function. Based on concepts from convex duality and information geometry, in [WJW05] a new class of upper bounds are derived. In this chapter, considering the sub-junction trees of a junction graph, we derive upper and lower bounds on the partition function.

The outline of this chapter is as follows. In Section 4.1, we derive subtree-based upper and lower bounds on the partition function. In Section 4.2, using the entropies of the sub-junction trees, we prove an inequality that compares the lower bounds obtained from different sub-trees. In Section 4.3, we show that among all sub-trees, the minimum entropy sub-tree has some optimality properties.

4.1 Subtree-Based Upper and Lower Bounds

In this section, for a general junction graph $\mathcal{G}$ we derive upper and lower bounds on the partition function. In this set-up, the key observation is that any junction graph has a large number of sub-junction trees on which the partition function can be computed efficiently, using the GDL or any other message passing algorithm,

see (2.16). The bounds depend on the partition function of $\mathcal{G}_T$ a sub-junction tree of $\mathcal{G}$.

Consider a probabilistic inference problem defined by $\mathcal{R} = \{R_1, R_2, \ldots, R_M\}$. Suppose $\mathcal{R}_T$ is a subset of $\mathcal{R}$ that has a junction tree representation. Also suppose that $q_T(\mathbf{x})$ denotes the global probability distribution and Z_T the partition function on $\mathcal{G}_T$ respectively.

With these assumptions, we can rewrite $p(\mathbf{x})$, the global probability distribution on $\mathcal{G}$ defined in (2.1) as

$$\begin{aligned} p(\mathbf{x}) &= \frac{1}{Z} \prod_{R \in \mathcal{R}_T} \alpha_R(\mathbf{x}_R) \prod_{R \in \mathcal{R} \setminus \mathcal{R}_T} \alpha_R(\mathbf{x}_R) \\ &= \frac{Z_T}{Z} q_T(\mathbf{x}) \prod_{R \in \mathcal{R} \setminus \mathcal{R}_T} \alpha_R(\mathbf{x}_R), \end{aligned} \tag{4.1}$$

where

$$Z_T = \sum_{\mathbf{x}} \prod_{R \in \mathcal{R}_T} \alpha_R(\mathbf{x}_R).$$

In the following, using the positivity of $D(p(\mathbf{x})||q_T(\mathbf{x}))$ and $D(q_T(\mathbf{x})||p(\mathbf{x}))$ and our reformulation in (4.1), we derive upper and lower bounds on Z.

4.1.1 Upper Bounds

To derive the upper bound, (4.1) allows us to write $D\big(p(\mathbf{x})||q_T(\mathbf{x})\big)$ as

$$D\big(p(\mathbf{x})||q_T(\mathbf{x})\big) = D\big(p(\mathbf{x})||\frac{Z}{Z_T} \cdot \frac{p(\mathbf{x})}{\prod_{R \in \mathcal{R} \setminus \mathcal{R}_T} \alpha_R(\mathbf{x}_R)}\big) \tag{4.2}$$

$$= \sum_{\mathbf{x}} p(\mathbf{x}) \ln(p(\mathbf{x})) - \ln(\frac{Z}{Z_T}) - \sum_{\mathbf{x}} p(\mathbf{x}) \ln(p(\mathbf{x})) \tag{4.3}$$

$$+ \sum_{\mathbf{x}} p(\mathbf{x}) \ln \prod_{R \in \mathcal{R} \setminus \mathcal{R}_T} \alpha_R(\mathbf{x}_R)$$

$$= \ln(\frac{Z_T}{Z}) + \sum_{\mathbf{x}_R} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_T} p(\mathbf{x}_R) \ln \alpha_R(\mathbf{x}_R). \tag{4.4}$$

Hence by the non-negativity of relative entropy we obtain

$$\ln(Z) \leq \ln(Z_T) + \sum_{\mathbf{x}_R} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_T} p(\mathbf{x}_R) \ln \alpha_R(\mathbf{x}_R). \tag{4.5}$$

4.1.2 Lower Bounds

To derive the lower bound, we take logarithms of both sides of (4.1), multiply both sides by $q_T(\mathbf{x})$, and sum over $\mathbf{x}$ to obtain.

$$\sum_{\mathbf{x}} q_T(\mathbf{x}) \ln p(\mathbf{x}) \;=\; \ln(\frac{Z_T}{Z}) + \sum_{\mathbf{x}} q_T(\mathbf{x}) \ln q_T(\mathbf{x}) + \sum_{\mathbf{x}} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_T} q_T(\mathbf{x}) \ln \alpha_R(\mathbf{x}_R). \quad (4.6)$$

Rearranging (4.6) gives

$$-\, D\big(q_T(\mathbf{x}) || p(\mathbf{x})\big) \;=\; \ln(\frac{Z_T}{Z}) + \sum_{\mathbf{x}} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_T} q(\mathbf{x}) \ln \alpha_R(\mathbf{x}_R). \quad (4.7)$$

Again, by non-negativity of relative entropy, we obtain

$$\sum_{\mathbf{x}} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_T} q_T(\mathbf{x}) \ln \alpha_R(\mathbf{x}_R) + \ln(Z_T) \leq \ln(Z). \quad (4.8)$$

4.1.3 Upper and Lower Bounds

From (4.5) and (4.8), we have the following upper and lower bounds on the partition function, see [MP05], [MP06]

$$\sum_{\mathbf{x}} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_T} q_T(\mathbf{x}) \ln \alpha_R(\mathbf{x}_R) \leq \ln(\frac{Z}{Z_T}) \leq \sum_{\mathbf{x}_R} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_T} p(\mathbf{x}_R) \ln \alpha_R(\mathbf{x}_R). \quad (4.9)$$

Let us denote the lower bound computed using q_T, i.e., the left hand side of (4.8) by $\mathcal{L}_{q_T}$. It follows immediately that

$$\mathcal{L}_{q_T} = \ln(Z) - D\big(q_T(\mathbf{x}) || p(\mathbf{x})\big) \quad (4.10)$$

4.2 Comparing Lower Bounds

According to (4.9) every sub-junction tree gives upper and lower bounds on the partition function. Let us recall that according to Theorem 3.2, on junction trees the entropy can be efficiently computed as the sum of the entropies of the vertices minus sum of the entropies of the edges.

In the following we prove a Theorem that suggests a method to compare the lower bounds derived in (4.8) according to the entropies of the sub-trees.

Theorem 4.1. *Consider $\mathcal{R}_1$ and $\mathcal{R}_2$, subsets of $\mathcal{R}$ with junction tree representations. Suppose that $q_1(\mathbf{x})$, Z_1, $q_2(\mathbf{x})$, and Z_2 denote the global probability distributions and the partition functions over $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively. Without loss of generality suppose $H(q_1) \leq H(q_2)$. Then the following inequality holds*

$$\mathcal{L}_{q_2} \leq \mathcal{L}_{q_1} + \min\big(D(q_1||\overline{q}_1) - D(q_2||q_1), D(q_1||q_2) + D(q_1||\overline{q}_2)\big), \tag{4.11}$$

where $\overline{q}_1$ and $\overline{q}_2$ denote the global probability distributions defined on $\mathcal{R} \setminus \mathcal{R}_1$ and $\mathcal{R} \setminus \mathcal{R}_2$ respectively, [MP05, Theorem 2] .

Proof: As in (4.1), we can rewrite $p(\mathbf{x})$ in the following forms

$$p(\mathbf{x}) = \frac{Z_1}{Z} q_1(\mathbf{x}) \prod_{R \in \mathcal{R} \setminus \mathcal{R}_1} \alpha_R(\mathbf{x}_R) \tag{4.12}$$

$$= \frac{Z_2}{Z} q_2(\mathbf{x}) \prod_{R \in \mathcal{R} \setminus \mathcal{R}_2} \alpha_R(\mathbf{x}_R). \tag{4.13}$$

Plugging (4.12) into $D(q_1(\mathbf{x})||p(\mathbf{x}))$, we have

$$D(q_1||p) = -\ln(\frac{Z_1}{Z}) - \sum_{\mathbf{x}} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_1} q_1(\mathbf{x}) \ln \alpha_R(\mathbf{x}_R). \tag{4.14}$$

And by plugging (4.12) into $D(q_2(\mathbf{x})||p(\mathbf{x}))$, we have

$$D(q_2||p) = D(q_2||q_1) - \ln(\frac{Z_1}{Z}) - \sum_{\mathbf{x}} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_1} q_2(\mathbf{x}) \ln \alpha_R(\mathbf{x}_R). \tag{4.15}$$

We now substitute $\ln(\frac{Z_1}{Z})$ from (4.15) into (4.14) to obtain

$$D(q_1||p) = D(q_2||p) - D(q_2||q_1) + \sum_{\mathbf{x}} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_{T_1}} (q_2(\mathbf{x}) - q_1(\mathbf{x})) \ln \alpha_R(\mathbf{x}_R). \tag{4.16}$$

After a little calculation

$$D(q_1||p) = D(q_2||p) - D(q_2||q_1) + \sum_{\mathbf{x}} (q_2(\mathbf{x}) - q_1(\mathbf{x})) \ln \overline{q}_1(\mathbf{x}). \tag{4.17}$$

Jensen's inequality allows us to write

$$\sum_{\mathbf{x}} q_2(\mathbf{x}) \ln \overline{q}_1(\mathbf{x}) \leq \sum_{\mathbf{x}} q_2(\mathbf{x}) \ln q_2(\mathbf{x}). \tag{4.18}$$

We can now add (4.17) and (4.18) to obtain

$$D(q_1||p) \leq D(q_2||p) - D(q_2||q_1) + \sum_{\mathbf{x}} q_2(\mathbf{x}) \ln q_2(\mathbf{x}) - \sum_{\mathbf{x}} q_1(\mathbf{x}) \ln \overline{q}_1(\mathbf{x}). \tag{4.19}$$

Now since $H(q_1) \leq H(q_2)$ we can write

$$\sum_{\mathbf{x}} q_2(\mathbf{x}) \ln q_2(\mathbf{x}) \leq \sum_{\mathbf{x}} q_1(\mathbf{x}) \ln q_1(\mathbf{x}). \tag{4.20}$$

We can now add (4.19) and (4.20) to obtain

$$D(q_1||p) \leq D(q_2||p) - D(q_2||q_1) + D(q_1||\overline{q}_1). \tag{4.21}$$

Using the definition of $\mathcal{L}_{q_T}$ in (4.7), we have

$$\mathcal{L}_{q_1} = \ln(Z) - D(q_1||p) \tag{4.22}$$
$$\mathcal{L}_{q_2} = \ln(Z) - D(q_2||p) \tag{4.23}$$

We can substitute $D(q_1||p)$ and $D(q_2||p)$ from (4.22) and (4.23) into (4.21) to derive

$$\mathcal{L}_{q_2} \leq \mathcal{L}_{q_1} + D(q_1||\overline{q}_1) - D(q_2||q_1). \tag{4.24}$$

Similarly, by plugging (4.13) into $D(q_2(\mathbf{x})||p(\mathbf{x}))$, we have

$$D(q_2||p) = -\ln(\frac{Z_2}{Z}) - \sum_{\mathbf{x}} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_2} q_2(\mathbf{x}) \ln \alpha_R(\mathbf{x}_R). \tag{4.25}$$

And by plugging (4.13) into $D(q_1(\mathbf{x})||p(\mathbf{x}))$, we have

$$D(q_1||p) = D(q_1||q_2) - \ln(\frac{Z_2}{Z}) - \sum_{\mathbf{x}} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_2} q_1(\mathbf{x}) \ln \alpha_R(\mathbf{x}_R). \tag{4.26}$$

We now substitute $\ln(\frac{Z_2}{Z})$ from (4.26) into (4.25) to obtain

$$D(q_2||p) = D(q_1||p) - D(q_1||q_2) + \sum_{\mathbf{x}} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_2} (q_1(\mathbf{x}) - q_2(\mathbf{x})) \ln \alpha_R(\mathbf{x}_R). \tag{4.27}$$

After a little calculation

$$D(q_2||p) = D(q_1||p) - D(q_1||q_2) + \sum_{\mathbf{x}} (q_1(\mathbf{x}) - q_2(\mathbf{x})) \ln \overline{q}_2(\mathbf{x}) \tag{4.28}$$

Jensen's inequality allows us to write

$$\sum_{\mathbf{x}} q_2(\mathbf{x}) \ln \overline{q}_2(\mathbf{x}) \;\leq\; \sum_{\mathbf{x}} q_2(\mathbf{x}) \ln q_2(\mathbf{x}). \tag{4.29}$$

We can now add (4.28) and (4.29) to obtain

$$D(q_1||p) - D(q_1||q_2) + \sum_{\mathbf{x}} q_1(\mathbf{x}) \ln \overline{q}_2(\mathbf{x}) - \sum_{\mathbf{x}} q_2(\mathbf{x}) \ln q_2(\mathbf{x}) \leq D(q_2||p). \tag{4.30}$$

Now since $H(q_1) \leq H(q_2)$ we can write

$$\sum_{\mathbf{x}} q_2(\mathbf{x}) \ln q_2(\mathbf{x}) \;\leq\; \sum_{\mathbf{x}} q_1(\mathbf{x}) \ln q_1(\mathbf{x}). \tag{4.31}$$

We can now add (4.30) and (4.31) to obtain

$$D(q_1||p) - D(q_1||q_2) - D(q_1||\overline{q}_2) \leq D(q_2||p). \tag{4.32}$$

Using the definition of $\mathcal{L}_{q_T}$ in (4.7) and (4.22) and (4.23), we derive the following inequality

$$\mathcal{L}_{q_2} \leq \mathcal{L}_{q_1} + D(q_1||q_2) + D(q_1||\overline{q}_2). \tag{4.33}$$

From (4.24) and (4.33), we conclude that

$$\mathcal{L}_{q_2} \leq \mathcal{L}_{q_1} + \min\big(D(q_1||\overline{q}_1) - D(q_2||q_1), D(q_1||q_2) + D(q_1||\overline{q}_2)\big). \tag{4.34}$$

∎

Corollary 4.2. *In the case that $\mathcal{R}_1 = \mathcal{R} \setminus \mathcal{R}_2$, namely when the junction graph decomposes into two junction trees (for example this can be the case when we choose a sub-tree in a graph with only one cycle); if $H(q_1) \leq H(q_2)$, the inequality in (4.11) simplifies to*

$$\mathcal{L}_{q_2} + D(q_2||q_1) \;\leq\; \mathcal{L}_{q_1} + D(q_1||q_2). \tag{4.35}$$

Note that, in general, the relative entropy is not symmetric therefore equation (4.35) does not tell if one bound is better than the other.

Substituting $\mathcal{L}_{q_1}$ and $\mathcal{L}_{q_2}$ from (4.22) and (4.23) we can rewrite (4.35) in the following form

$$D(q_2||q_1) + D(q_1||p) \;\leq\; D(q_1||q_2) + D(q_2||p). \tag{4.36}$$

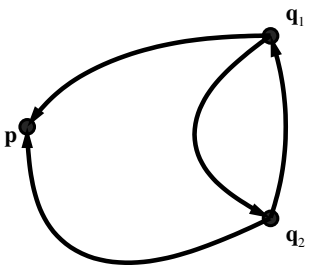

Figure 4.1: To p via q_1 or via q_2

In other words the distance from q_2 to p via q_1 is shorter than the distance from q_1 to p via q_2. See Fig. 4.1.

Note that, in general, the relative entropy does not satisfy the triangular inequality therefore equation (4.36) does not tell if one bound is better than the other either.

From (4.36) and the positivity of $D(q_2||q_1)$ one concludes that the distance from q_1 to p is shorter than the distance from q_1 to p via q_2 since

$$D(q_1||p) \;\leq\; D(q_1||q_2) + D(q_2||p). \tag{4.37}$$

4.3 On Properties of the Minimum Entropy Sub-tree to Compute Lower Bounds on Partition Function

In this section we investigate the properties of one specific lower bound on the partition function, namely the lower bound computed by the minimum entropy sub-tree. By extending the results of Theorem 4.1 and stating new theorems and corollaries we prove that there is an upper bound on how much any other lower bound can be better than the one obtained from that the minimum entropy sub-tree. We also show that the probability distribution over the sub-tree that gives the best lower bound and the one with minimum entropy are close in divergence.

Corollary 4.3. *Consider a subset $\mathcal{R}_S$ of $\mathcal{R}$ with junction tree representation. Also suppose that the probability distribution q_S over $\mathcal{R}_S$, has the smallest entropy among all the probability distributions on sub-trees. Then for any subset $\mathcal{R}_T$ of $\mathcal{R}$ with junction tree representation the following inequality holds*

$$\mathcal{L}_{q_T} \;\leq\; \mathcal{L}_{q_S} \;+\; D(q_S||\overline{q}_S). \tag{4.38}$$

Proof: According to Theorem 4.1

$$\mathcal{L}_{q_T} \leq \mathcal{L}_{q_S} + \quad \min\big(D(q_S||\overline{q}_S) - D(q_T||q_S), D(q_S||q_T) + D(q_S||\overline{q}_T)\big). \tag{4.39}$$

Hence

$$\begin{aligned} \mathcal{L}_{q_T} &\leq \mathcal{L}_{q_S} + D(q_S||\overline{q}_S) - D(q_T||q_S) \\ &\leq \mathcal{L}_{q_S} + D(q_S||\overline{q}_S). \end{aligned} \tag{4.40}$$

∎

In other words the lower bound obtained from any sub-tree can not be better than the lower bound obtained from the minimum entropy sub-tree by more than $D(q_S||\overline{q}_S)$, a value that does not depend on q_T. This gives us a quality guarantee for the lower bound obtained from the minimum entropy sub-tree.

Theorem 4.4. *Consider subsets $\mathcal{R}_S$ and $\mathcal{R}_B$ of $\mathcal{R}$ with junction tree representations. Also suppose the probability distribution q_S over $\mathcal{R}_S$, has the smallest entropy and the probability distribution q_B over $\mathcal{R}_B$, gives the best lower bound, then the following inequality holds*

$$D(q_B||q_S) \leq D(q_S||\overline{q}_S). \tag{4.41}$$

Proof: Since $H(q_S) \leq H(q_B)$ according to Theorem 4.1 we have

$$\mathcal{L}_{q_B} \leq \mathcal{L}_{q_S} + D(q_S||\overline{q}_S) - D(q_B||q_S). \tag{4.42}$$

Since q_B gives the best lower bound

$$\mathcal{L}_{q_S} \leq \mathcal{L}_{q_B}. \tag{4.43}$$

Adding (4.42) and (4.43) completes the proof. ∎

Theorem 4.4 gives us another quality guarantee regarding the minimum entropy sub-tree (which is the least random, least uncertain, and most biased sub-tree). This theorem shows that the probability distribution on the tree that gives the best lower bound and the minimum entropy distribution are close, where closeness is measured by relative entropy. The upper bound is $D(q_S||\overline{q}_S)$ which does not depend on q_B. See Fig. 4.2.

Remark 4.5. The inequality in (4.41) holds for any probability distribution that gives a better lower bound compared to the lower bound derived from q_S, i.e. $\mathcal{L}_{q_S}$.

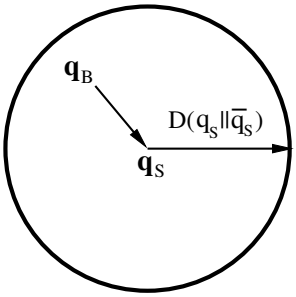

Figure 4.2: Upper bound for divergence between q_B and q_S

Corollary 4.6. *In the case that $\mathcal{R}_S = \mathcal{R} \setminus \mathcal{R}_B$ we have the following inequality.*

$$D(q_B||q_S) \leq D(q_S||q_B). \tag{4.44}$$

Remark 4.7. We showed that the minimum entropy sub-tree has some interesting optimality properties, namely the lower bound obtained from this tree can not be far from the lower bound obtained from any other sub-tree, and the probability distribution on this tree and the probability distribution on the tree that gives the best lower bound are close in divergence. However, finding the minimum entropy sub-tree does not seem like an easy task. One might still hope for approximate algorithms to find this sub-tree.

Remark 4.8. In almost all Theorems and Corollaries we insisted that the subsets of $\mathcal{R}$ have junction-tree representations. This assumption can be relaxed and the Theorems and Corollaries are still valid for the sub-graphs. However, having a junction tree representation makes the computation of the entropy and the partition function easier (using GDL or any other iterative message passing algorithm).

Remark 4.9. In the derivation of the lower bound in Section 4.1.2 we assumed that $q_T(\mathbf{x})$, like $p(\mathbf{x})$, depends on all the random variables and not only on the random variables on the sub-tree. Similar results can be obtained by writing the distribution on subtrees as a function of the random variables on the sub-tree.

Let us assume the variables on the sub-tree by $\mathbf{x}_T$ and the variables outside the sub-tree by $\mathbf{x}_{\setminus T}$.

Define

$$Z_T = \sum_{\mathbf{x}_T} \prod_{R \in \mathcal{R}_T} \alpha_R(\mathbf{x}_R). \tag{4.45}$$

Now it is easy to see that

$$- D\big(q_T(\mathbf{x}_T)||p(\mathbf{x}_T)\big) \;\geq\; \ln(\frac{Z_T}{Z}) + \sum_{\mathbf{x}_T} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_T} q(\mathbf{x}_T) \ln \alpha_R(\mathbf{x}_R). \tag{4.46}$$

Hence by the non-negativity of $D\big(q_T(\mathbf{x}_T)||p(\mathbf{x}_T)\big)$ we will obtain

$$\sum_{\mathbf{x}_T} \sum_{R \in \mathcal{R} \setminus \mathcal{R}_T} q_T(\mathbf{x}_T) \ln \alpha_R(\mathbf{x}_R) + \ln(Z_T) \leq \ln(Z). \tag{4.47}$$

The result in (4.47) should be compared to (4.8).

Chapter 5

A Greedy Algorithm to Compute Upper and Lower Bounds on the Partition Function

In Chapter 4 we derived upper and lower bounds on the partition function that depend on the partition function of any sub-junction tree of a given junction graph. We proved Theroem 4.1 that compares such lower bounds using the entropies of the sub-trees. We also showed that the minimum entropy sub-tree has some interesting optimality properties.

The number of all the sub-junction trees of a junction graph with moderate size is simply too big to consider. For example, a complete graph with N vertices has exactly N^{N-2} spanning trees. In this chapter by simplifying our previous bounds on the partition function, we derive new and simpler upper and lower bounds with a nice property that allows us to use a greedy scheme.

The outline of this chapter is as follows. In Section 5.1, we derive new and simple bounds on the partition function. We prove that the optimization for the new bounds can be done over the maximal sub-trees and not on all sub-trees. Based on new bounds, in Section 5.2 we propose a greedy algorithm that computes upper and lower bound on the partition function. In Section 5.3 we report simulation results for the greedy algorithm on two-dimensional square grids.

5.1 New Upper and Lower Bounds

Let us define

$$\alpha_R^{\max} \triangleq \max_{\mathbf{x}_R}\{\alpha_R(\mathbf{x}_R)\}, \tag{5.1}$$

$$\alpha_R^{\min} \triangleq \min_{\mathbf{x}_R}\{\alpha_R(\mathbf{x}_R)\}. \tag{5.2}$$

With the above definitions, we may still be able to simplify the bounds in (4.9) and obtain the following simple and low-complexity upper and lower bounds on the partition function.

$$\sum_{R\in\mathcal{R}\setminus\mathcal{R}_T} \ln \alpha_R^{\min} \leq \ln(\frac{Z}{Z_T}) \leq \sum_{R\in\mathcal{R}\setminus\mathcal{R}_T} \ln \alpha_R^{\max}. \tag{5.3}$$

For a given junction graph, these bounds can be computed over its sub-junction trees. However the following Theorem shows that we only need to consider *maximal* sub-trees and *not all* the sub-junction trees.

Theorem 5.1. *If $\mathcal{R}_1$ and $\mathcal{R}_2$ are subsets of $\mathcal{R}$ with junction tree representations and $\mathcal{R}_2 \subseteq \mathcal{R}_1$, then the bounds of (5.3) corresponding to $\mathcal{R}_1$ are at least as good as those corresponding to $\mathcal{R}_2$.*

Proof: Consider $\mathcal{T}_1 = (V_1, E_1, L_1)$ the junction tree representing $\mathcal{R}_1$, remove vertices with indices in $\mathcal{R}_1 \setminus \mathcal{R}_2$ from V_1 and build a new junction tree $\mathcal{T}_2 = (V_2, E_2, L_2)$ representing $\mathcal{R}_2$.

Now starting from the lower bound obtained from $\mathcal{T}_1$, we can write

$$\ln(Z_1) + \sum_{R\in\mathcal{R}\setminus\mathcal{R}_1} \ln \alpha_R^{\min} \geq \ln(Z_2) + \sum_{R\in\mathcal{R}_1\setminus\mathcal{R}_2} \ln \alpha_R^{\min} + \sum_{R\in\mathcal{R}\setminus\mathcal{R}_1} \ln \alpha_R^{\min} \tag{5.4}$$

$$= \ln(Z_2) + \sum_{R\in\mathcal{R}\setminus\mathcal{R}_2} \ln \alpha_R^{\min}. \tag{5.5}$$

The proof for the upper bound follows from the same lines.

∎

Definition 5.2. A sub-junction tree is maximal if it is not contained within any other sub-junction tree.

According to Theorem 5.1 if $\mathcal{R}_1$ and $\mathcal{R}_2$ are subsets of $\mathcal{R}$ with junction tree representations and $\mathcal{R}_2 \subseteq \mathcal{R}_1$, the bounds of (5.3) corresponding to $\mathcal{R}_1$ are at least as good as those corresponding to $\mathcal{R}_2$. We can then propose the following greedy algorithm.

5.2 A Greedy Algorithm

Using the results of Theorem 5.1, we propose the following greedy algorithm to find a *maximal* sub-junction tree that gives lower and upper bounds on the partition function. According to Theorem 5.1, the bounds are at least as good as, if not better, the lower and upper bounds obtained from all the sub-junctiontrees of this maximal sub-tree.

Algorithm 5.3	GREEDY-UPPER-LOWER-BOUNDS($\mathcal{G}$)
Input	Junction graph $\mathcal{G}$ representing an inference problem
Output	Upper and lower bounds on partition function

1. Set $\mathcal{T}$ a sub-junction tree, U the upper bound, and L the lower bound.
2. **while** junction tree condition not violated **do**
3. Recursively add nodes to $\mathcal{T}$
4. Compute bounds in (5.3) for the maximal tree
5. **if** new bound(s) tighter than old bound(s)
6. Replace old bound(s) with new bound(s): update U and L
7. Repeat for another maximal sub-junction tree ∎

5.2.1 Properties of the Greedy Algorithm

Here we state a few remarks and methods to improve over the greedy algorithm and existing bounds in (5.3).

1. The minimization and maximization as stated in (5.3) are done over single local kernels (nodes) separately; better results can be obtained if we do this over a group of neighboring nodes. In this case using max(min)-product algorithms can be an option if after removing the maximal tree, the remaining graph is cycle-free or has at most one cycle.
2. A kernel for which the values of $\alpha_R^{\max}$ and $\alpha_R^{\min}$ are close to each other, once not included in the maximal tree, tries to make the bounds tight. Therefore it seems reasonable to try to leave such weak nodes out of the sub-junction tree to obtain better bounds.

3. Consider a junction graph with only one cycle. In this case, only one kernel (node) remains out of the maximal sun-junction tree. If on this kernel the values of $\alpha_R^{\max}$ and $\alpha_R^{\min}$ are close, so are the upper and lower bounds. We call this property of our bounds **linearization**.

5.3 Simulation Results

Here we apply our greedy algorithm on two-dimensional square grids to compute upper and lower bounds on the log partition function. The connectivity of a two-dimensional grid is sparse enough so that each maximal tree (in this case being a spanning tree) represents well enough the whole graph.

Our simulations are limited to 3×3 square grids with $N = 9$ variables, and 5×5 square grids with $N = 25$ variables. We suppose variables are binary and consider the *spin* representation where variables take their values in $\{-1, +1\}$.

We use the standard *Ising model* of statistical physics with the global probability distribution expressed as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{s \in V} e^{\beta_s x_s} \prod_{(s,t) \in E} e^{\beta_{st} x_s x_t}, \tag{5.6}$$

where β_s is the node parameter for node s and β_{st} is the strength of edge (s, t).

For each node, β_s is uniformly and independently distributed in $[-0.025, 0.025]$. And for each edge, $\beta_{i,j}$ is uniformly and independently distributed in $[0, t]$ where t represents the edge strength and ranges from 0 to 2. We performed 10 trials to compute each value. For the edge strength t, we considered increments of 0.5 in the interval $[0, 2]$. Our set-up is very similar to the one used in [WJW05, Section V] to compute upper bounds on the partition function.

In Fig. 5.1, for a 3×3 grid we plot the average upper bound, GDL approximation, and the average lower bound on the **log partition function** versus the edge strength t where all values are normalized by $N = 9$. Shown in 5.2, are plots for a 5×5 grid where all values are normalized by $N = 25$.

For the two sizes of grids Fig. 5.3 shows $\frac{U-L}{U}$, i.e. the gap between the bounds when normalized by the upper bound, versus the edge strength.

Remark 5.4. Note that, in general, GDL approximation does not provide a bound on the log partition function; but in our experiments it seems to always lie between the upper and lower bounds obtained from the greedy algorithm.

Remark 5.5. As the size of the grid becomes larger, the number of maximal trees also becomes very big, however, for each maximal tree our greedy algorithm guarantees bounds that are at least as good as all the bounds obtained from all the sub-trees of this maximal tree. And a bound is a bound!

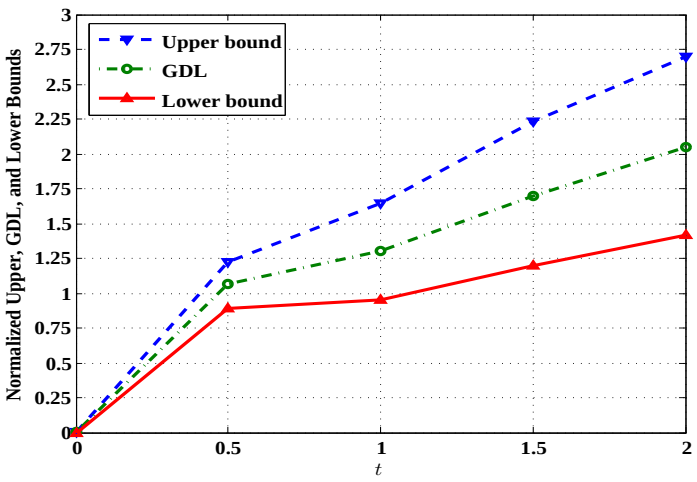

Figure 5.1: Upper and lower bounds and GDL approximation on the log partition function for grid with $N = 9$ for different edge strengths

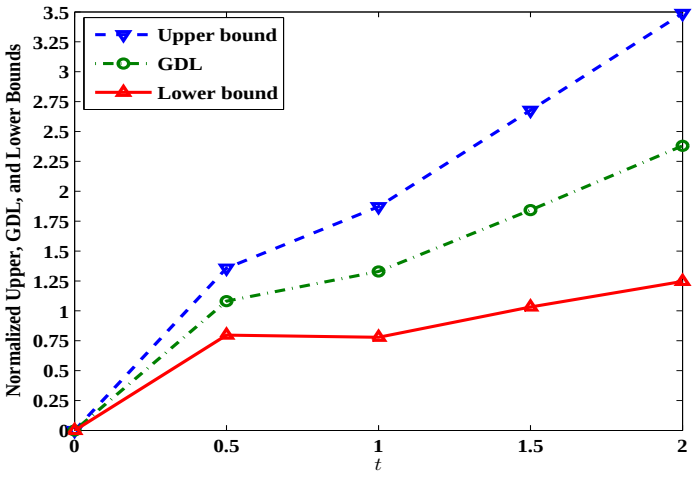

Figure 5.2: Upper and lower bounds and GDL approximation on the log partition function for grid with $N = 25$ for different edge strengths

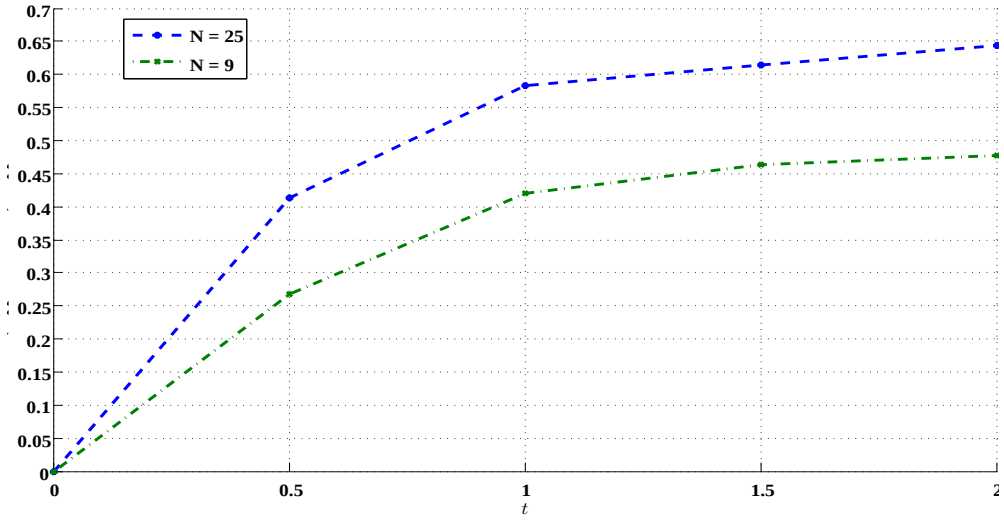

Figure 5.3: Relative gap for grids with sizes 9 and 25 for different edge strengths

Chapter 6

Simulation-Based Estimation of the Partition Function

In Chapter 4 we discussed deterministic upper and lower bounds on the partition function. There is also substantial literature on the use of Monte Carlo simulation techniques for approximating the partition function. For example, in [PG93] by expressing the partition function as an expectation, the authors introduce an importance sampling approach to estimate the partition function. A randomised algorithm to evaluate the partition function is proposed in [JS93]. Monte Carlo methods are also proposed to estimate the ratio of partition functions of probabilistic models [CS97].

In this chapter we will also use simulation techniques to approximate the partition function. We present estimators that use samples drawn from the state space according to different distributions. In one of the estimators, we demonstrate how to efficiently combine Monte Carlo methods with message passing algorithms, like GDL from Section 2.3.

The outline of this chapter is as follows. In Section 6.1, we describe Monte Carlo methods and particularly focus on Gibbs sampling. In Section 6.2, we introduce various estimators for the partition function and describe how they use Gibbs sampling to estimate the partition function. Simulation results for our estimators on two-dimensional square grids are reported in Section 6.3.

6.1 Monte Carlo Methods

Historically, Monte Carlo methods, first introduced in [MRR+53], were developed for performing calculations in statistical physics (like estimating the partition function which is also the goal of this chapter). However, today the range of applications of Monte Carlo methods is enormous, such as combinatorics (like approximate counting), Integration, combinatorial optimization, and economics. A good survey on how Monte Carlo Methods has been used in practice is [RS95].

The aims of Monte Carlo methods are

1. to generate samples $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(K)}$ from a probability distribution $p(\mathbf{x})$ - which we call *target distribution*.

2. to estimate expectations of functions under this distribution.

Solving the first problem allows us to solve the second one by using samples $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(K)}$ to give the estimator

$$\hat{f} = \frac{1}{K} \sum_{k=1}^{K} f(\mathbf{x}^{(k)}). \tag{6.1}$$

If the samples are generated from the target distribution $p(\mathbf{x})$ then the expectation of $\hat{f}$ is equal to the expectation of f under $p(\mathbf{x})$, [Mac03], [Nea93].

In other words, Monte Carlo methods give us the possibility to compute different quantities by expressing them as expectations for some distribution and then estimate this expectation by drawing samples from that distribution.

Except for a few densities, generating samples from a distribution (usually high-dimensional) is a difficult task. In general there are two difficulties [Mac03].

1. The major difficulty is that we usually do not know the value of the partition function (normalization constant).

2. Another difficulty is that typically the probability distributions are concentrated in regions of the state space that occupy a tiny fraction of the whole. To generate samples from such distributions, the sampling procedure must search for these relevant regions.

Sampling methods based on Monte Carlo methods incorporate the search aspect for large regions of high probability in a framework where it can be proved that, as the number of samples grow, the correct distribution is generated.

There are many sampling algorithms based on Monte Carlo methods such as *importance sampling*, *rejection sampling*, *Metropolis methods*, and *Gibbs sampling*. In the following we introduce one particular Monte Carlo method, namely Gibbs sampling, and will propose estimators that use this method to estimate the value of the partition function for a given probabilistic inference problem.

6.1.1 Gibbs Sampling

Gibbs sampling, also known as *heat bath method*, is one of the simplest Monte Carlo methods. It was first introduced in the context of image processing in [GG84], where authors presented a method to iteratively sample the value for each pixel conditioned on the value of the neighboring pixels. However, Gibbs sampling is widely applicable to a broad class of Bayesian problems, statistical physics, and combinatorics.

As stated above, it is usually very difficult to draw samples directly from the target distribution. The key to the Gibbs sampler is that it considers only conditional distributions - the distributions where all random variables except for one are assigned fixed values.

In our setup, such conditional distributions are easy to sample from. In fact this is commonly the case in many Bayesian and likelihood computations, see [GS90]. As we will see in Section 6.2, for the target distribution defined in (6.5) it is enough to be able to compute $p(\mathbf{x})$ up to a (normalizing) constant and the knowledge of the partition function is not necessary.

Let $p(\mathbf{x})$ be a probability function over $x_1, x_2, \ldots, x_n$. Suppose the conditional distributions $p(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ are given. The following iterative algorithm, known as Gibbs sampling, generates samples with distribution $p(\mathbf{x})$.

Algorithm 6.1	GIBBS SAMPLING 1
Input	Conditional distributions $p(x_i\|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$
Output	Samples with distribution $p(\mathbf{x})$

1. Choose an initial state $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$.
2. Choose a coordinate k equally likely from the coordinates $1, 2, \ldots, n$.
3. Draw a sample x_k from
$$p(x_k|\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{k-1}, \hat{x}_{k+1}, \ldots, \hat{x}_n).$$
4. Set the next state accordingly.

5. Iterate $2-4$ a large number of times. ∎

An alternative version of the Gibbs sampler scans the coordinates sequentially. On iteration j, we start from the current state $\mathbf{x}^{(j)}$, then x_1 is sampled from $p(x_1|x_2^{(j)}, x_3^{(j)}, \ldots, x_n^{(j)})$. using the new value of x_1 a sample x_2 is then made from $p(x_2|x_1^{(j+1)}, x_3^{(j)}, \ldots, x_n^{(j)})$ and so on. One cycle of the algorithm is completed by simulating $x_1, x_2, \ldots, x_n$ from the conditional distributions and recursively refreshing the conditioning variables.

The Gibbs sampler in which the variables are revised in fixed order is defined as follows

Algorithm 6.2	GIBBS SAMPLING 2
Input	Conditional distributions $p(x_i\|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$
Output	Samples with distribution $p(\mathbf{x})$

1. Choose an initial state $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$.

2. **for** $j \leftarrow 1$ **to** M.

3. **for** $k \leftarrow 1$ **to** n.

4. Draw a sample x_k from

$$p(x_k|x_1^{(j+1)}, x_2^{(j+1)}, \ldots, x_{k-1}^{(j+1)}, x_{k+1}^{(j)}, \ldots, x_n^{(j)}).$$

5. Set state k accordingly.

6. Return $\mathbf{x}^{(j)}$ ∎

It can be proved that as the number of iterations grows, the probability distribution of $\mathbf{x}^{(j)}$ tends to $p(\mathbf{x})$, therefore for large values of M we expect the algorithm to return vectors from the target distribution, [Nea93], [Bre95].

We use the latter implementation of the Gibbs sampling in Algorithm 6.2 in our simulations.

There is a generalization for the Gibbs sampling algorithm called multigrid Monte Carlo method (MGMC). Instead of updating one coordinate at time, MGMC suggests moving several highly correlated ones simultaneously, see [LW99], [LS00].

6.2 Estimators for the Partition Function

Similar to Section 2.1, consider a set $\{X_1, X_2, \ldots, X_N\}$ of N discrete *binary* random variables. Let x_i represent the possible realizations of X_i. Let $\mathbf{x}$ stand for $\{x_1, x_2, \ldots, x_N\}$ and $\mathbf{X}$ for $\{X_1, X_2, \ldots, X_N\}$. Suppose $R_1, R_2, \ldots, R_M$ are subsets of $\{1, 2, \ldots, N\}$ and suppose $\mathcal{R} = \{R_1, R_2, \ldots, R_M\}$ is a collection of subsets of the indices of the random variables.

Let $\mathcal{X}$ represent the sample space, and assume that $p(\mathbf{x})$ the joint probability mass function, factors into the product of *positive* and finite local kernels as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{R \in \mathcal{R}} \alpha_R(\mathbf{x}_R), \tag{6.2}$$

where each local kernel $\alpha_R(\mathbf{x_R})$ is a function of the variables whose indices appear in R and Z is the partition function.

Let us define

$$\alpha(\mathbf{x}) \triangleq \prod_{R \in \mathcal{R}} \alpha_R(\mathbf{x}_R), \tag{6.3}$$

$$\gamma \triangleq \frac{1}{Z}. \tag{6.4}$$

Therefore, we can rewrite (6.2) as

$$p(\mathbf{x}) = \frac{1}{Z}\alpha(\mathbf{x}) = \gamma\alpha(\mathbf{x}). \tag{6.5}$$

With these assumptions we can define Z as

$$Z = \sum_{\mathbf{x} \in \mathcal{X}} \alpha(\mathbf{x}). \tag{6.6}$$

We assume that $\alpha(\mathbf{x})$ can be feasibly evaluated for any $\mathbf{x}$.

In the following, we propose some estimators that use samples drawn with different distributions from $\mathcal{X}$ to estimate the value of Z (or γ). In Section 6.2.1, we present estimators that use samples drawn according to different target distributions. The estimators in Section 6.2.2 combine the message passing algorithms (to compute the partition function on trees) with Monte Carlo methods. In Section 6.2.3 we present estimators that use samples drawn uniformly from $\mathcal{X}$.

6.2.1 Estimating with Samples Drawn According to the Target Distribution p(x)

Here we propose estimators for the partition function that use samples drawn from $\mathcal{X}$ according to the target distribution $p(\mathbf{x})$. To generate these samples we use Gibbs sampling explained in Section 6.1.1.

Theorem 6.3. *Let $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(K)}$ be i.i.d. samples drawn with distribution $p(\mathbf{x})$ from $\mathcal{X}$. For the estimators $\tilde{\gamma}$ and $\hat{\gamma}_K$*

$$\tilde{\gamma} = \frac{1}{K|\mathcal{X}|} \sum_{k=1}^{K} \frac{1}{\alpha(\mathbf{x}^{(k)})} \tag{6.7}$$

$$\hat{\gamma}_K = \frac{1}{|\mathcal{X}|^K} \prod_{k=1}^{K} \frac{1}{\alpha(\mathbf{x}^{(k)})} \tag{6.8}$$

we have

$$E[\tilde{\gamma}] = \gamma, \tag{6.9}$$

$$E[\hat{\gamma}_K] = \gamma^K. \tag{6.10}$$

Proof: For $\tilde{\gamma}$ we can write

$$E[\tilde{\gamma}] = \frac{1}{K|\mathcal{X}|} E\left[\sum_{k=1}^{K} \frac{1}{\alpha(\mathbf{x}^{(k)})}\right] \tag{6.11}$$

$$= \frac{1}{K|\mathcal{X}|} \sum_{k=1}^{K} E\left[\frac{1}{\alpha(\mathbf{x}^{(k)})}\right] \tag{6.12}$$

$$= \frac{1}{|\mathcal{X}|} E\left[\frac{1}{\alpha(\mathbf{X})}\right] \tag{6.13}$$

$$= \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \frac{p(\mathbf{x})}{\alpha(\mathbf{x})} \tag{6.14}$$

$$= \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \gamma \tag{6.15}$$

$$= \gamma, \tag{6.16}$$

where (6.12) follows from the linearity of expectation, (6.13) from the fact that the samples are identically distributed, and (6.15) from the definition of γ in (6.4).

Similarly, for $\hat{\gamma}_K$ one can write

$$
\begin{aligned}
E[\hat{\gamma}_K] &= \frac{1}{|\mathcal{X}|^K} E\Big[\prod_{k=1}^{K} \frac{1}{\alpha(\mathbf{x}^{(k)})}\Big] && (6.17)\\
&= \frac{1}{|\mathcal{X}|^K} \prod_{k=1}^{K} E\Big[\frac{1}{\alpha(\mathbf{x}^{(k)})}\Big] && (6.18)\\
&= \left(\frac{1}{|\mathcal{X}|} E\Big[\frac{1}{\alpha(\mathbf{X})}\Big]\right)^K && (6.19)\\
&= \left(\frac{1}{|\mathcal{X}|} \sum_{x\in\mathcal{X}} \frac{p(\mathbf{x})}{\alpha(\mathbf{x})}\right)^K && (6.20)\\
&= \Big(\frac{1}{|\mathcal{X}|} \sum_{x\in\mathcal{X}} \gamma\Big)^K && (6.21)\\
&= \gamma^K, && (6.22)
\end{aligned}
$$

where (6.18) follows from the fact that samples are independent, (6.19) from the fact that samples are identically distributed, and (6.21) from the definition of γ in (6.4). ∎

Remark 6.4. For the estimator $\tilde{\gamma}$ in (6.7), the samples do not have to be independent. Since we only use the linearity of expectation, the same proof can be given if the samples are i.d. (and not necessarily i.i.d.).

Remark 6.5. Since $E[\tilde{\gamma}] = \gamma$, we can say that (6.7) is an unbiased estimator for γ.

For the estimator in (6.7), the following algorithm uses the Gibbs sampling to estimate the value of γ.

Algorithm 6.6	ESTIMATING γ WITH TARGET DISTRIBUTION $p(\mathbf{x})$
Input	Target distribution $p(\mathbf{x}) = \gamma\alpha(\mathbf{x})$
Output	Estimation for γ

1. Choose an initial state $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$.

2. **for** $k \leftarrow 1$ **to** K.

3. **for** $j \leftarrow 1$ **to** n.

4. Draw a sample x_j from

$$p(x_j) = \frac{\alpha(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_{j-1}^{(k+1)}, x_j, x_{j+1}^{(k)}, \ldots, x_n^{(k)})}{\sum_{x_j} \alpha(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_{j-1}^{(k+1)}, x_j, x_{j+1}^{(k)}, \ldots, x_n^{(k)})}.$$

5. Set the next state accordingly.

6. Calculate $\frac{1}{k|\mathcal{X}|} \sum_{i=1}^{k} \frac{1}{\alpha(\mathbf{x}^{(i)})}$ ∎

Simulation results for Algorithm 6.6 on two-dimensional square grids are reported in Section 6.3.

6.2.2 Estimating with Samples Drawn According to the Target Distribution p(z)

In general the probabilistic inference problem defined in Section 2.1 has a graphical representation, in our case junction graphs, that contains cycles. From (2.16), we know that on junction trees there is an efficient message passing algorithm to compute the partition function.

Based on the estimators in Section 6.2.1, here we present new estimators for the partition function that use both Monte Carlo methods and ideas from GDL to estimate the partition function in a more structured manner.

Suppose we partition $\mathbf{x}$ into $(\mathbf{y}, \mathbf{z})$ and write $\alpha(\mathbf{x})$ as

$$\alpha(\mathbf{x}) = \alpha(\mathbf{y}, \mathbf{z}). \tag{6.23}$$

Let us define

$$\alpha(\mathbf{z}) \triangleq \sum_{\mathbf{y}} \alpha(\mathbf{y}, \mathbf{z}). \tag{6.24}$$

We assume that $\mathbf{z}$ is simply a subset of $\mathbf{x}$. Later we will see that a suitable choice of the variables in $\mathbf{z}$ enables us to use the ideas from message passing algorithms on trees in our estimators.

With these assumptions, it is possible to show that the marginal of $p(\mathbf{x})$ with respect to $\mathbf{z}$ has the same partition function as of the global distribution.

$$p(\mathbf{z}) = \sum_{\mathbf{y}} p(\mathbf{y}, \mathbf{z}) \tag{6.25}$$

$$= \gamma \sum_{\mathbf{y}} \alpha(\mathbf{y}, \mathbf{z}) \tag{6.26}$$

$$= \alpha p(\mathbf{z}). \tag{6.27}$$

Let us also suppose that $\mathcal{Z}$ represents the sample space for $p(\mathbf{z})$. We propose estimators for the partition function that use samples drawn from $\mathcal{Z}$ according to $p(\mathbf{z})$. To generate these samples we use Gibbs sampling explained in Section 6.1.1.

Theorem 6.7. *Let $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(K)}$ be i.i.d. samples drawn with distribution $p(\mathbf{z})$ from $\mathcal{Z}$. For the estimators $\tilde{\gamma}$ and $\hat{\gamma}_K$*

$$\tilde{\gamma} = \frac{1}{K|\mathcal{Z}|} \sum_{k=1}^{K} \frac{1}{\alpha(\mathbf{z}^{(k)})} \tag{6.28}$$

$$\hat{\gamma}_K = \frac{1}{|\mathcal{Z}|^K} \prod_{k=1}^{K} \frac{1}{\alpha(\mathbf{z}^{(k)})} \tag{6.29}$$

we have

$$E[\tilde{\gamma}] = \gamma \tag{6.30}$$

$$E[\hat{\gamma}_K] = \gamma^K. \tag{6.31}$$

Remark 6.8. For the estimator $\tilde{\gamma}$ in (6.28), the samples do not have to be independent. Since we only use the linearity of expectation, the same proof can be given if the samples are i.d. (and not necessarily i.i.d.)

Remark 6.9. Since $E[\tilde{\gamma}] = \gamma$, we can say that (6.28) is an unbiased estimator for γ.

For the estimator in (6.28), the following algorithm uses the Gibbs sampling to estimate the value of γ.

Algorithm 6.10	ESTIMATING γ WITH TARGET DISTRIBUTION $p(\mathbf{z})$
Input	Target distribution $p(\mathbf{z}) = \gamma\alpha(\mathbf{z})$
Output	Estimation for γ

1. Choose an initial state $\mathbf{z}^{(0)} = (z_1^{(0)}, z_2^{(0)}, \ldots, z_n^{(0)})$.

2. **for** $k \leftarrow 1$ **to** K.

3. **for** $j \leftarrow 1$ **to** n.

4. Draw a sample z_j from

$$p(z_j) = \frac{\sum_{\mathbf{y}} \alpha(\mathbf{y}, z_1^{(k+1)}, z_2^{(k+1)}, \ldots, z_{j-1}^{(k+1)}, z_j, z_{j+1}^{(k)}, \ldots, z_n^{(k)})}{\sum_{\mathbf{y}, z_j} \alpha(\mathbf{y}, z_1^{(k+1)}, z_2^{(k+1)}, \ldots, z_{j-1}^{(k+1)}, z_j, z_{j+1}^{(k)}, \ldots, z_n^{(k)})}.$$

5. Set the next state accordingly.

6. Calculate $\frac{1}{k|\mathcal{Z}|} \sum_{i=1}^{k} \frac{1}{\alpha(\mathbf{z}^{(i)})}$ ∎

In Algorithm 6.10, in order to draw samples according to $p(z_j)$, the following computation is done in each iteration.

$$p(z_j) = \frac{\sum_{\mathbf{y}} \alpha(\mathbf{y}, z_1^{(k+1)}, z_2^{(k+1)}, \ldots, z_{j-1}^{(k+1)}, z_j, z_{j+1}^{(k)}, \ldots, z_n^{(k)})}{\sum_{\mathbf{y}, z_j} \alpha(\mathbf{y}, z_1^{(k+1)}, z_2^{(k+1)}, \ldots, z_{j-1}^{(k+1)}, z_j, z_{j+1}^{(k)}, \ldots, z_n^{(k)})}. \tag{6.32}$$

The question is whether by carefully choosing the variables in $\mathbf{z}$, the calculation in (6.32) can be done efficiently.

Assume that for each instance of $\mathbf{z}$ the probabilistic inference problem defined in (6.5) has a junction-tree representation. With this assumption the following sum is basically the partition function of the underlying junction tree

$$\sum_{\mathbf{y}} \alpha(\mathbf{y}, z_1^{(k+1)}, z_2^{(k+1)}, \ldots, z_{j-1}^{(k+1)}, z_j, z_{j+1}^{(k)} \ldots, z_n^{(k)}). \tag{6.33}$$

Therefore the sum in (6.33) can be calculated efficiently with message passing algorithms on trees, for example GDL on junction trees and using equation (2.16).

In the following examples, we show how a clever choice of $\mathbf{z}$ leaves the underlying graph cycle-free.

Example 6.11. *Consider a* 5×5 *two-dimensional grid with junction graph representation in Fig. 6.1. Here* $\mathbf{z}$ *consists of* 6 *variables, namely* $x_3, x_7, x_9, x_{13}, x_{17}, x_{19}$*, out of* 25 *variables. Each instance of* $\mathbf{z}$ *leaves the underlying graph cycle free and with a junction tree representation.*

Example 6.12. *On a* 9×9 *two-dimensional grid there are* 81 *variables and* $\mathbf{z}$ *consists of* 26 *variables. See Fig. 6.2.*

Remark 6.13. In the above examples, 6 is the minimum number of variables in $\mathbf{z}$ on a 5×5 grid, so is 26 for a 9×9 grid.

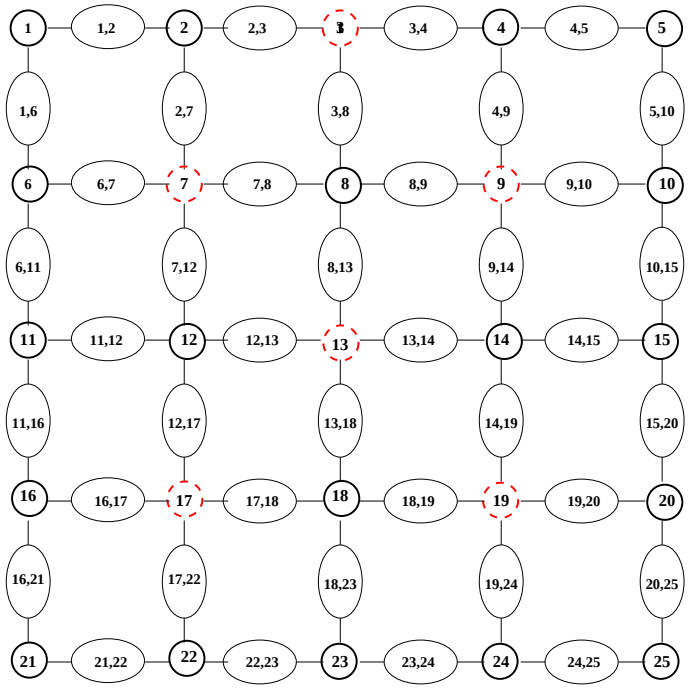

Figure 6.1: On a 5×5 Grid $\mathbf{z}$ contains 6 Variables

Remark 6.14. Imagine that for a particular graph configuration, a certain choice of $\mathbf{z}$ leaves the remaining graph cycle-free but not necessarily connected. In this case, the remaining graph would be a disjoint union of junction trees. The partition function of this graph is the multiplication of the partition functions of all its sub-junction trees and still can be computed efficiently with message passing algorithms. This is the case for both examples abovefe on grids.

Simulation results for Algorithm 6.10 on two-dimensional square grids are reported in Section 6.3.

6.2.3 Estimating Using Uniform Sampling

In Section 6.1, we discussed the difficulty of sampling from $p(\mathbf{x})$. In order to circumvent this problem one might draw samples *uniformly* from $\mathcal{X}$.

Here we propose estimators for the partition function that use samples drawn uniformly from $\mathcal{X}$.

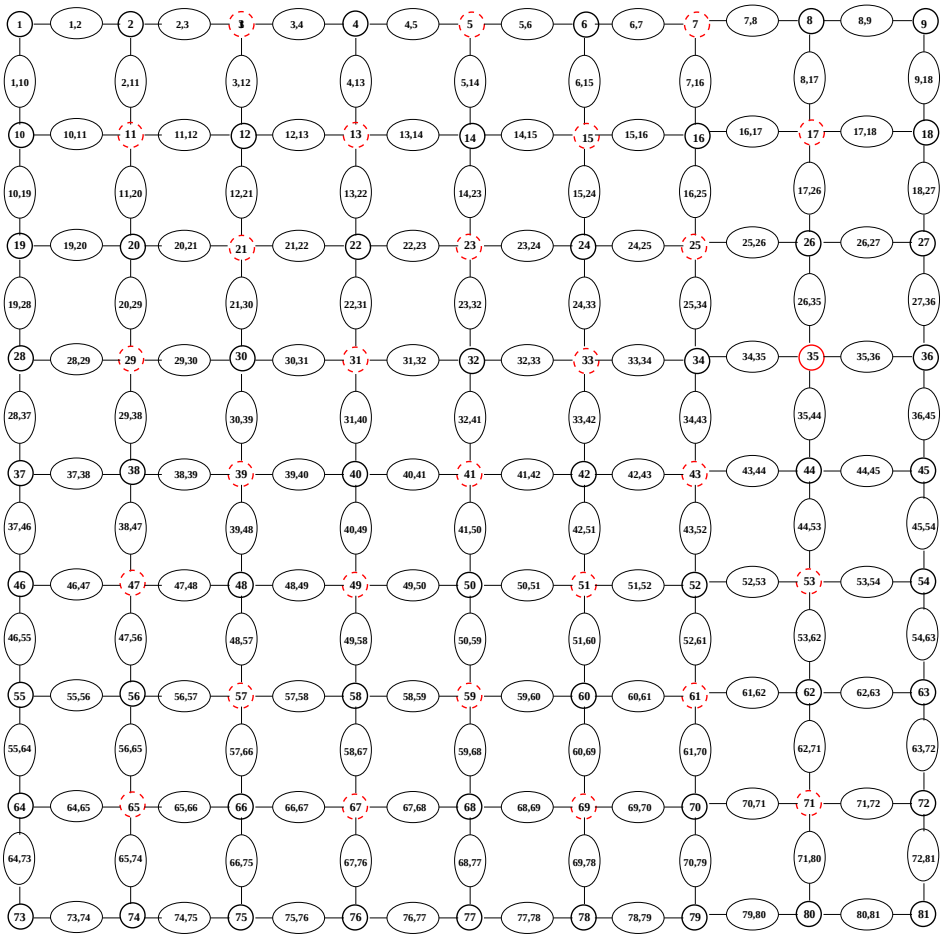

Figure 6.2: On a 9×9 Grid $\mathbf{z}$ contains 26 Variables

Theorem 6.15. *Let $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(K)}$ be i.i.d. samples drawn uniformly from $\mathcal{X}$. For the estimators $\tilde{Z}$ and $\hat{Z}$*

$$\tilde{Z} = \frac{|\mathcal{X}|}{K} \sum_{k=1}^{K} \alpha(\mathbf{x}^{(k)}) \tag{6.34}$$

$$\hat{Z}_K = |\mathcal{X}|^K \prod_{k=1}^{K} \alpha(\mathbf{x}^{(k)}) \tag{6.35}$$

we have

$$E[\tilde{Z}] = Z \tag{6.36}$$
$$E[\hat{Z}_K] = Z^K. \tag{6.37}$$

Proof: For $\tilde{Z}$ we can write

$$\begin{aligned} E[\tilde{Z}] &= \frac{|\mathcal{X}|}{K} E\Big[\sum_{k=1}^{K} \alpha(\mathbf{x}^{(k)})\Big] && (6.38)\\ &= \frac{|\mathcal{X}|}{K} \sum_{k=1}^{K} E\Big[\alpha(\mathbf{x}^{(k)})\Big] && (6.39)\\ &= |\mathcal{X}| E\big[\alpha(\mathbf{X})\big] && (6.40)\\ &= |\mathcal{X}| \sum_{x\in\mathcal{X}} \frac{\alpha(\mathbf{x})}{|\mathcal{X}|} && (6.41)\\ &= Z, && (6.42) \end{aligned}$$

where (6.39) follows from the linearity of expectation, (6.40) from the fact that the samples are identically distributed, and (6.41) from the fact that samples are drawn uniformly.

Similarly, for $\hat{Z}_K$ one can write

$$\begin{aligned} E[\hat{Z}_K] &= |\mathcal{X}|^K E\Big[\prod_{k=1}^{K} \alpha(\mathbf{x}^{(k)})\Big] && (6.43)\\ &= |\mathcal{X}|^K \prod_{k=1}^{K} E\Big[\alpha(\mathbf{x}^{(k)})\Big] && (6.44)\\ &= \Big(|\mathcal{X}| E\big[\alpha(\mathbf{X})\big]\Big)^K && (6.45)\\ &= \Big(|\mathcal{X}| \sum_{x\in\mathcal{X}} \frac{\alpha(\mathbf{x})}{|\mathcal{X}|}\Big)^K && (6.46)\\ &= Z^K, && (6.47) \end{aligned}$$

where (6.44) follows from the fact that samples are independent, (6.45) from the fact that samples are identically distributed, and (6.46) from the fact that samples are drawn uniformly. ∎

Remark 6.16. For the estimator $\tilde{Z}$ in (6.34), the samples do not have to be independent. Since we only use the linearity of expectation, the same proof can be given if the samples are i.d. (and not necessarily i.i.d.)

Here we propose estimators for the partition function that use samples drawn from $\mathcal{Z}$.

Theorem 6.17. *Let $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(K)}$ be i.i.d. samples drawn uniformly from $\mathcal{Z}$. For the estimator $\tilde{Z}$.*

$$\tilde{Z} = \frac{|\mathcal{Z}|}{K} \sum_{k=1}^{K} \alpha(\mathbf{z}^{(k)}) \tag{6.48}$$

$$\hat{Z}_K = |\mathcal{Z}|^K \prod_{k=1}^{K} \alpha(\mathbf{z}^{(k)}) \tag{6.49}$$

we have

$$E[\tilde{Z}] = Z \tag{6.50}$$

$$E[\hat{Z}_K] = Z^K. \tag{6.51}$$

Remark 6.18. For $K = |\mathcal{X}|$ and $K = |\mathcal{Z}|$, i.e. if we sample from the whole space (without repetition), the estimators in (6.34) and (6.48) eventually calculate the *exact* value of the partition respectively.

Remark 6.19. Since $E[\tilde{Z}] = Z$, we can say that (6.34) and (6.48) are unbiased estimators for Z.

As mentioned in Section 6.1, typically the high-dimensional probability distributions that we deal with in practice are concentrated in regions of the state space that occupy a tiny fraction of the whole. So uniform estimators will only give good outputs if the number of samples K is sufficiently large that we are likely to hit these regions. In the cases that the global distribution of a model tends to a uniform distribution (for small values of t for our set-up in Section 6.3), uniform sampling might be an option. But in general it gives poor results in practice, see [Mac03], [Nea93].

We only used the estimator in (6.48) on 5×5 two-dimensional square grids to compute the exact value of the partition function. Simulation results are reported in Section 6.3.

6.3 Simulation Results

Similar to Section 5.3, we use the standard Ising model of statistical physics for our simulations. Ising model is a well-known and extensively researched model in statistical physics.

For an Ising model the global probability distribution can be expressed as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{s \in V} e^{\beta_s x_s} \prod_{(s,t) \in E} e^{\beta_{st} x_s x_t}, \tag{6.52}$$

where β_s is the node parameter for node s and β_{st} is the strength of edge (s, t).

For each node, we chose β_s uniformly and independently in $[-0.025, 0.025]$. And for each edge, $\beta_{i,j}$ is chosen uniformly and independently in $[0, t]$. We considered only two values, 0.5 and 1 for t. Same as Section 5.3, we suppose binary random variables and consider the spin representation where variables take their values in $\{-1, +1\}$.

We ran our simulations on 5×5 square grids with $N = 25$ variables, and 9×9 square grids with $N = 81$ variables for one instance of the Ising model defined in (6.52).

In the following, we report simulation results for the estimators in (6.7), (6.28), and (6.48) for two sizes of grids and two different edge strengths.

6.3.1 5×5 Grids

Shown in Fig. 6.3 are plots of the estimator in (6.48) for both values of $t = 0.5$ and $t = 1$. In both cases it is possible to compute the exact value of the Z with 64 samples, see remark 6.18. For the instance of the Ising model in our simulations, the exact value of $\ln(Z)$ is 19.759862 for $t = 0.5$, and 25.717765 for $t = 1$.

Shown in Figs. 6.4 and 6.5 are plots for the estimators in (6.7) and (6.28) respectively. For both estimators we set $t = 0.5$ and use 10^4 samples. We plot $\ln(\tilde{Z})$ versus number of samples for five random initial states.

Shown in Figs. 6.6 and 6.7 are plots for the estimators in (6.7) and (6.28) respectively. For both estimators we set $t = 1$. We use 10^6 samples for the estimator in (6.7) and 10^4 samples for the estimator in (6.28). The plots show $\ln(\frac{1}{\tilde{\gamma}})$ versus number of samples for five random initial states.

6.3.2 9×9 Grids

Shown in Figs. 6.8 and 6.9 are plots for the estimators in (6.7) and (6.28) respectively. For both estimators $t = 0.5$. We use 10^7 samples for the estimator in (6.7)

and $5 \cdot 10^4$ samples for (6.28). We plot $\ln(\frac{1}{\tilde{\gamma}})$ versus number of samples for five different random initial states.

Shown in Figs. 6.10 and 6.11 are plots for the estimators in (6.7) and (6.28) respectively. For both estimators $t = 1$. We use 10^8 samples for the estimator in (6.7) and about 10^6 samples for the estimator in (6.28). The plots show $\ln(\frac{1}{\tilde{\gamma}})$ versus number of samples for five random initial states.

6.4 Discussion

The simulation results on 5×5 and 9×9 two-dimensional grids show that if we use $p(\mathbf{z})$ as the target distribution and use Algorithm 6.10, we observe faster convergence and less variance compared to Algorithm 6.6 - specially for bigger values of t. The number of samples needed in Algorithm 6.10 for convergence is typically 10^2 to 10^4 times less than the number of samples needed in Algorithm 6.6. However, in Algorithm 6.10 to generate each sample, we need to run the GDL on a sub-junction tree of the grid. It is also important to notice that the correlation between the variables in $\mathbf{z}$ is smaller than the set of all variables in $\mathbf{x}$, since the variables in $\mathbf{z}$ are not directly involved with each other in the kernels of the junction graph. Also by considering only the varibles in $\mathbf{z}$ the dimensionality of the sample space reduces.

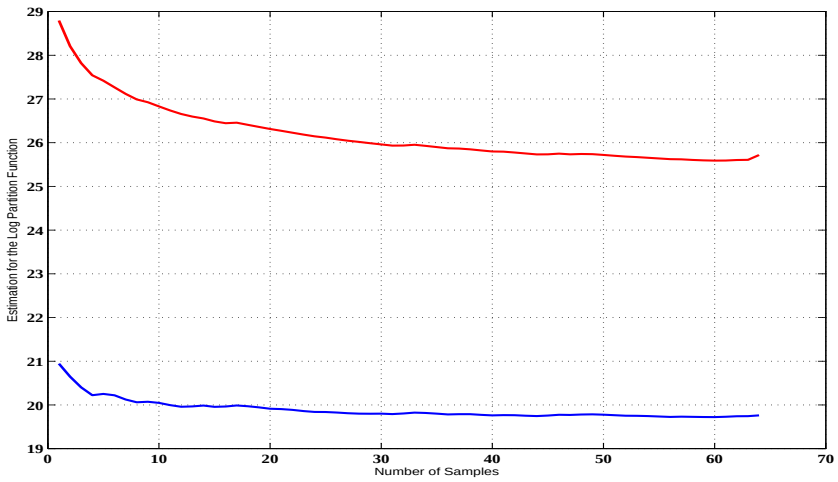

Figure 6.3: Exact value of $\log(Z)$ on a 5×5 grid for $t = 0.5$ and 1

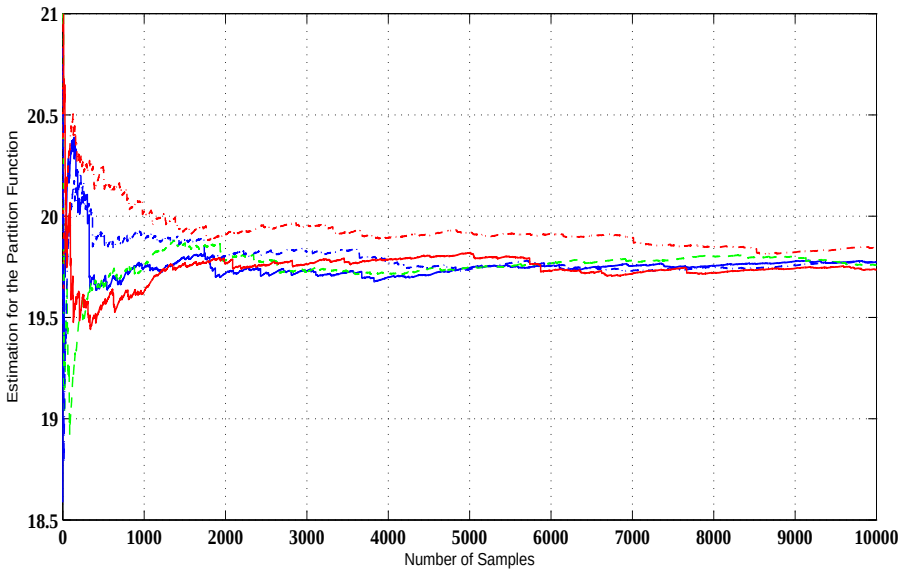

Figure 6.4: Estimation of $\log(Z)$ on a 5×5 grid with (6.7) where $t = 0.5$

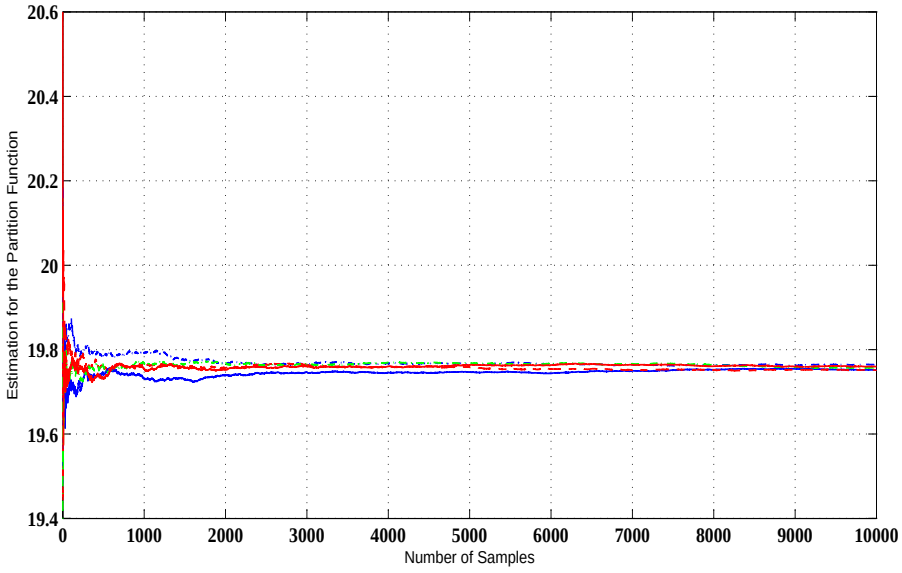

Figure 6.5: Estimation of $\log(Z)$ on a 5×5 grid with (6.28) where $t = 0.5$

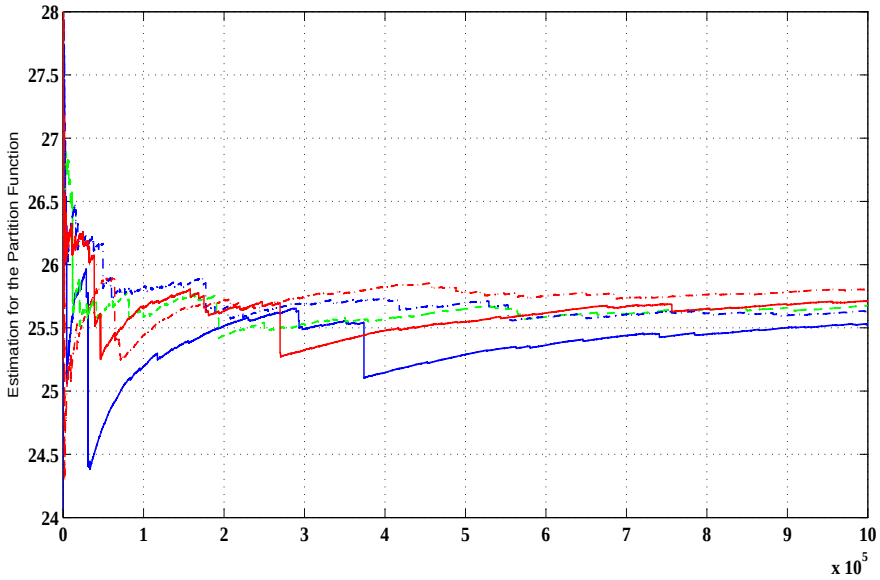

Figure 6.6: Estimation of $\log(Z)$ on a 5×5 grid with (6.7) where $t = 1$

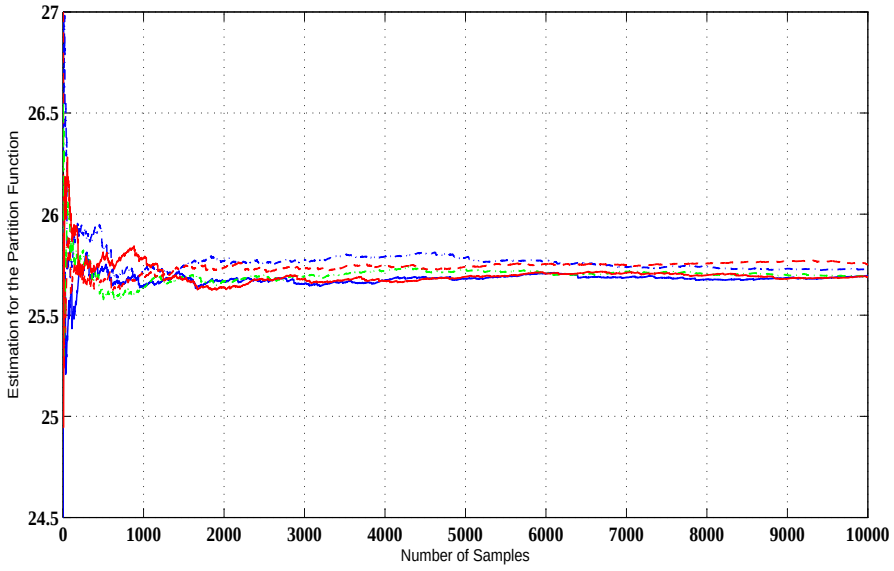

Figure 6.7: Estimation of $\log(Z)$ on a 5×5 grid with (6.28) where $t = 1$

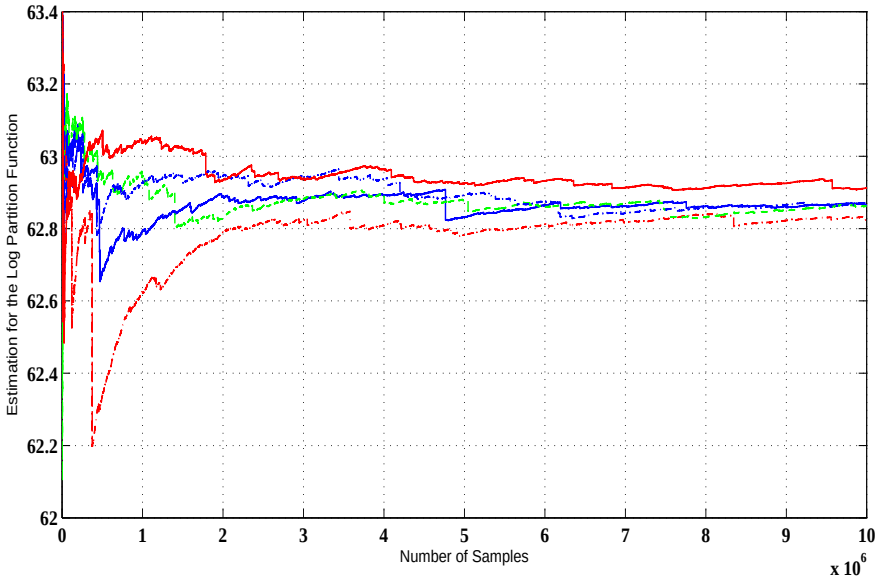

Figure 6.8: Estimation of $\log(Z)$ on a 9×9 grid with (6.7) where $t = 0.5$

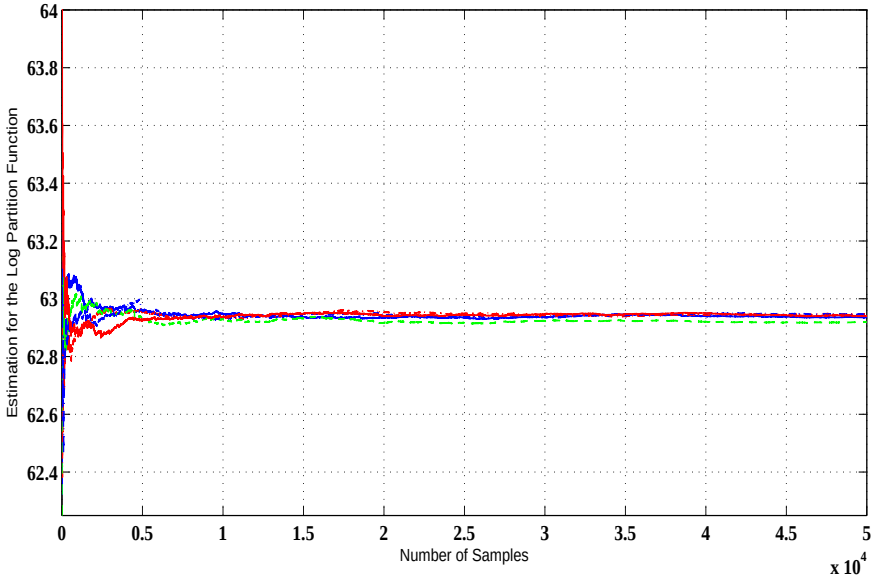

Figure 6.9: Estimation of $\log(Z)$ on a 9×9 grid with (6.28) where $t = 0.5$

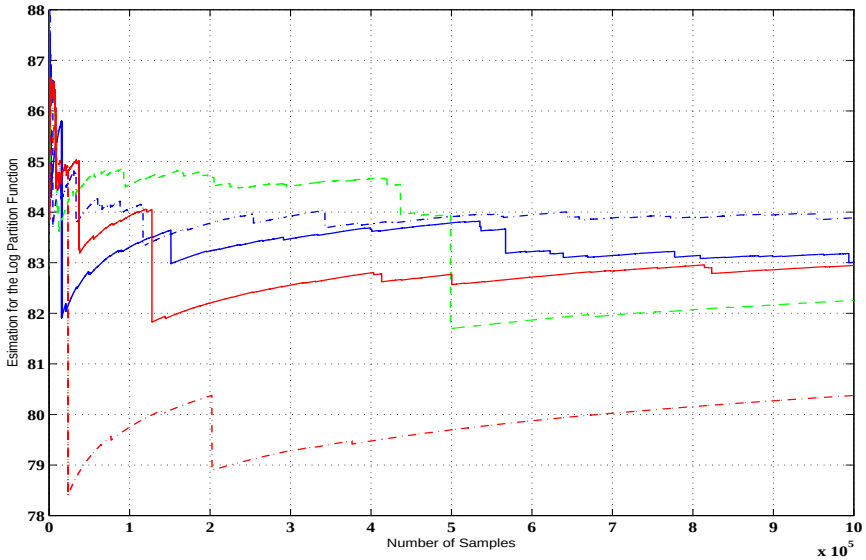

Figure 6.10: Estimation of $\log(Z)$ on a 9×9 grid with (6.7) where $t = 1$

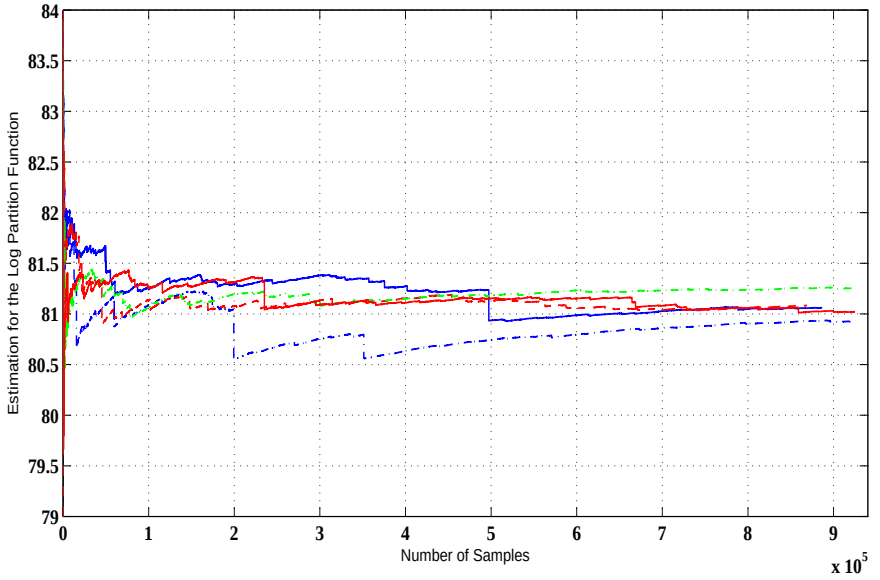

Figure 6.11: Estimation of $\log(Z)$ on a 9×9 grid with (6.28) where $t = 1$

Bibliography

[AM00] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000. (Cited on pages 9, 11, 12, 13 and 14.)

[AM01] S. M. Aji and R. J. McEliece. The generalized distributive law and free energy minimization. In *Proceedings of Allerton Conference*, pages 672–681, 2001. (Cited on pages 3, 10 and 20.)

[Bre95] P. Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulations, and Queues*. Springer, 1995. (Cited on page 43.)

[Cow98] R. Cowell. *Advanced Inference in Bayesian Networks*. MIT Press, 1998. (Cited on page 13.)

[CS97] M. Chen and Q. Shao. On Monte Carlo methods for estimating ratios of normalizing constants. *The Annals of Stat.*, pages 1563–1594, 1997. (Cited on page 40.)

[Dau06] J. Dauwels. *On Graphical Models for Communications and Machine Learning: Algorithms, Bounds, and Analog Implementation*. PhD thesis, ETHZ, 2006. (Cited on page 7.)

[DLMO04] J. Dauwels, H.-A. Loeliger, P. Merkli, and M. Ostojicc. On Markov-structured summary propagation and LFSR synchronization. In *Proceedings of Allerton Conference*, pages 672–681, 2004. (Cited on page 22.)

[For01] G. D. Forney. Codes on graphs: Normal realizations. *IEEE Transactions on Information Theory*, 47(2):520–548, 2001. (Cited on page 9.)

[Fre04] D. Frenkel. Introduction to Monte Carlo methods. *Computational Soft Matter*, 2004. (Cited on page 5.)

[GG84] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and Bayesian restoration of images. *IEE Transactions on Pattern Analysis and Machine Intelligence*, pages 721–741, 1984. (Cited on page 42.)

[GS90] A. E. Gelfand and A. F. Smith. Sampling-based approaches to calculating marginal densities. *Journal Amer. Stat. Sci.*, pages 457–472, 1990. (Cited on page 42.)

[JJ96] T.S. Jaakkola and M. I. Jordan. Recursive algorithms for approximating probabilities in graphical models. In *NIPS 9*, pages 487–493, 1996. (Cited on page 23.)

[JS93] M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for Ising model. *SIAM Journal of Computing*, 22:1087–1116, 1993. (Cited on pages 23 and 40.)

[KFL01] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001. (Cited on pages 9 and 14.)

[LK01] M.A.R. Leisink and H.J. Kappen. A tighter bound for graphical models. In *NIPS 13*, pages 266–272, 2001. (Cited on page 23.)

[LS00] J. S. Liu and C. Sabatti. Generalized gibbs sampler and multigrid Monte Carlo for bayesian computation. *Biometrika*, 87:353–369, 2000. (Cited on page 43.)

[LW99] J. S. Liu and Y. N. Wu. Parameter expansion scheme for data augmentation. *Journal Amer. Stat. Assoc.*, 94:1264–1274, 1999. (Cited on page 43.)

[Mac03] D.J.C. Mackay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. (Cited on pages 41 and 53.)

[MM06] M. Mezard and A. Montanari. *Constraint Satisfaction Networks in Physics and Computation*. Clarendon Press, 2006. (Cited on pages 15 and 17.)

[MP05] M. Molkaraie and P. Pakzad. On entropy decomposition and new bounds on the partition function. In *Proceedings of the International Symposium on Information Theory*, 2005. (Cited on pages 22, 25 and 26.)

[MP06] M. Molkaraie and P. Pakzad. Sub-tree based upper and lower bounds on partition function. In *Proceedings of the International Symposium on Information Theory*, 2006. (Cited on page 25.)

[MRR+53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. N. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal Chem. Phys.*, pages 1087–1092, 1953. (Cited on page 41.)

[Mur01] K. P. Murphy. An introduction to graphical models. Available online at ⟨http://www.cs.ubc.ca/~murphyk/Papers/intro_gm.pdf⟩, 2001. (Cited on page 9.)

[Nea93] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. 1993. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Waterloo. (Cited on pages 15, 41, 43 and 53.)

[PA04] P. Pakzad and V. Anantharam. A new look at the generalized distributive law. *IEEE Transactions on Information Theory*, 50(6):1132–1155, 2004. (Cited on page 14.)

[PA05] P. Pakzad and V. Anantharam. Estimation and marginalization using Kikuchi approximation methods. *Neural Computation*, 46(2):1836–1873, 2005. (Cited on page 20.)

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988. (Cited on page 13.)

[PG93] G. Potamianos and J. Goutsias. Partition function estimation of Gibbs random field images using Monte Carlo simulations. *IEEE Transactions on Information Theory*, 39(4):1322–1332, 1993. (Cited on page 40.)

[RS95] W. R. Gilks S. Richardson and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1995. (Cited on page 41.)

[SS90] G. R. Shafer and P. P. Shenoy. Probability propagation. In *Annals Math. Art. Intel*, pages 327–352, 1990. (Cited on page 12.)

[WF01] Y. Weiss and W. T. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):736–744, 2001. (Cited on page 9.)

[WJ93] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. 1993. Technical Report 649, Dept. of Statistics, University of California Berkeley. (Cited on pages 7, 9 and 22.)

[WJW05] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7):2313–2335, 2005. (Cited on pages 22, 23 and 36.)

[YFW05] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005. (Cited on pages 14, 15, 17, 20 and 23.)

[Zha96] J. Zhang. The application of the Gibbs-Bogoliubov-Feynmann inequality in mean-field calculations for Markov random fields. *IEEE Transactions on Image Processing*, 5(7):1208–1214, 1996. (Cited on page 23.)

Mehdi Molkaraie

Ave. de Béthusy 21
CH-1005 Lausanne
Switzerland

Date of birth: July 4, 1976
Place of Birth: Tehran, Iran

Tel (professional): +41 (21) 693 1240
Tel (private): +41 (79) 661 2363
Email: meft@ieee.org

Research Interests:

Information Theory, Coding Theory, Graphical Models.

Education:

- 2004-2007: Ecole Polytechnique Fédérale de Lausanne, Switzerland, Ph.D.
- 1999-2002: University of Tehran, Iran, MS in Telecommunications.
- 1994-1999: University of Tehran, Iran, BS in Electrical Engineering.

Publications:

- M. Molkaraie, P. Pakzad, "On Properties of the Minimum Entropy Sub-tree to Compute Lower Bounds on Partition Function," *preprint*, 2007.
- M. Molkaraie, P. Pakzad, "Sub-tree Based Upper and Lower Bounds on the Partition Function,"*International Symposium on Information Theory*, Seattle, 2006.
- M. Molkaraie, P. Pakzad, "On Entropy Decomposition and New Bounds on the Partition Function,"*International Symposium on Information Theory*, Adelaide, 2005.
- O. Etesami, M. Molkaraie and A. Shokrollahi, "Raptor Codes on Symmetric Channels," *International Symposium on Information Theory*, Chicago, 2004.
- M. Molkaraie, S. Boroojerdian, "Information Theory and Approximations to Persian Texts," *Iranian Journal of Linguistics*, Tehran, 2001.