LOW COMPLEXITY, HIGH PERFORMANCE ALGORITHMS FOR ESTIMATION AND DECODING

by

Payam Pakzad

Memorandum No. UCB/ERL M04/20

1 May 2004

LOW COMPLEXITY, HIGH PERFORMANCE ALGORITHMS FOR ESTIMATION AND DECODING

by

Payam Pakzad

Memorandum No. UCB/ERL M04/20

1 May 2004

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Low Complexity, High Performance Algorithms for Estimation and Decoding

by

Payam Pakzad

B.S. (California Institute of Technology) 1998
M.S. (University of California, Berkeley) 2001

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering
and Computer Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Venkatachalam Anantharam, Chair
Professor Michael Jordan
Professor Bernd Sturmfels

Spring 2004

Low Complexity, High Performance Algorithms for Estimation and Decoding

by

Payam Pakzad

Abstract

Low Complexity, High Performance Algorithms for Estimation and Decoding

by

Payam Pakzad

Doctor of Philosophy in Engineering – Electrical Engineering
and Computer Science

University of California, Berkeley

Professor Venkatachalam Anantharam, Chair

The tremendous success of the graphical error-correcting codes has generated great interest in iterative message-passing algorithms for approximate decoding and estimation; indeed the applications of such approximation techniques extend to a surprisingly vast range of fields of science and engineering, from thermal physics to artificial intelligence, and from computer vision to communications and information theory. In this thesis we will discuss two novel approaches to the general estimation problem based on graphs; each approach results in methods which are superior, in terms of performance and/or complexity, to the existing algorithms that solve the same problem exactly or approximately.

In the first approach we reformulate the general estimation problem in a measure-theoretic framework, free of the notion of 'variables', upon which the conventional 'junction tree' based methods heavily rely. This enables us to exploit all the underlying structure in the data of the problem, in order to reduce the complexity of the underlying marginalization task. We introduce appropriate notions of independence and junction trees, and the corresponding measure-theoretic junction tree algorithm, and we give an automatic procedure, called 'lifting', that finds such low complexity junction trees for a given marginalization problem. This automatic procedure often gives rise to algorithms that are substantially less complex than those obtained using

the conventional methods.

The second approach is based on a recently discovered connection between the fixed points of the well-known loopy belief propagation algorithm, and certain 'variational free energy' concepts from statistical physics. We discuss the general 'Kikuchi approximation method' for solving a marginalization problem. We define appropriate concepts of graphs and minimal graphs representing such problems, and derive several results on the strengths and limitations of the Kikuchi method based on the graph-theoretic properties of these graphs.

In addition to theoretical discussions, for each approach we provide detailed examples and simulation results on several applications of interest, such as exact decoding of low-density parity-check (LDPC) codes, and joint decoding of a partial response magnetic recording channel and an LDPC code.

Acknowledgements

Throughout my academic life, I have been the fortunate recipient of contributions, both academically and personally, from many individuals, whose help has alleviated the long process of completing a doctoral degree.

I am most grateful to my advisor, Prof. Venkat Anantharam for his support and encouragement. His unique insight, and the vast scope of his knowledge in engineering and mathematics have been a source of intriguing research ideas and directions. I am also thankful to Prof. Bora Nikolić for the opportunity to learn about the magnetic recording applications; and to my dissertation committee members, Profs. Michael Jordan and Bernd Sturmfels for their interest, support and feedback.

I would like to thank my wife, Irina, without whose love and support I would not have accomplished this feat. My parents, Houshang and Farzin have been constant sources of inspiration for me; their love and encouragement have carried me through difficult times and against the odds. I have been fortunate to have my brother and sister-in-law, Shamim and Niloofar, and my younger brother, Pooya live close-by for most of my stay at Berkeley. Their love and assistance has made the experience of these years quite pleasurable. My aunt and uncle, Mrs. Akhtar Mobini and Mr. Naghi Mobini, and my cousins Kambiz and Sima have been my family away from home, especially during my undergraduate years in southern California. I am grateful for their constant support.

My good friend, Brian, who was one of the first people I met at Caltech, has been a great source of information on most things related to computers and the internet. Engling, whom I consider a true jack-of-all-trades (and even a master-of-a-few), has been a fantastic friend and colleague. I would also like to thank Dr. Dimitrios Antsos for all his help throughout the years. Finally, I would like to thank Ruth, Mary, and the other friendly and helpful staff of the EE graduate office here at U.C. Berkeley.

Payam Pakzad

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Overview

The ultimate objective of the communication theory is reliable transfer of information in presence of noise. To achieve this objective, the information is typically first encoded using an error-correcting code, suitably designed for the specific noisy channel of communication. The noisy observations are then used to decode the original information. Probability theory gives a convenient framework to deal with this problem. First, the noisy channel is modelled by the joint probability distribution of its inputs and outputs; the challenge involved here is to devise models, which are both accurate enough to represent the underlying physical system, and simple enough to allow for efficient analysis. Next, a code is designed to combat the channel noise by introduction of redundancy; once again, in general there is a trade-off between the error-correcting capabilities of the code and the complexity of the encoding and decoding process. The optimal decoding is then performed using the *maximum a posteriori* (MAP) method, in which the configuration with the highest probability given the observations, is chosen.

More generally, an *estimation* problem is that of finding the best 'estimates', in the sense of minimizing the probability of error, of a random variable given a body of observations. This is also known as the *probabilistic inference* problem. Applications of the probabilistic inference problem range from medical diagnosis to image processing, and from speech recognition to error-correcting codes.

The fundamental problem with the straightforward approach to the inference

problem is that computing the optimal estimates can be prohibitively complex. Real systems have myriads of states, and unless an appropriate form of 'modularity' is introduced, the probabilistic calculations will not be feasible to perform. Graphical models are an attractive way to represent such 'modular' models, and the corresponding efficient estimation algorithms are described in form of local message-passing algorithms on such graphs.

Graphical models and message-passing algorithms on graphs have seen a resurgence of interest in the communications and coding communities because of the success of the recently invented turbo codes [6] and the older low density parity check (LDPC) codes [20] which use such decoding algorithms. They have also long been of interest to the artificial intelligence community, see e.g. [33; 10]. In fact, 'codes on graphs' have been so successful that they have fundamentally changed the face of the coding theory, effectively replacing the more complex algebraic codes as the state-of-the-art error-correcting codes. Using iterative message-passing decoding algorithms, it is now possible to decode, in real time, LDPC codes with block-lengths of *hundreds of thousands* of bits on an average personal computer, achieving rates very close to the Shannon capacity, see e.g. [9]; decoding good codes with such large block sizes were unthinkable just over a decade ago.

In spite of these success stories, the conventional graphical methods have their own shortcomings, and there is always room for improvement. In this document we discuss two novel approaches to the general estimation problem based on graphical models; each approach results in methods, which are superior, in terms of performance and/or complexity, to the existing algorithms that solve the same problem exactly or approximately.

The rest of this chapter is devoted to a brief overview of the graphical models and the corresponding message-passing algorithms, which will serve as the background for our main discussion in Parts I and II. An outline of the rest of this dissertation is given in Section 1.3.

1.1 Marginalization Problems

In this section we formulate the central estimation problem addressed in this document, as one of two types of marginalization problems: marginalizing a product function (MPF), or the more specialized problem of marginalizing a product distribution (MPD).[1] Part I of this document discusses a method to find exact solutions to an MPF problem, while Part II deals with finding exact or approximate solutions to an MPD problem.

Let $\mathbf{x} := (x_1, \cdots, x_N)$, where for each $i \in [N] := \{1, \cdots, N\}$, x_i is a variable taking value in the discrete finite set $[q_i] := \{1, \cdots, q_i\}$, with $q_i \geq 2$. For each subset $s \subseteq [N]$, we denote by $\mathbf{x}_s \in \prod_{i \in s}[q_i]$ the vector comprised of the variables, whose indices appear in s.

Let R be a collection of subsets of $[N]$; we call each $r \in R$ a *local domain* (usually in the context of an MPF problem), or a *region* (in the context of an MPD problem). We assume that each variable index $i \in [N]$ appears in at least one such $r \in R$.

Associated with each region $r \in R$ is a *local kernel function*, $\alpha_r(\mathbf{x}_r)$, depending only on the variables that appear in r. Then the corresponding *R-decomposable product function* is defined as

$$\Pi(\mathbf{x}) := \prod_{r \in R} \alpha_r(\mathbf{x}_r) \tag{1.1}$$

For a subset $s \subset [N]$, we define $\Pi_r(\mathbf{x}_r) := \sum_{\mathbf{x}_{[N]\backslash r}} \Pi(\mathbf{x})$ as the *r-marginal* of $\Pi(\mathbf{x})$, where the summation is taken over the complete range of the underlying vector, i.e. $\mathbf{x}_{[N]\backslash r} \in \prod_{i \in [N]\backslash r}[q_i]$.

Problem 1.1 (Marginalizing a product function (MPF)). *For one or more local domains $r \in R$, find the r-marginal $\Pi_r(\mathbf{x}_r)$.* □

In estimation, one is usually interested in the probability distribution which is induced by the local kernels. Assuming that all the local kernels are nonnegative, we

[1] These formulations and the corresponding notation are, for the most part, adopted from [2].

define the corresponding *R-decomposable* (*Boltzmann*) *product distribution* as

$$B(\mathbf{x}) := \frac{1}{Z} \prod_{r \in R} \alpha_r(\mathbf{x}_r) \tag{1.2}$$

Here $Z := \sum_{\mathbf{x}} \prod_{r \in R} \alpha_r(\mathbf{x}_r)$ is the normalizing constant and is called the *partition function.*

Problem 1.2 (Marginalizing a product distribution (MPD)). *For one or more regions $r \in R$, find the r-marginal $B_r(\mathbf{x}_r)$, as well as the partition function Z.* □

It is evident that the MPD problem can be viewed as a special case of the MPF problem, where the partition function Z is viewed as a marginal, $\Pi_\emptyset$, of the product function, and where other marginals of $B(\mathbf{x})$ are obtained from the corresponding marginals of $\Pi(\mathbf{x})$ after dividing them by Z. However, because of the importance of the problems that fall in this category, we have chosen to define MPD as a separate class of problems.

1.2 Graphical Models

Although easy to state, the marginalization problems defined in the previous section are fundamentally hard to solve. The obvious reason is that the underlying state space is exponentially large in the number of the variables, and the straightforward 'brute force' ways to carry out the calculations will require as many operations as the number of states; in fact, for the most general marginalization problems there are no better ways known, than this very approach. Fortunately, however, in many cases of interest one can satisfactorily model the underlying system, so that the marginalization problem will have enough 'structure' to be efficiently solvable. *Conditional independence* relations are the best form to express such structures; each such relation will allow the marginalization task to be broken up into smaller, and more manageable, subproblem.

Graphical models are the powerful tools used to efficiently store these conditional independencies and represent the structure in the problem. The graphical represen-

tations of complex probabilistic relationships are not only compact, but also often more humanly comprehensible. In addition, as it turns out, some of the most efficient methods to solve the marginalization problem can be viewed as message-passing algorithms on such graphs.

There have been numerous articles and books discussing different variations of graphical models, from Bayesian networks and chain graphs, to factors graphs and trellises. The common feature of all these models is that conditional independencies are represented by an appropriate notion of separation on the graph. We will not discuss the nuances of different classes of graphical models, and rather refer the reader to the following, or a number of other good references: [33; 10; 25; 17; 41; 21]. For the purposes of this document and in the context of the marginalization problems posed in Section 1.1, by a *graphical representation* we mean an 'independency map' or I-map, as defined in [33]: A (directed or undirected) graph $\mathbf{G}$ with one-to-one correspondence between the elements of R and the nodes of $\mathbf{G}$, such that separation on the graph implies conditional independence. Our concept of graph separation is the conventional one, where two (subsets of) nodes are said to be separated, given a third node if every (directed or undirected) path connecting the two passes through the third node.

We define a conventional junction tree as follows:

Definition 1.1. Let R be a collection of subsets of the index set $[N]$. A tree/forest $\mathbf{G}$ with vertices corresponding to the elements of R is called a *junction* tree/forest on R if for each $i \in [N]$, the subgraph of $\mathbf{G}$ consisting of all the vertices that contain i is connected. An equivalent condition is that, for every triple of regions $r, s, t \in R$ such that node s separates r and t on $\mathbf{G}$, we must have $r \cap t \subseteq s$. □

Although junction trees are traditionally defined as undirected trees, in the above definition we do not make distinction between directed and undirected graphs; we call a directed graph a junction tree if its undirected version is a junction tree in the usual sense.

The junction tree algorithm, which we will discuss in the next section, is an efficient message-passing algorithm that solves an MPF problem exactly. The complexity of that algorithm depends on the sizes of the regions associated with the nodes of the underlying junction tree.

A junction tree may or may not exist on a given a collection R of regions. There is a rather simple way to answer the question of existence of a junction tree on R, and find a junction tree if one exists:
As described in Section 4 of [2], we can define a *local domain graph* G_{LD} for the collection R. This is a weighted complete graph with vertices that correspond one-to-one with the local domains $r \in R$, with the weight of the edge (r, s) given by $w_{r,s} = |r \cap s|$. Then a junction tree exists on R iff $w_{max} = \sum_{r \in R} |r| - N$, where w_{max} is the weight of a maximum weight spanning tree of G_{LD}, and as before N is the total number of variables (see [2]). In case that equality holds, any maximum weight spanning tree of G_{LD} will be a junction tree. Therefore the problem of existence of a junction tree can be solved using a greedy algorithm (such as Prim's algorithm or Kruskal's algorithms, see e.g. [8]) to find a maximum weight spanning tree of G_{LD}.

When a junction tree does not exist, there are systematic ways to expand the regions of R to guarantee existence of a junction tree on the expanded collection. Given the setup used to define the marginalization problems of Section 1.1, we define the *moral graph* for the problem as an undirected graph with vertices $1, \cdots, N$, where an edge (i, j) exists iff $\{i, j\} \subseteq r$ for some $r \in R$. It is known that a junction tree of the cliques of a graph $\mathbf{G}$ exists iff $\mathbf{G}$ is *triangulated* or *chordal*, i.e. every cycle of length 4 or more on $\mathbf{G}$ has a chord (see e.g. [10] Section 4.3). Then the general procedure to construct a junction tree is the following (*moralization and triangulation process*): form the moral graph and triangulate it; then a junction tree will exist on the cliques of the triangulated graph. Note that by construction, each local domain $r \in R$ is contained in some clique of the moral graph, and in this sense a junction tree on these cliques can also be viewed as a junction tree on R. The triangulation

process in effect expands the cliques of the moral graph – by adding variables – in a way to create conditional independencies which are required on a junction tree.

There are efficient algorithms to find a triangulation of a given graph, although the general problem of finding the optimal triangulation for moral graph of a GDL problem is NP-hard (see [10]), where optimality is measured in terms of the complexity of the message-passing algorithm on the junction tree created from the cliques of the triangulated graph.

1.2.1 Junction Tree Algorithm, GDL

In this section we describe the well-known message-passing algorithms on junction trees, which solve the marginalization problems of Section 1.1. A general framework for describing such algorithms was described by Shafer and Shenoy [38]. Aji and McEliece [2] gave an equivalent, and for our purposes, slightly more convenient, framework known as the generalized distributive law (GDL) to describe such algorithms. Throughout this document we will use both names – *GDL* and the *junction tree algorithm*– interchangeably, but will frequently use the GDL notation from [2] to show connections with this work.

Suppose $\mathbf{G}$ is a junction tree on R, with local kernels $\{\alpha_r(\mathbf{x}_r), r \in R\}$. Let $\{E_1, \cdots, E_T\}$ be a message-passing *schedule*, viz. the 'message' along the directed edge (r, s) of the graph is updated at time t iff $(r, s) \in E_t$. The following asynchronous message-passing algorithm (GDL) will solve the MPF Problem 1.1:

Algorithm 1.1 (JT, GDL). *At each time n and for all pairs (r, s) of neighboring nodes in the graph let the 'message' from r to s be a function $\mu^n_{r,s} : \prod_{i \in r \cap s}[q_i] \longrightarrow R$. Initialize all messages to* 1. *At each time $n \in \{1, \cdots, T\}$, if the edge $(r, s) \in E_n$ then update the message from node r to s as follows*

$$\mu^n_{r,s}(\mathbf{x}_{r \cap s}) = \sum_{\mathbf{x}_{r \setminus s}} \alpha_r(\mathbf{x}_r) \prod_{t \in N_r \setminus \{s\}} \mu^{n-1}_{t,r}(\mathbf{x}_{t \cap r}) \tag{1.3}$$

where N_r is the set of neighbors of r in $\mathbf{G}$.

This algorithm will converge in finite time, at which time we have:

$$b_r(\mathbf{x}_r) := \alpha_r(\mathbf{x}_r) \prod_{t \in N_t} \mu_{t,r}(\mathbf{x}_{t \cap r}) = \Pi_r(\mathbf{x}_r) := \sum_{\mathbf{x}_{[N] \setminus r}} \Big(\prod_{t \in R} \alpha_t(\mathbf{x}_t) \Big) \tag{1.4}$$

Proof. See [2]. □

Remark. As noted in [38] and [2], the marginalization problems of Section 1.1, as well as Algorithm 1.1 above, can be generalized on any *semiring*; A (commutative) *semiring* is a set with operations $+$ and $\times$ such that both $+$ and $\times$ are commutative and associative and have identity elements in the set, denoted by 0 and 1 respectively, and $\times$ is distributive over $+$. The obvious semiring with usual addition and multiplication is called *sum-product.* Another very useful semirings is the *max-product* semiring, in which the objective is to find the maximum of the product function over the different configurations of the state vector, e.g. in order to perform maximum *a posteriori* (MAP) estimation.

Algorithm 1.1 above will solve the MPF problem with the given semiring algebra, as long as equations (1.3) and (1.4) are also interpreted in the same semiring. Note that the key property of a semiring, which allows for localization of calculations, is the distributivity of $\times$ over $+$. □

Historically, variations of Algorithm 1.1 are known by different names in different scientific and engineering communities. Most notably, the *belief propagaion* (BP) algorithm of [33] is a version of the above algorithm, designed to solve an MPD problem. Note that in an MPD problem, where the desired marginals are probability distributions, each message-update rule (1.3) can be conveniently scaled by a constant factor, without any loss of information; at termination we simply re-normalize the functions $b_r(\mathbf{x}_r)$, which are now called the *beliefs*, so to obtain valid probability distributions, summing to 1.

1.2.2 Loopy Belief Propagation

Although convergence and correctness of Algorithm 1.1 depends on the graph $\mathbf{G}$ being a junction tree and hence loop-free, note that the BP algorithm can be applied on a graph with loops, as an attempt to approximate solutions of an MPD problem, see e.g. [26] for an empirical study. Existence of loops can create undesirable situations, where the 'old information' from a part of the loop will be regarded as new information; in fact, in the general case, there is no guarantee that an iterative Algorithm 1.1 on a loopy graph will even converge. However, if the loops of the graph are large (say, larger than the number of iterations of the algorithm,) or the 'interdependencies' around the loops are weak in an appropriate sense, then one expects the algorithm to produce reasonable approximations to the desired marginals. Indeed, loopy belief propagation algorithm has been used with enormous success for decoding turbo codes [6] and low-density parity check (LDPC) codes [13; 20]. However, despite a number of excellent partial results which have considerably increased our understanding of the dynamics of such algorithms (see e.g. [37; 46; 36; 11; 35; 20]), so far a general characterization of the quality of approximation and convergence properties of loopy belief propagation has not been discovered.

1.3 Outline of the Rest of This Dissertation

This document is organized in two main parts. Part I introduces a novel, measure-theoretic, junction-tree based approach to solve the MPF problem (1.1) exactly. In Chapter 2 we give motivation for a measure-theoretic approach, and introduce some basic concepts from measure theory, which we will use throughout Part I. We will take careful steps to ensure that concepts such as conditional independence and expectation are well-defined for signed measures. As we will see, the ability to work with signed measures simplifies the task of creating independencies, which is an essential requirement for creating junction trees.

In Chapter 3 we reformulate the marginalization problem at hand in the measure-theoretic framework, and define the appropriate notion of a junction tree. We then give the probabilistic version of the junction tree algorithm in Section 3.1. Sections 3.2 and 3.3 discuss the existence of junction trees and give an algorithm to find one, if one exists.

Chapter 4 contains the core of our contribution in Part I. There we introduce the concept of 'lifting', as a way to create the required conditional independencies on a junction tree. This results in the algorithm of Section 4.2 to create a junction tree. In Section 4.3, in light of the theory that is developed, we compare and contrast the conventional variable-based approaches with our measure-theoretic approach.

In Chapter 5 we discuss the complexity of our methodology. We start with the complexity of computing a single conditional expectation in Section 5.1. Then in Sections 5.2 and 5.3 we discuss the complexity of our probabilistic junction tree algorithm, and the algorithm to create a junction tree.

In Chapter 6 we give several examples of application of our method, in each case comparing the resulting message-passing algorithm with the conventional junction tree solution. Of particular interest is the observation that the minimal complexity trellis-based decoding algorithm for linear block codes can be very naturally described as an instance of lifting in our framework. Finally in Chapter 7 we further discuss and summarize our results.

Part II of this document is devoted to a discussion of the *Kikuchi approximation method*, which is a "free energy minimization"-based approach to obtain exact or approximate solution to an MPD problem (1.2). The junction tree algorithm and the loopy belief propagation discussed in Sections 1.2.1 and 1.2.2 can be obtained from special cases of the Kikuchi approximation method. In Chapter 8 we give an overview of the approximation methods based on free energy minimization, and set up the notation for the rest of Part II.

In Chapter 9 we present a detailed description of the Kikuchi approximation method. Section 9.1 discusses the connection between the marginalization problems of Section 1.1, and free energy concepts from statistical physics. In Section 9.2 we define the Kikuchi approximation method in terms of the solutions of a constrained minimization problem. As we will discuss in Section 9.3, Lagrange multipliers method can be used to define iterative algorithms whose fixed points coincide with the stationary points of this constrained minimization problem. We also discuss conditions for the convexity of this problem in Section 9.4.

We introduce graphical representations for a Kikuchi problem in Chapter 10. For each such problem we specify the *minimal graphical representations*, as those with the fewest edges. From a practical point of view, these minimal graphs result in iterative solutions with the fewest number of messages at each iteration, and are hence the most compact. At the same time, the minimal graphs are closely related to the traditional junction trees, as we will discuss in Section 10.1. In Section 10.2 we report our main theoretical results of Part II, in form of necessary and sufficient conditions for the exactness of the Kikuchi approximation method. In particular, we will show that for generic MPD problems, the Kikuchi approximation method can produce exact solutions if and only if the corresponding minimal graph is loop-free.

In Chapter 11 we derive a version of the generalized belief propagation (GBP) algorithm of [49] on any graphical representation of the Kikuchi problem. The results reported here are a generalization of those in [49], [50] and [23].

In Chapter 12 we report the simulation results of running the GBP algorithm on a few specific choices of Kikuchi problems. In each case we compare the convergence behavior of our compact version of GBP on the minimal graph with those of the conventional GBP of [50] (or the Poset-BP of [23]). Finally in Section 12.1 we give the result of applying the Kikuchi approximation method to the specific problem of joint decoding of a low-density parity-check code and a partial response channel. In Chapter 13 we discuss and summarize the results of Part II.

Part I

Probabilistic Junction Trees

Chapter 2

Introduction and Setup

2.1 Motivation

Local message passing algorithms such as GDL aim to solve the MPF problem 1.1 of Section 1.1, to find the desired set of marginals of a product function. As discussed in Section 1.2.1, GDL makes use of the *distributivity* of the 'product' operation over 'summation' in an underlying semiring to reduce the complexity of the required calculations for a given marginalization problem. In many cases this translates to substantial savings over brute-force computation. However, as we shall see in this document, sometimes there is more structure available in the product function than the conventional way of thinking about GDL can discover. This is because GDL relies solely on the notion of variables; any structure at a finer level than that of variables will be ignored by GDL. We illustrate these limitations in the following simple example:

Example 2.1. Let X and Y be arbitrary real functions on $\{1, \cdots, n\}$. Let $\mu(i,j)$ be a fixed real weight function for $i,j \in \{1, \cdots, n\}$, given by an $n \times n$ matrix M with $\mu(i,j) = M_{i,j}$. We would like to calculate the weighted average of $X \cdot Y$: $E = \sum_{i=1}^{n} \sum_{j=1}^{n} X(i)Y(j)\mu(i,j)$.

The general GDL-type algorithm (assuming no structure on the weight function μ) will suggest the following:

$$E = \sum_{i=1}^{n} X(i) \sum_{j=1}^{n} Y(j)\mu(i,j),$$

requiring $n(n+1)$ multiplications and $(n-1)(n+1)$ additions. But this is not always the simplest way to calculate E.

Consider a 'luckiest' case when the matrix M has rank 1, i.e. the weight function $\mu(i,j)$ factors as $f_1(i) \cdot f_2(j)$. In this case $E = \big(\sum_{i=1}^n X(i)f_1(i)\big)\big(\sum_{j=1}^n Y(j)f_2(j)\big)$, requiring only $2n+1$ multiplications and $2n-2$ additions.

Suppose next that $\mu(i,j)$ does not factor as above, but the matrix M has a low rank of 2, so that $\mu(i,j) = f_1(i)f_2(j) + g_1(i)g_2(j)$. Then we can compute E as follows:

$$E = \big(\sum_{i=1}^n X(i)f_1(i)\big)\big(\sum_{j=1}^n Y(j)f_2(j)\big) + \big(\sum_{i=1}^n X(i)g_1(i)\big)\big(\sum_{j=1}^n Y(j)g_2(j)\big)$$

This requires $4n+2$ multiplications and $4n-4$ additions.

Next suppose that the fixed weight matrix M is sparse, for example with $\mu(i,j) = m_i\, 1(i+j-n-1=0)$, where $1(\cdot)$ is the indicator function. Then

$$E = \sum_{i=1}^n m_i X(i)Y(n+1-i),$$

requiring only $2n$ multiplications and $n-1$ additions.

It would be nice to have an automatic procedure to discover the existence of such complexity reducing opportunities. The goal of the first part of this dissertation is to develop such a procedure. □

Note that in each case in Example 2.1, some (manual) introduction of hidden variables and/or redefinition of functions would allow the conventional GDL to also discover the best method of calculation. However we do not consider this preprocessing phase as part of the GDL treatment. Our aim is to develop an *automatic* procedure that aims to discover such potentially useful structures in the data.

We introduce a measure-theoretic framework which goes beyond the focus on 'variables' to represent the states of the data. With the conventional GDL approach, once one chooses the set of variables to represent the state-space, the consequent attempts to create independencies in order to find a junction tree (i.e. the moralization and triangulation procedure) are confined to working with that chosen set of variables.

The primary advantage of our reformulation is to get rid of this restriction. The alternative we provide to the moralization and triangulation procedure, which we call 'lifting,' *automatically* discovers a way to exploit structure that is not aligned to the variables, which the usual approach is unable to discover. For instance, in the example above, we can automatically discover the advantage of the low rank of matrix M.

Our measure-theoretic framework replaces GDL's concept of 'local domains' with σ-fields in an appropriate sample-space. We also replace GDL's 'local kernels' with random variables measurable with respect to the corresponding σ-fields. The problem of finding the marginal with respect to a local domain is then naturally replaced by that of taking the conditional expectation given the corresponding σ-field. As we shall see, this representation has the flexibility to capture both full and partial independencies in the state space.

Our formalism includes the conventional GDL as a special case, in the sense that any junction tree that moralization and triangulation can produce in the conventional GDL can also be discovered using our framework.

Remember that GDL algorithm of Section 1.2.1 generalizes to arbitrary semirings. Similarly, our results are generalizable to an arbitrary *semifield*[1]. However in the rest of this document we focus on the sum-product algebra in order to avoid abstract distractions.

2.2 Preliminaries

Let $(\Omega, \mathcal{M})$ be a *discrete measurable space*, i.e. Ω is a finite or countable set and $\mathcal{M}$ is a σ-field on Ω. Let $\mu : \mathcal{M} \to (-\infty, \infty)$ be a *signed measure* on $(\Omega, \mathcal{M})$, i.e. $\mu(\emptyset) = 0$ and for any sequence $\{a_i\}_{i=1}^{\infty}$ of disjoint sets in $\mathcal{M}$, $\mu(\bigcup_1^\infty a_i) = \sum_1^\infty \mu(a_i)$, where the

[1] A semifield is an algebraic structure with addition and multiplication, both of which are commutative and associative and have identity element. Further, multiplication is distributive over addition, and every nonzero element has a multiplicative inverse. Such useful algebras as the sum-product and the max-sum are examples of semifields (see [4]).

infinite sum is implicitly assumed to exist. Then $(\Omega, \mathcal{M}, \mu)$ is called a *measure space.* As a matter of notation, we usually write $\mu(a_1, a_2, \cdots, a_n)$ for $\mu(a_1 \cap a_2 \cap \cdots \cap a_n)$. Also, given events a and b with $\mu(b) \neq 0$, we write $\mu(a|b)$ for the ratio $\frac{\mu(a,b)}{\mu(b)}$.
Let $\mathcal{F}, \mathcal{G}, \mathcal{H}$ and $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$ be sub σ-fields of $\mathcal{M}$.

Definition 2.1.

- *Atoms of a σ-field:* We define the set of *atoms* of $\mathcal{F}$ to be the collection of the minimal nonempty measurable sets in $\mathcal{F}$ w.r.t. inclusion:

$$\mathcal{A}(\mathcal{F}) := \{f \in \mathcal{F} \ : \ f \neq \emptyset, \text{ and } \forall g \in \mathcal{F}, \ f \cap g \in \{\emptyset, f\}\}.$$

  We also denote by $\mathcal{A}'(\mathcal{F})$ the set of atoms of $\mathcal{F}$ with nonzero measure:

$$\mathcal{A}'(\mathcal{F}) := \{f \in \mathcal{A}(\mathcal{F}) \ : \ \mu(f) \neq 0\}.$$

- *Augmentation of σ-fields:* We denote by $\mathcal{F} \vee \mathcal{G}$ the *span* of $\mathcal{F}$ and $\mathcal{G}$, i.e. the smallest σ-field containing both $\mathcal{F}$ and $\mathcal{G}$. For a set A of indices, we write $\mathcal{F}_A$ for $\bigvee_{i \in A} \mathcal{F}_i$, with $\mathcal{F}_\emptyset := \{\emptyset, \Omega\}$, the trivial σ-field on Ω. Note that the atoms of $\mathcal{F} \vee \mathcal{G}$ are all in the form $f \cap g$ for some $f \in \mathcal{A}(\mathcal{F})$ and $g \in \mathcal{A}(\mathcal{G})$.

- *Conditional Independence:* We say $\mathcal{F}$ is *conditionally independent of $\mathcal{G}$ given $\mathcal{H}$* and write $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \mid \mathcal{H}$ w.r.t. μ when for every atom h of $\mathcal{H}$,

  - if $\mu(h) = 0$ then $\forall f \in \mathcal{F}, g \in \mathcal{G}, \quad \mu(f, g, h) = 0$
  - if $\mu(h) \neq 0$ then $\forall f \in \mathcal{F}, g \in \mathcal{G}, \quad \mu(f, g, h)\mu(h) = \mu(f, h)\mu(g, h)$.

  When the underlying measure is obvious from the context, we omit the explicit mention of it. Similarly, we say a collection $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$ of sub σ-fields are *mutually conditionally independent given $\mathcal{H}$*, and write $\mathcal{F}_1 \perp\!\!\!\perp \mathcal{F}_2 \perp\!\!\!\perp \cdots \perp\!\!\!\perp \mathcal{F}_M | \mathcal{H}$, if $\mathcal{F}_i \perp\!\!\!\perp \bigvee_{j \neq i} \mathcal{F}_j \mid \mathcal{H}$ for all $i \in \{1, \cdots, M\}$.

- *Independence:* We say $\mathcal{F}$ is independent of $\mathcal{G}$ or $\mathcal{F} \perp\!\!\!\perp \mathcal{G}$ w.r.t. μ when $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \mid \mathcal{F}_\emptyset$.

Note that these definitions are consistent with the usual definitions of independence when μ is a probability measure.

- *Partially-defined Random Variables:* A *partially-defined* random variable X in $\mathcal{F}$ is a partially-defined function on Ω, where for each r in the range of X, $X^{-1}(r)$ is measurable in $\mathcal{F}$. We write $X \in \mathcal{F}$, and denote by $\mathcal{A}_X(\mathcal{F})$ the subset of $\mathcal{A}(\mathcal{F})$ where X is defined.

- *Expectation and Conditional Expectation:* Assuming $\mu(\Omega) \neq 0$ and that the sum exists, we define the *expectation* of X as

$$\begin{aligned} \mathbf{E}[X] &:= \frac{1}{\mu(\Omega)} \sum_{f \in \mathcal{A}_X(\mathcal{F})} X(f)\mu(f) \\ &= \sum_{f \in \mathcal{A}_X(\mathcal{F})} X(f)\mu(f|\Omega) \end{aligned} \tag{2.1}$$

When the sum exists, we define the *conditional expectation* of X given $\mathcal{G}$, as a partially-defined random variable Y in $\mathcal{G}$, defined on $\mathcal{A}_Y(\mathcal{G}) := \mathcal{A}'(\mathcal{G})$, as:

$$\begin{aligned} \mathbf{E}[X|\mathcal{G}](g) &:= \frac{1}{\mu(g)} \sum_{f \in \mathcal{A}_X(\mathcal{F})} X(f)\mu(g,f) \quad \text{for each atom } g \in \mathcal{A}'(\mathcal{G}) \\ &= \sum_{f \in \mathcal{A}_X(\mathcal{F})} X(f)\mu(f|g) \qquad\qquad \text{for } g \in \mathcal{A}'(\mathcal{G}) \end{aligned} \tag{2.2}$$

□

It is important to note that equations (2.1) and (2.2) should not be taken at face value, as prescribing the way to carry out the summation; in many cases, such as the ones in Example 2.1, the calculation can be simplified. We will address this in detail in Section 5.1 in the context of our general method to create independencies.

Remark. It should be kept in mind that the definitions above are tailored to work with a *signed* measure μ. Indeed some of the specifications above would be unnecessary if the measure μ were unsigned; e.g. in the definition of conditional independence

above, the first condition is automatically true for an unsigned measure, because in that case $\mu(f,g,h) \leq \mu(h)$ for all f,g,h. Similarly, the introduction of the partially-defined random variables is only necessary in order to deal with such anomalous situations with signed measures, where a zero-measured event can have *non*-zero-measured subsets, i.e. where we have $\mu(f \cup g) = 0$ while $\mu(f) \neq 0$ and $\mu(g) \neq 0$ for some $f,g \in \mathcal{M}$. As we shall see, however, the derivation of the forthcoming theory with *signed* measures significantly simplifies the essential task of creating conditional independencies. □

The following simple example illustrates some of the basic concepts above:

Example 2.2. Consider a simple sample space $\Omega = \{0,\cdots,9\}$.

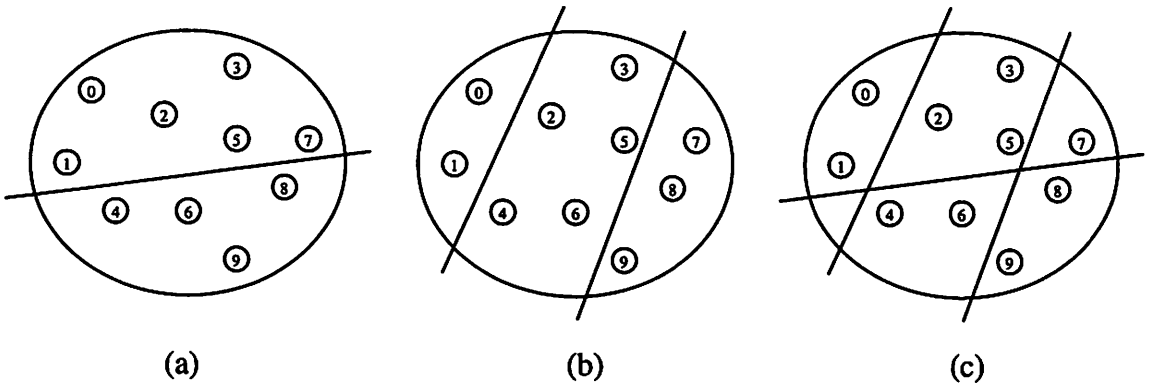


Figure 2.1: Pictorial illustration of Example 2.2
(a) Atoms of $\mathcal{F}_a$, (b) atoms of $\mathcal{F}_b$, and (c) atoms of $\mathcal{F}_a \vee \mathcal{F}_b$.

Each σ-field is defined by its set of atoms, as a partition of Ω. Figures 2.1(a) and (b) illustrate two σ-fields $\mathcal{F}_a$ and $\mathcal{F}_b$, where $\mathcal{A}(\mathcal{F}_a) = \{\{0,1,2,3,5,7\},\{4,6,8,9\}\}$ and $\mathcal{A}(\mathcal{F}_b) = \{\{0,1\},\{2,3,4,5,6\},\{7,8,9\}\}$. Figure 2.1(c) illustrates the augmented σ-field $\mathcal{F}_a \vee \mathcal{F}_b$, where $\mathcal{A}(\mathcal{F}_a \vee \mathcal{F}_b) = \{\{0,1\},\{2,3,5\},\{4,6\},\{7\},\{8,9\}\}$.

If X is a random variable in $\mathcal{F}_a$, then X would be constant over each atom of $\mathcal{F}_a$, i.e. $X(0) = X(1) = X(2) = X(3) = X(5) = X(7)$, and $X(4) = X(6) = X(8) = X(9)$. On the other hand, Y could be a partially-defined random variable in $\mathcal{F}_b$, defined on $\mathcal{A}_Y(\mathcal{F}_b) := \{\{0,1\},\{7,8,9\}\}$. Then we would have $Y(0) = Y(1)$, and $Y(7) = Y(8) = Y(9)$, while $Y(2),\cdots,Y(6)$ are undefined.

Both $X \in \mathcal{F}_a$ and $Y \in \mathcal{F}_b$ can be viewed as (partially-defined) random variables in $\mathcal{F}_a \vee \mathcal{F}_b$, since each atom of $\mathcal{F}_a \vee \mathcal{F}_b$ is contained in an atom of $\mathcal{F}_a$ and $\mathcal{F}_b$. □

The signed conditional independence relation satisfies certain properties (inference rules) that we now state. See [33] and [10] for discussion of inference rules for the case when μ is a probability measure[2].

Theorem 2.1. *Let $\mathcal{F}, \mathcal{G}, \mathcal{X}, \mathcal{Y}$ be σ-fields. Then the following properties hold:*

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{X} \implies \mathcal{G} \perp\!\!\!\perp \mathcal{F} \,|\, \mathcal{X} \qquad \textit{Symmetry} \qquad (2.3a)$$

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{X} \,|\, \mathcal{Y} \implies \mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{Y} \;\;\&\;\; \mathcal{F} \perp\!\!\!\perp \mathcal{X} \,|\, \mathcal{Y} \qquad \textit{Decomposition} \qquad (2.3b)$$

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{X} \;\&\; \mathcal{F} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{G} \vee \mathcal{X} \implies \mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{Y} \,|\, \mathcal{X} \qquad \textit{Contraction} \qquad (2.3c)$$

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{X} \;\&\; \mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{X} \implies \mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{Y} \,|\, \mathcal{X} \qquad (2.3d)$$

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{X} \;\&\; \mathcal{F} \vee \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{G} \implies \mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{Y} \,|\, \mathcal{X} \qquad (2.3e)$$

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{X} \,|\, \mathcal{Y} \;\&\; \mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{X} \implies \mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{Y} \,|\, \mathcal{X} \;\&\; \mathcal{F} \vee \mathcal{Y} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{X} \qquad (2.3f)$$

Further, if measure μ is known to be unsigned, the following also holds:

$$\mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{X} \,|\, \mathcal{Y} \implies \mathcal{F} \perp\!\!\!\perp \mathcal{G} \,|\, \mathcal{X} \vee \mathcal{Y} \qquad \textit{Weak Union} \qquad (2.3g)$$

Proof. See Appendix A.1 □

[2]Note that the signed conditional independence relation satisfies *symmetry, decomposition* and *contraction*, but in general *weak union* does not hold. So this relation is not a *semi-graphoid*, see [33]. The unsigned conditional independence, on the other hand is a semi-graphoid.

Chapter 3

Probabilistic Junction Trees

We now formulate a probabilistic version of the MPF problem 1.1 and introduce the corresponding concept of junction trees. We also describe a probabilistic version of the GDL algorithm to solve this marginalization problem.

Throughout this document let $(\Omega, \mathcal{M}, \mu)$ be a measure space, $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$ be sub σ-fields of $\mathcal{M}$, and let $\{X_1, \cdots, X_M\}$ be a collection of partially defined random variables with $X_i \in \mathcal{F}_i$. We will speak of the measure space $(\Omega, \{\mathcal{F}_1, \cdots, \mathcal{F}_M\}, \mu)$ since the choice of $\mathcal{M}$ will not be relevant. We are interested in solving the following:

Problem 3.1 (Probabilistic MPF). *For one or more $i \in \{1, \cdots, M\}$, find $Y_i := \mathbf{E}[\prod_j X_j | \mathcal{F}_i]$, the conditional expectation of the product, given $\mathcal{F}_i$.* □

Given a conventional MPF problem, one can choose a subset of local kernels whose product is viewed as a measure function in our framework, and the other kernels can be viewed as random variables, each measurable w.r.t. the σ-field defined by the variables comprising its local domain. Suppose that $R = \{r_1, \cdots, r_n\}$. Then for a configuration x_{r_i} of the variables corresponding to a local domain $r_i \in R$, we have

$$\mathbf{E}[\prod_{j=1}^{M} X_j | \mathcal{F}_i](\mathbf{x}_{r_i}) = \frac{1}{\mu(\mathbf{x}_{r_i})} \sum_{\mathbf{x}_{r_i^c}} \Big(\prod_{j=1}^{n} \alpha_{r_j}(\mathbf{x}_{r_j}) \Big) \propto \Pi_{r_i}(\mathbf{x}_{r_i}) \tag{3.1}$$

where, out of the n original local kernels, the first M were treated as random variables, and the last $n - M$ have been relegated to the measure, so that $\mu(\mathbf{x}) =$

$\prod_{k=M+1}^{n} \alpha_{r_k}(\mathbf{x}_{r_k})$. Equation (3.1) shows that solving the probabilistic MPF problem with this setup will in effect amount to a solution to the conventional MPF problem.

In most applications, for a family of MPF problems the local kernels can be categorized as either *fixed* or *arbitrary*. For example, in an LDPC decoding problem, which we will consider in Example 6.6, the code itself is fixed, so the local kernels at the check-nodes are fixed; we only receive new observations and try to find the most likely codeword given each observation set. As another example, when finding the Hadamard transform $\sum_{x_1,\cdots,x_n} \prod_{i=1}^{n} (-1)^{x_i y_i} f(x_1, \cdots, x_n)$ of an arbitrary function f, the functions $(-1)^{x_i y_i}$ are fixed, see Example 6.4. Typically, we want to assign (some of) the fixed kernels as the measure function, and the arbitrary kernels as the marginalizable random variables; this way, once a junction tree has been found for one problem, it can be used to marginalize the product of any arbitrary collection of random variables measurable in the same σ-fields. See Chapter 6 for more examples.

We now define (probabilistic) junction trees as follows:

Definition 3.1. Let $\mathbf{G}$ be a tree with nodes $\{1, \cdots, M\}$. We say subsets A and B of $\{1, \cdots, M\}$ are *separated* by a node i if $\forall x \in A, y \in B$, the path from x to y contains i. Then we call $\mathbf{G}$ a *junction tree* if $\forall A, B \subset \{1, \cdots, M\}$ and $i \in \{1, \cdots, M\}$ s.t. i separates A and B on the tree, we have $\mathcal{F}_A \perp\!\!\!\perp \mathcal{F}_B \mid \mathcal{F}_i$. □

The definition above should be compared to that of the conventional junction tree in Section 1.2. As was the case there, junction trees have the key property that *separation on the tree imply conditional independence.* This is the very property that allows the marginalization task to be performed locally, and hence, with reduced computational complexity.

3.1 Probabilistic Junction Tree (PJT) Algorithm

Suppose $\mathbf{G}$ is a junction tree with nodes labelled by σ-fields $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$ as defined above, and let $\{X_1, \cdots, X_M\}$ be random variables with $X_i \in \mathcal{F}_i$. Then the following message-passing algorithm will solve the probabilistic MPF problem 3.1:

Algorithm 3.1 (PJT). *For each edge (i,j) on the graph, define a 'message' $Y_{i,j}$ from node i to j as a partially-defined random variable measurable w.r.t. $\mathcal{F}_j$, initialized to* 1.

For each $i = 1, \cdots, M$ let N_i denote the set of neighbors of i on $\mathbf{G}$, and for each edge (i,j) in the tree, define $N_{i,j} = N_i \backslash \{j\}$.

For each edge (i,j) in the tree, update the message $Y_{i,j}$ (asynchronously) as:

$$Y_{i,j} = \mathbf{E}\Big[X_i \prod_{k \in N_{i,j}} Y_{k,i} \Big| \mathcal{F}_j\Big] \tag{3.2}$$

This algorithm will converge in finite time, at which time we have:

$$Y_i = X_i \prod_{k \in N_i} Y_{k,i} \tag{3.3}$$

where as before, $Y_i := \mathbf{E}\Big[\prod_{j=1}^{M} X_j \Big| \mathcal{F}_i\Big]$ is the objective random variable.

Proof of correctness of Algorithm 3.1. See Appendix A.2 □

It is easy to notice the similarities between the PJT algorithm above, and the conventional junction tree (GDL) Algorithm 1.1 of Section 1.2.1: the outgoing message from a node of the junction tree is computed by local marginalization – i.e. taking the conditional expectation – of the product of the corresponding local random variable, and the incoming messages from all other neighbors of that node. To emphasize this connection further, note that message $Y_{i,j}$ in the algorithm above can be viewed as a function $Y_{i,j}(f_j)$ on $\mathcal{A}'(\mathcal{F}_j)$, where f_j ranges over $\mathcal{A}'(\mathcal{F}_j)$. Then the update rule (3.2) can be rewritten as (c.f. equation (1.3)):

$$Y_{i,j}(f_j) = \sum_{f_i \in \mathcal{A}'(\mathcal{F}_i)} \mu(f_i|f_j) X_i(f_i) \prod_{k \in N_{i,j}} Y_{k,i}(f_i) \tag{3.4}$$

Similarly, equation (3.3) can be rewritten as (c.f. equation (1.4)):

$$Y_i(f_i) = X_i(f_i) \prod_{k \in N_i} Y_{k,i}(f_i) \tag{3.5}$$

One should note that rarely is the way suggested by equation (3.4) an efficient way to carry out the summation (we will discuss this further in Section 5.1). However, there may be some value to seeing equations (3.2) and (3.3) written as (3.4) and (3.5), since in some sense the language of σ-fields has been translated into one of 'variables.' However it should be remembered that these 'variables' $f_j \in \mathcal{A}'(\mathcal{F}_j)$ for $j = 1, \cdots, M$ do not in any sense play the role of what are called variables in the original GDL algorithm. We will discuss the computational complexity of Algorithm 3.1 in Section 5.2.

3.2 Existence of Junction Trees

As described in Section 1.2, in the conventional GDL formulation, given the collection R of the local domains, there is a rather simple way to determine whether a junction tree exists, requiring only a greedy algorithm to find a maximum-weight spanning tree of the so-called local domain graph G_{LD}.

In our measure-theoretic framework on the other hand, this problem is not as simple. The reason is that in the GDL framework the conditional independence property is much easier to verify: given three local domains $r, s, t \in R$, r and s are conditionally independent given t iff $r \cap s \subseteq t$. In our framework, on the other hand, in general the σ-fields are not rectangular, and hence cannot be represented in terms of underlying variables; indeed elimination of the variables was the key objective in switching to the probabilistic framework. This lack of compact representation makes it harder to determine if conditional independencies exist.

In this section we present an algorithm to determine whether a junction tree exists on the given σ-fields, and to find one if it does exist. We will prove results that will allow us, in a divide-and-conquer way, to break down the problem of finding a junction

tree into smaller problems and recursively build a junction tree from smaller trees.

Definition 3.2. A *valid partition* of $\{1,\cdots,M\}\backslash\{i\}$ with respect to a node i is a partition $\{p_1,\cdots,p_l\}$ of $\{1,\cdots,M\}\backslash\{i\}$ (i.e. $\bigcup_{j=1}^{l} p_j = \{1,\cdots,M\}\backslash\{i\}$ and $p_j \cap p_k = \emptyset$ for $j \neq k$) such that $\mathcal{F}_{p_j}$'s are mutually conditionally independent, given $\mathcal{F}_i$. □

Definition 3.3. Let $P = \{p_1,\cdots,p_l\}$ be any partition of $\{1,\cdots,M\}\backslash\{i\}$. A tree with nodes $\{1,\cdots,M\}$ is called *compatible* with partition P at node i if its subtrees hanging from i correspond to the elements of P. □

It is clear that the conditional independence relations satisfied by the elements of a valid partition are precisely of the kind that is implied by the junction tree definition. The following results make this connection precise:

Lemma 3.1. *Given $\{\mathcal{F}_1,\cdots,\mathcal{F}_M\}$, a tree with nodes $\{1,\cdots,M\}$ is a junction tree iff at each node i it is compatible with some valid partition of $\{1,\cdots,M\}\backslash\{i\}$ w.r.t. i.*

Proof. Immediate from the definitions. □

Lemma 3.2. *$\forall i \in \{1,\cdots,M\}$, there is a* finest valid partition *w.r.t. i, which we shall denote by P_i, such that every other valid partition w.r.t. i is a coarsening of P_i. Further, if p is an element of P_i and p is the disjoint union of nonempty sets e_1 and e_2, then $\mathcal{F}_{e_1} \not\perp\!\!\!\perp \mathcal{F}_{e_2} \mid \mathcal{F}_i$.*

Proof. Suppose $A = \{p_1,\cdots,p_l\}$ and $B = \{q_1,\cdots,q_m\}$ are valid partitions w.r.t. node i. Now construct another partition, $S = \{p \cap q : p \in A \;\&\; q \in B\}$. We claim that S is also a valid partition w.r.t. i, (finer than both A and B): To see this, we need to show that for each $d = p \cap q \in S$, $\mathcal{F}_d \perp\!\!\!\perp \mathcal{F}_{d^c} \mid \mathcal{F}_i$, where $d^c := \{1,\cdots,M\}\backslash\{i\}\backslash d$ is the complement of d. Using simple manipulations like $\mathcal{F}_p = \mathcal{F}_{p\cap(q\cup q^c)} = \mathcal{F}_{p\cap q} \vee \mathcal{F}_{p\cap q^c}$ we get:

$$\mathcal{F}_{p\cap q} \vee \mathcal{F}_{p\cap q^c} \perp\!\!\!\perp \mathcal{F}_{p^c} \mid \mathcal{F}_i$$

$$\mathcal{F}_{p\cap q} \vee \mathcal{F}_{p^c\cap q} \perp\!\!\!\perp \mathcal{F}_{p\cap q^c} \vee \mathcal{F}_{p^c\cap q^c} \mid \mathcal{F}_i \Rightarrow \mathcal{F}_{p\cap q} \perp\!\!\!\perp \mathcal{F}_{p\cap q^c} \mid \mathcal{F}_i \qquad \text{by (2.3b)}$$

And finally, the last two relations and (2.3d) imply that $\mathcal{F}_{p\cap q} \perp\!\!\!\perp \mathcal{F}_{p^c\cup(p\cap q^c)} \mid \mathcal{F}_i$, and hence $\mathcal{F}_{p\cap q} \perp\!\!\!\perp \mathcal{F}_{(p\cap q)^c} \mid \mathcal{F}_i$. So a finest valid partition w.r.t. i exists, whose atoms are the intersections of atoms of all the valid partitions w.r.t. i.

Now suppose p is an element of P_i and p is the disjoint union of nonempty sets e_1 and e_2, and $\mathcal{F}_{e_1} \perp\!\!\!\perp \mathcal{F}_{e_2} \mid \mathcal{F}_i$. We also have $\mathcal{F}_{e_1} \vee \mathcal{F}_{e_2} \perp\!\!\!\perp \mathcal{F}_{p^c} \mid \mathcal{F}_i$. Then from the last two relations and by (2.3d) we get $\mathcal{F}_{e_1} \perp\!\!\!\perp \mathcal{F}_{p^c} \vee \mathcal{F}_{e_2} \mid \mathcal{F}_i$, and hence $\mathcal{F}_{e_1} \perp\!\!\!\perp \mathcal{F}_{e_1^c} \mid \mathcal{F}_i$. Then e_1 and e_2 would be elements in a finer valid partition which is a contradiction. □

The following lemma states that, given some 'boundary' conditions, junction trees on subsets of $\{1, \cdots, M\}$ can be joined to form a junction tree on the complete set:

Lemma 3.3. *Let d be a subset of $\{1, \cdots, M\}$ and let $d' := \{1, \cdots, M\} \backslash d$ be its complement. Suppose there exist $t \in d$ and $i \in d'$ such that $\mathcal{F}_d \perp\!\!\!\perp \mathcal{F}_{d'} \mid \mathcal{F}_t$ and $\mathcal{F}_d \perp\!\!\!\perp \mathcal{F}_{d'} \mid \mathcal{F}_i$. Let $\mathbf{G}$ be any junction tree on d and $\mathbf{G}'$ any junction tree on d'. Then the tree obtained by connecting $\mathbf{G}$ and $\mathbf{G}'$ by adding an edge between t and i is a junction tree on $\{1, \cdots, M\}$ (see Figure 3.1.)*

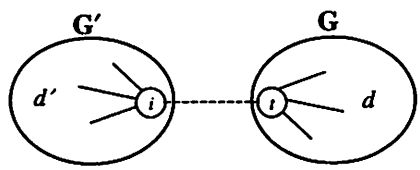


Figure 3.1: Augmenting junction trees; Lemma 3.3

Proof. Let x be any node that separates A and B on the resultant tree. We will show that $\mathcal{F}_A \perp\!\!\!\perp \mathcal{F}_B \mid \mathcal{F}_x$ and hence we have a junction tree.
Let $A_1 = A \cap d, A_2 = A \cap d', B_1 = B \cap d$ and $B_2 = B \cap d'$ and without loss of generality (WLOG) suppose $x \in d$. Then at least one of A_2 and B_2 must be empty, or else x would not separate A and B. Suppose $A_2 = \emptyset$.

First suppose $x = t$. Then we have:

$$\mathcal{F}_{A_1} \perp\!\!\!\perp \mathcal{F}_{B_1} \mid \mathcal{F}_t \qquad \text{by j.t. property on } \mathbf{G}$$

$$\mathcal{F}_{A_1} \vee \mathcal{F}_{B_1} \perp\!\!\!\perp \mathcal{F}_{B_2} \mid \mathcal{F}_t \qquad \text{since } A_1 \cup B_1 \subset d \text{ and } B_2 \subset d'$$

So by (2.3d) we have $\mathcal{F}_{A_1} \perp\!\!\!\perp \mathcal{F}_{B_1} \vee \mathcal{F}_{B_2} \mid \mathcal{F}_t$, i.e. $\mathcal{F}_A \perp\!\!\!\perp \mathcal{F}_B \mid \mathcal{F}_t$ and we are done.
Next suppose $x \in d \backslash \{t\}$. Then we must also have that x separates A_1 from $B_1 \cup \{t\}$ (assuming WLOG that B_2 is nonempty.) Then:

$$\mathcal{F}_{A_1} \perp\!\!\!\perp \mathcal{F}_{B_1} \vee \mathcal{F}_t \mid \mathcal{F}_x \tag{3.6}$$

$$\mathcal{F}_{A_1} \vee \mathcal{F}_x \vee \mathcal{F}_{B_1} \perp\!\!\!\perp \mathcal{F}_{B_2} \mid \mathcal{F}_t \qquad \text{since } A_1 \cup B_1 \cup \{x\} \subset d \text{ and } B_2 \subset d' \tag{3.7}$$

We will show that $\mathcal{F}_{A_1} \perp\!\!\!\perp \mathcal{F}_{B_1} \vee \mathcal{F}_{B_2} \vee \mathcal{F}_t \mid \mathcal{F}_x$ and hence $\mathcal{F}_A \perp\!\!\!\perp \mathcal{F}_B \mid \mathcal{F}_x$.
Let $\chi, \tau, \alpha, \beta_1$ and β_2 be arbitrary atoms of $\mathcal{F}_x, \mathcal{F}_t, \mathcal{F}_A, \mathcal{F}_{B_1}$ and $\mathcal{F}_{B_2}$ respectively.

- *Case* $\mu(\chi) = \mu(\tau) = 0$. Then from (3.7) we have that $\mu(\alpha, \beta_1, \beta_2, \chi, \tau) = 0$, and so we are done.

- *Case* $\mu(\chi) = 0$ *and* $\mu(\tau) \neq 0$. Then from (3.7) we have $\mu(\alpha, \beta_1, \beta_2, \chi, \tau) = \mu(\alpha, \beta_1, \chi, \tau)\mu(\beta_2, \tau)/\mu(\tau)$. But from (3.6), $\mu(\alpha, \beta_1, \chi, \tau) = 0$ since $\mu(\chi) = 0$. Thus $\mu(\alpha, \beta_1, \beta_2, \chi, \tau) = 0$ and we are done.

- *Case* $\mu(\chi) \neq 0$ *and* $\mu(\tau) = 0$. Then from (3.7) we have that $\mu(\alpha, \beta_1, \beta_2, \chi, \tau) = \mu(\beta_1, \beta_2, \chi, \tau) = 0$, and so we have the equality $\mu(\alpha, \beta_1, \beta_2, \chi, \tau)\mu(\chi) = \mu(\alpha, \chi)\mu(\beta_1, \beta_2, \chi, \tau) = 0$ and we are done.

- *Case* $\mu(\chi) \neq 0$ *and* $\mu(\tau) \neq 0$. Then from (3.7),

$$\mu(\alpha, \beta_1, \beta_2, \chi, \tau) = \mu(\alpha, \beta_1, \chi, \tau)\mu(\beta_2, \tau)/\mu(\tau),$$

and from (3.6), $\mu(\alpha, \beta_1, \chi, \tau) = \mu(\alpha, \chi)\mu(\beta_1, \chi, \tau)/\mu(\chi)$. Substituting the latter into the former, we obtain

$$\mu(\alpha, \beta_1, \beta_2, \chi, \tau) = \mu(\alpha, \chi)\mu(\beta_1, \chi, \tau)\mu(\beta_2, \tau)/(\mu(\chi)\mu(\tau)).$$

But by (3.7), $\mu(\beta_1, \chi, \tau)\mu(\beta_2, \tau)/\mu(\tau) = \mu(\beta_1, \beta_2, \chi, \tau)$, so

$$\mu(\alpha, \beta_1, \beta_2, \chi, \tau) = \mu(\alpha, \chi)\mu(\beta_1, \beta_2, \chi, \tau)/\mu(\chi)$$

and we are done.

□

We now state our main theorem on the existence of junction trees:

Theorem 3.4. *Given a set of σ-fields $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$, if there exists a junction tree on $\{1, \cdots, M\}$, then for every $i \in \{1, \cdots, M\}$ there exists a junction tree compatible with P_i, the finest valid partition w.r.t. i.*

Proof. The claim is trivial for $M \leq 3$. We will prove the theorem for $M > 3$ by induction: Let $P_i = \{c_1, \cdots, c_l\}$ with $\bigcup_{j=1}^{l} c_j = \{1, \cdots, M\}\backslash\{i\}$ and $c_j \cap c_k = \emptyset$ for $j \neq k$. Let $\mathbf{G}$ be a junction tree. Let $Q = \{d_1, \cdots, d_n\}$ be the partition of $\{1, \cdots, M\}\backslash\{i\}$ compatible with $\mathbf{G}$. Let $d = d_j$ be an arbitrary element of Q, and let $d' = \bigcup_{k \neq j} d_k \cup \{i\}$. Let $t = N_i \cup d$ be the node in d that is neighbor to i in tree $\mathbf{G}$. By Lemmas 3.2 and 3.1 above, d is the union of some of c_k's. WLOG assume that $d = \bigcup_{k=1}^{K} c_k$ where $K \leq l$, and also assume that $t \in c_K$.

Then from the junction tree property, we have

$$\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_d \mid \mathcal{F}_t \tag{3.8}$$

Since $\mathbf{G}$ is a junction tree, the subtree on d is also a junction tree. Now $|d| < M$, and so by induction hypothesis there exists a junction tree on d compatible with P'_t, the finest valid partition w.r.t. t of $d\backslash\{t\}$.

Now we claim that $R = \{c_k \backslash \{t\} \ : \ 1 \leq k \leq K\}$ is a valid partition of $d\backslash\{t\}$ w.r.t. t. To see this, let $c = c_k$ for some arbitrary $k = 1, \cdots, K$, and let $c' = d\backslash\{c\}$, so $\mathcal{F}_d = \mathcal{F}_c \vee \mathcal{F}_{c'}$. But one of c and c' contains t. Then by the properties of valid partition w.r.t. i, we have:

$$\mathcal{F}_c \vee \mathcal{F}_t \perp\!\!\!\perp \mathcal{F}_{c'} \mid \mathcal{F}_i \qquad \text{or} \qquad \mathcal{F}_c \perp\!\!\!\perp \mathcal{F}_{c'} \vee \mathcal{F}_t \mid \mathcal{F}_i$$

$$\text{also,} \qquad \mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_c \vee \mathcal{F}_{c'} \mid \mathcal{F}_t \quad \text{since } t \text{ separates } i \text{ from } d \text{ on } \mathbf{G}$$

Then by (2.3f) followed by (2.3b), the last relations imply that $\mathcal{F}_c \perp\!\!\!\perp \mathcal{F}_{c'} \mid \mathcal{F}_t$ and we are done.

Next we show that for all $k \in \{1, \cdots, K-1\}$ (so that $t \notin c_k$), c_k is an element of P'_t. If not, then there exists a $c = c_k \in R$, with $t \notin c$, s.t. c is the disjoint union of some subsets e_1 and e_2 and $\mathcal{F}_{e_1} \perp\!\!\!\perp \mathcal{F}_{e_2} \mid \mathcal{F}_t$. Also $\mathcal{F}_{e_1} \vee \mathcal{F}_{e_2} \perp\!\!\!\perp \mathcal{F}_i \mid \mathcal{F}_t$ so by (2.3d) we get $\mathcal{F}_{e_1} \perp\!\!\!\perp \mathcal{F}_{e_2} \vee \mathcal{F}_t \mid \mathcal{F}_i$. We also have $\mathcal{F}_{e_1} \vee \mathcal{F}_{e_2} \perp\!\!\!\perp \mathcal{F}_t \mid \mathcal{F}_i$ since $e_1 \cup e_2 = c$ and t belongs to another set in the finest valid partition w.r.t. i. From the last two relations and by (2.3f) followed by (2.3b) we get $\mathcal{F}_{e_1} \perp\!\!\!\perp \mathcal{F}_{e_2} \mid \mathcal{F}_i$. But by Lemma 3.2, $c \in P_i$ cannot be so decomposed, so $\{e_1, e_2\} = \{c, \emptyset\}$ and we have proved the claim.

So we have shown, by induction, that there exists a junction tree, $\mathbf{G}_d$ on d, where node t has at least $K-1$ neighbors with subtrees corresponding to c_k, $1 \leq k \leq K-1$. Now we modify the original junction tree, $\mathbf{G}$ in $K+1$ steps to get trees $\mathbf{H}$, $\mathbf{H}_0$,$\cdots$, $\mathbf{H}_{K-1}$ as follows:

First we form $\mathbf{H}$ by replacing the subtree in $\mathbf{G}$ on d, with $\mathbf{G}_d$ above, connecting i to t with an edge. By Lemma 3.3, $\mathbf{H}$ is a junction tree on $\{1, \cdots, M\}$.

Let $\mathbf{H}_0$ be the subtree of $\mathbf{H}$ after removing the subtrees around t on c_k, $1 \leq k \leq K-1$. Then $\mathbf{H}_0$ is a junction tree. For each $j = 1, \cdots, K-1$ let $\mathbf{L}_j$ be the subtree of $\mathbf{H}$ on c_j, and let x_j be the node on c_j that was connected to t in $\mathbf{H}$. Then at each step $j = 1, \cdots, K-1$ we form $\mathbf{H}_j$ by joining $\mathbf{H}_{j-1}$ and $\mathbf{L}_j$ by adding the edge between i and x_j (see Figure 3.2.)

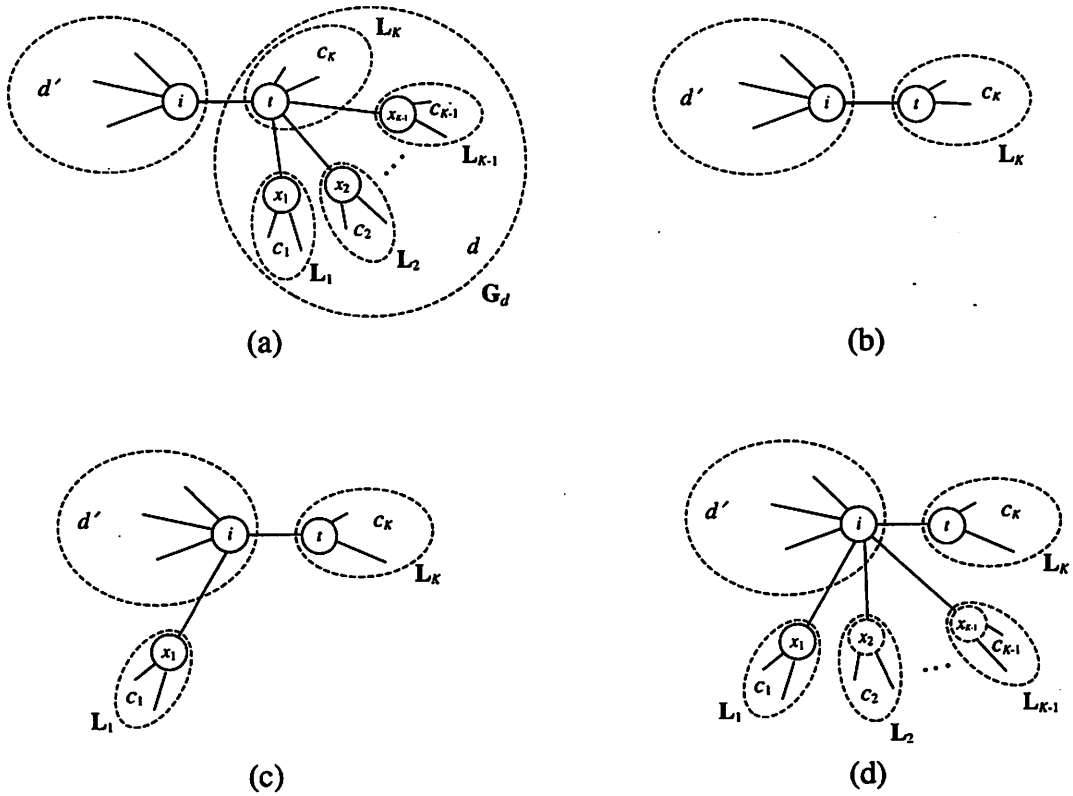


Figure 3.2: Transformation of junction trees used in Theorem 3.4
Dashed ovals depict the junction subtrees, whose name is labelled just outside the oval, on the subset denoted just inside the oval. (a) The original junction tree $\mathbf{H}$, (b) $\mathbf{H}_0$, (c) $\mathbf{H}_1$, and (d) the final junction tree $\mathbf{H}_{K-1}$.

We now show inductively that each $\mathbf{H}_j$ is a junction tree. By induction hypothesis $\mathbf{H}_{j-1}$ is a junction tree. At the same time, $\mathbf{L}_j$, being a subtree of a junction tree, is also a junction tree. Further $\mathcal{F}_{c_j} \perp\!\!\!\perp \bigvee_{r=1}^{j-1} \mathcal{F}_{c_r} \vee \mathcal{F}_{c_K} \vee \mathcal{F}_{d'} \mid \mathcal{F}_i$, since c_j is a set in a valid partition w.r.t. i.

Also, $\mathcal{F}_{c_j} \perp\!\!\!\perp \bigvee_{r=1}^{j-1} \mathcal{F}_{c_r} \vee \mathcal{F}_{c_K} \vee \mathcal{F}_{d'} \mid \mathcal{F}_{x_j}$, since on the junction tree $\mathbf{H}$, node x_j separates c_j from $c_K \cup d' \bigcup_{r=1}^{j-1} c_r$. Then by Lemma 3.3, each $\mathbf{H}_j$ is a junction tree (Note that $\mathbf{H}_{K-1}$ is a junction tree on $\{1, \cdots, M\}$.)

Next we perform the same transformation on $\mathbf{H}_{K-1}$, starting with other neighbors of i. The resulting tree will be a junction tree, and will be compatible with P_i. □

Remark. Notice that the results of this section prescribe a recursive method to find a junction tree, when one exists; we shall describe that method in the next section. In a

sense, Theorem 3.4 and Lemma 3.3 can be viewed as tools to 'cut' and 'paste' smaller junction subtrees, in the quest to find a junction tree on the full set $\{1, \cdots, M\}$. □

3.3 Algorithm to Find a Junction Tree

We will now give an algorithm to find a junction tree when one exists.
Given a set of σ-fields $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$,

Algorithm 3.2. *Pick any node $i \in \{1, \cdots, M\}$ as the root.*

- *If $M = 2$ then the single edge $(1, 2)$ is a junction tree. Stop.*
- *Find the finest valid partition of $\{1, \cdots, M\} \backslash \{i\}$ w.r.t. i, $P_i = \{c_1, \cdots, c_l\}$ (see remarks below).*
- *For j=1 to l*
- *Find a node $t \in c_j$ s.t. $\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_{c_j} \mid \mathcal{F}_t$. If no such node exists, then stop; no junction tree exists.*
- *Find a junction tree on c_j with node t as root. Attach this tree, by adding edge (i, t).*
- *End For*

Proof of correctness of Algorithm 3.2. At each iteration, t is chosen so $\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_{c_j} \mid \mathcal{F}_t$. But we also had $\mathcal{F}_{c_j^c} \perp\!\!\!\perp \mathcal{F}_t \vee \mathcal{F}_{c_j} \mid \mathcal{F}_i$. By (2.3e) the last two relations imply $\mathcal{F}_{c_j} \perp\!\!\!\perp \mathcal{F}_i \vee \mathcal{F}_{c_j^c} \mid \mathcal{F}_t$. But we also have $\mathcal{F}_{c_j} \perp\!\!\!\perp \mathcal{F}_i \vee \mathcal{F}_{c_j^c} \mid \mathcal{F}_i$. So by Lemma 3.3 we have a junction tree at each step. Also, from Theorem 3.4 if the algorithm fails, then there is no junction tree. □

Remark. In the general case of the signed conditional independence, we know of no better way to find the finest valid partition than an exhaustive search in an exponential subset of all the partitions. In the case of unsigned measures, however, we can show that when a junction tree exists, the finest valid partition coincides with the *finest pairwise partition*, which can be found in polynomial time, see Appendix C.

Therefore, starting with a conventional MPF problem, the question of existence of a *probabilistic* junction tree can be answered in polynomial time, as long as an unsigned measure is chosen. □

Chapter 4

Construction of Junction Trees - Lifting

In the previous section we gave an algorithm to find a junction tree, when one exists. In this section we deal with the case when Algorithm 3.2 declares that no junction tree exists for the given set of σ-fields. In particular, we would like to expand the σ-fields in some *minimal* sense, so as to ensure that we can construct a junction tree.

Recall from Section 1.2 that the standard systematic way to construct a conventional junction tree is through the process of moralization and triangulation. The triangulation process in effect expands the cliques of the moral graph – by adding variables,– in a way to create conditional independencies which are required on a junction tree. We will see that our lifting procedure achieves the same goal, with the added possibility of expansion of σ-fields without adding a whole variable direction. In a sense, this amounts to automatically discovering new hidden variables in the space and using them to minimally expand the local domains.

4.1 Lifting

Definition 4.1. Let $(\Omega, \{\mathcal{F}_1, \cdots, \mathcal{F}_M\}, \mu)$ be a given measure space. We call a measure space $(\Omega', \{\mathcal{F}'_1, \cdots, \mathcal{F}'_M\}, \mu')$ a *lifting* of $(\Omega, \{\mathcal{F}_1, \cdots, \mathcal{F}_M\}, \mu)$ if there is a map $f : \Omega' \longrightarrow \Omega$ such that:

- μ' is consistent with μ under the map f, i.e.

$$\forall\, A \in \mathcal{F}_{\{1,\cdots,M\}}, \quad \mu(A) = \mu'(f^{-1}(A)).$$

- For all $i = 1, \cdots, M$, f is $(\mathcal{F}'_i, \mathcal{F}_i)$-*measurable*, i.e.
  $\forall\, A \in \mathcal{F}_i,\ f^{-1}(A) \in \mathcal{F}'_i$.

where for $A \in \Omega$, $f^{-1}(A) := \{\omega' \in \Omega' \ :\ f(\omega') \in A\}$.

□

In words, up to some renaming of the elements, each σ-field $\mathcal{F}_i$ is a sub-σ-field of $\mathcal{F}'_i$, and $\mathcal{F}'_i$ is obtained from $\mathcal{F}_i$ by *splitting* some of the atoms.

Example 4.1. Consider a simple measure space $(\Omega, \mathcal{F}, \mu)$, where $\Omega = \{0, \cdots, 8\}$, and $\mathcal{F}$ is described by its atoms, $\mathcal{A}(\mathcal{F}) = \big\{\{0,1,2,3,5,7\}, \{4,6,8\}\big\}$, see Figure 4.1 below.

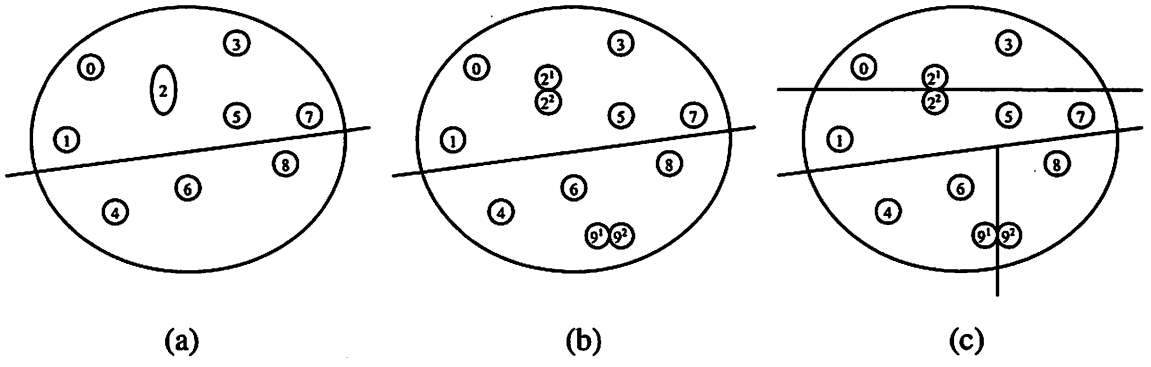


Figure 4.1: Pictorial illustration of Example 4.1
(a) Atoms of $\mathcal{F}$ on Ω, (b) atoms of $f^{-1}(\mathcal{F})$ on Ω', and (c) atoms of $\mathcal{F}'$ on Ω'.

A lifted space $(\Omega', \mathcal{F}', \mu')$ is also pictured in Figure 4.1(c). In particular, we have $\Omega' = \{0, 1, 2^1, 2^2, 3, 4, 5, 6, 7, 8, 9^1, 9^2\}$. The lifting function is defined as $f(2^1) = f(2^2) = 2$ and $f(9^1) = f(9^2) = \emptyset$. The the consistency condition implies that $\mu'(2^1) + \mu'(2^2) = \mu(2)$, $\mu'(9^1) = -\mu'(9^2)$, and $\mu'(i) = i$ for all other $i \in \Omega'$. Pictorially, the element $2 \in \Omega$ has been split into two new elements 2^1 and 2^2, and a virtual element with measure zero has been split into two elements 9^1 and 9^2 with cancelling measures. We can consider $f^{-1}(\mathcal{F})$ as a σ-field on Ω', whose atoms are $\mathcal{A}(f^{-1}(\mathcal{F})) = f^{-1}(\mathcal{A}(\mathcal{F})) = \big\{\{0,1,2^1,2^2,3,5,7\}, \{4,6,8,9^1,9^2\}\big\}$, where we have chosen to assign

the pair of cancelling elements 9^1 and 9^2 to the second atom; note that this σ-field is essentially the same as $\mathcal{F}$. Now let $\mathcal{F}'$ be another σ-field on Ω', with atoms $\mathcal{A}(\mathcal{F}') = \{\{0, 2^1, 3\}, \{1, 2^2, 5, 7\}, \{4, 6, 9^1\}, \{8, 9^2\}\}$, as depicted in Figure 4.1(c). Note that the $(\mathcal{F}', \mathcal{F})$-measurability condition is satisfied, since atoms of $f^{-1}(\mathcal{F})$ are a coarsening of those of $\mathcal{F}'$. □

We now describe the connection of the above concept with our problem.

Let $(\Omega', \{\mathcal{F}'_1, \cdots, \mathcal{F}'_M\}, \mu')$ be a lifting of $(\Omega, \{\mathcal{F}_1, \cdots, \mathcal{F}_M\}, \mu)$ with the lifting map $f : \Omega' \longrightarrow \Omega$ as described in Definition 4.1. Let $\mathbf{G}'$ be a junction tree on $\{1, \cdots, M\}$ corresponding to σ-fields $\{\mathcal{F}'_1, \cdots, \mathcal{F}'_M\}$. We will construct a junction tree $\mathbf{G}''$ from $\mathbf{G}'$ such that the running Algorithm 3.1 on $\mathbf{G}''$ will produce the desired conditional expectations at the appropriate nodes.

For each $i = 1, \cdots, M$, let $\mathcal{G}_i$ be the σ-field on Ω' with atoms $\mathcal{A}(\mathcal{G}_i) = \{f^{-1}(a) : a \in \mathcal{A}(\mathcal{F}_i)\}$, and let $Y_i \in \mathcal{G}_i$ be the random variable with $Y_i(f^{-1}(a)) = X_i(a)$ for all $a \in \mathcal{A}(\mathcal{F}_i)$; so that up to a renaming of the atoms and elements, $(\Omega, \mathcal{F}_i, \mu)$ and $(\Omega', \mathcal{G}_i, \mu')$ are identical measure spaces and X_i and Y_i are identical random variables. Let $\mathbf{G}''$ be a tree with nodes $\{1, \cdots, M, M+1, \cdots, 2M\}$, – with corresponding σ-fields $\{\mathcal{F}'_1, \cdots, \mathcal{F}'_M, \mathcal{G}'_1, \cdots, \mathcal{G}'_M\}$ and random variables $\{1, \cdots, 1, Y_1, \cdots, Y_M\}$,– which is generated by starting with $\mathbf{G}'$ and adding edges $(j, M + j)$ for each $j = 1, \cdots, M$. In words, $\mathbf{G}''$ is a graph obtained from $\mathbf{G}'$ by adding and attaching each node with σ-fields $\mathcal{G}_i$ for $i = 1, \cdots, M$ (which are in turn equivalent to the original $\mathcal{F}_i$'s,) to the node with σ-field $\mathcal{F}'_i$. Then by Lemma 3.3, $\mathbf{G}''$ is a junction tree and hence running Algorithm 3.1 on $\mathbf{G}''$ will produce $\mathbf{E}[\prod_{i=1}^{M} Y_i | \mathcal{G}_j]$ at the node labelled $(M + j)$ for each $j = 1, \cdots, M$. But these are equivalent to $\mathbf{E}[\prod_{i=1}^{M} X_i | \mathcal{F}_j]$ for $j = 1, \cdots, M$ and we have thus solved the probabilistic MPF problem.

So we only need to establish how to lift a certain collection of σ-fields to create the required independencies and form a junction tree.

Suppose we have three σ-fields, $\mathcal{F}_1, \mathcal{F}_2$ and $\mathcal{F}_3$, and we would like to find a lifting $(\Omega', \{\mathcal{F}_1', \mathcal{F}_2', \mathcal{F}_3'\}, \mu')$ of $(\Omega, \{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3\}, \mu)$ so as to have the conditional independence relation $\mathcal{F}_1' \perp\!\!\!\perp \mathcal{F}_3' \mid \mathcal{F}_2'$. Let $a_i \in \mathcal{A}(\mathcal{F}_1)$, $c_k \in \mathcal{A}(\mathcal{F}_2)$ and $b_j \in \mathcal{A}(\mathcal{F}_3)$ be arbitrary atoms. For each c_k, let A_k be the matrix with (i,j) entry equal to $\mu(a_i, b_j, c_k)$. Let $A_k = A_k^1 + A_k^2$ be an additive decomposition of A_k. Then this decomposition corresponds to a lifting of the measure space obtained by splitting the atom c_k of $\mathcal{F}_2$ into two, say c_k^1 and c_k^2, where $\mu'(a_i, b_j, c_k^1)$ and $\mu'(a_i, b_j, c_k^2)$ are defined as the (i,j) entries of A_k^1 and A_k^2 respectively. We will use this decomposition technique to obtain a lifting that makes $\mathcal{F}_1' \perp\!\!\!\perp \mathcal{F}_3' \mid \mathcal{F}_2'$.

Remember first that in order to have this independence, if $\mu(c_k) = 0$, we must have $\mu(a_i, b_j, c_k) = 0$ for all i, j, i.e. the matrix A_k must be zero. Therefore if A_k is a nonzero matrix with zero sum of entries, we first decompose it as the sum of two matrices with nonzero sum of entries. This corresponds to splitting atom c_k in a way that the new atoms have nonzero measure.

Next for each such matrix A_k with nonzero sum of entries, the independence condition corresponding to c_k is exactly the condition that A_k is rank-one. Then if A_k is *not* rank-one, we can use an 'optimal' decomposition of A_k as the sum of say q rank-one matrices (so that none of the matrices are zero-sum).[1] This corresponds to splitting the atom c_k into q atoms, $\{c_k^1, \cdots, c_k^q\}$ where each of c_k^i's render $\mathcal{F}_1$ and $\mathcal{F}_3$ independent. c_k^i's are then the new atoms of the lifted σ-fields $\mathcal{F}_2'$.

It is now clear why we have taken the trouble to develop this theory for *signed* measures. The signed, rank-one decomposition of a matrix can be done in $O(m^{3/2})$, e.g. using standard singular value decomposition techniques, where m is the number of elements of the matrix. On the other hand, the positive rank-one decomposition of a matrix is very hard to find, see e.g. [7]. See also Appendix D for a geometric interpretation of the positive rank-one decomposition.

[1]This can always be done with $q = rank(A_k)$. Obviously $rank(A_k)$ is also a lower bound for q. An optimal decomposition, however, not only aims to minimize q, but also involves minimizing the number of nonzero entries of the decomposing matrices, as discussed in Section 5.

4.2 Algorithm to Construct a Junction Tree

Combining the above ideas with the Algorithm 3.2 we obtain:

Algorithm 4.1. *Pick any node* $i \in \{1, \cdots, M\}$ *as the root.*

- *If* $M = 2$ *then the single edge* $(1, 2)$ *is a junction tree. Stop.*
- *Find any*[2] *valid partition of* $\{1, \cdots, M\} \backslash \{i\}$ *w.r.t.* i, $P_i = \{c_1, \cdots, c_l\}$.
- *For j=1 to l*
- *Find a node* $t \in c_j$ *s.t.* $\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_{c_j} \mid \mathcal{F}_t$. *If no such node exists, then pick any* $t \in c_j$. *Lift* $\mathcal{F}_t$ *by splitting some or all of its atoms as discussed above, so to have* $\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_{c_j} \mid \mathcal{F}_t$.
- *Find a junction tree on* c_j *with node* t *as root. Attach this tree, by adding edge* (i, t).
- *End For*

The resulting measure space $(\Omega', \{\mathcal{F}'_1, \cdots, \mathcal{F}'_M\}, \mu')$ *is a lifting of the original measure space* $(\Omega, \{\mathcal{F}_1, \cdots, \mathcal{F}_M\}, \mu)$, *and the tree generated by this algorithm is a junction tree corresponding to this lifted collection of* σ*-fields.*

Once a junction tree is available, Algorithm 3.1 can be applied to solve the probabilistic MPF problem of Section 3. Note that any algorithm obtained in this manner is simply a reformulation of the original marginalization problem, and can be viewed as a GDL-type algorithm after introduction of certain new variables and potentially unintuitive manipulation of the objective functions. The advantage of our measure-theoretic framework is that it allows for *automation* of this process, without the need for discovering 'hidden variables.'

The complexity of Algorithm 4.1 will be discussed in Section 5.

[2]Although any valid partition will work, in general finer partitions should result in better and less complex algorithms (see Section 5).

4.3 Measure Theory vs. Variables

In Section 3.1, after presenting the measure-theoretic version of Algorithm 3.1, we emphasized the connection with the original GDL by rewriting Algorithm 3.1 in terms of 'variables' representing atoms of each σ-field. Our objective in doing so was to show that, while the language of σ-fields might seem exotic, all our algorithms can be discussed in more conventional terms. We emphasize once again that the 'variables' appearing in equations (3.4) and (3.5) bear no direct relation to the 'variables' of the conventional GDL.

It would probably be useful if one could similarly describe Algorithm 4.1 in terms of some variables, in order to emphasize once again that our algorithms can be thought of in conventional terms. We now describe how to rephrase Algorithm 4.1 in the language of 'variables.' One should note that the main step of Algorithm 4.1, namely the splitting of atoms of a σ-field in order to create conditional independencies, in a sense introduces new 'variables.' We illustrate how to think of Algorithm 4.1 in terms of variables by first considering a basic triangulation step in the conventional GDL along the lines of the basic step of our algorithm, and then describing how the basic step of our algorithm can be though of in terms of 'variables.'

Consider state space $\Omega := \{(x_1, \cdots, x_n) : x_i \in [q_i] := \{1, \cdots, q_i\}\}$ for integers $q_i \geq 2$, with a uniform measure, i.e. $\mu(\mathbf{x}) = 1$ for all $\mathbf{x} \in \Omega$. Let r, s and t be subsets of $\{1, \cdots, n\}$, and define $s' := (r \cap t) \backslash s$. Define $\mathcal{F}_1, \mathcal{F}_2$ and $\mathcal{F}_3$ to be the σ-fields with atoms corresponding to (the level sets of) $\mathbf{x}_r, \mathbf{x}_s$ and $\mathbf{x}_t$ respectively, so that for example $\mathcal{A}(\mathcal{F}_1) = \{\{\mathbf{x} \in \Omega : \mathbf{x}_r = \mathbf{x}_r^1\}, \forall \mathbf{x}_r^1\}$, and so on. We will index the atoms of $\mathcal{F}_1$ by $\mathbf{x}_r^1 \in [q_r]$ where $q_r := \prod_{i \in r} q_i$, and similarly for $\mathcal{F}_2$ and $\mathcal{F}_3$. Note that for convenience we have overloaded symbol $\mathbf{x}_r$, as either an integer in $[q_r]$, or isomorphically as an $|r|$-tuple in $\prod_{i \in r}[q_i]$. The distinction will be apparent from the context. For each atom $\mathbf{x}_s^2$ of $\mathcal{F}_2$, let $A_{\mathbf{x}_s^2}$ be the $(q_r \times q_t)$ matrix of joint measure, with $(\mathbf{x}_r^1, \mathbf{x}_t^3)$ entry equal to $\mu(\mathbf{x}_r^1, \mathbf{x}_t^3, \mathbf{x}_s^2)$, where we use $\mu(\mathbf{x}_r^1, \mathbf{x}_t^3, \mathbf{x}_s^2)$ as the shorthand for the measure of the intersection of the $\mathbf{x}_r^1$st atom of $\mathcal{F}_1$, the $\mathbf{x}_t^3$rd atom of $\mathcal{F}_3$ and the $\mathbf{x}_s^2$nd

atom of $\mathcal{F}_2$. But μ is the uniform measure, so $\mu(\mathbf{x}_r^1, \mathbf{x}_t^3, \mathbf{x}_s^2)$ is 1 iff $\mathbf{x}_r^1, \mathbf{x}_s^2$ and $\mathbf{x}_t^3$ are consistent, i.e.

$$\begin{aligned}\mu(\mathbf{x}_r^1, \mathbf{x}_t^3, \mathbf{x}_s^2) &= 1(\mathbf{x}_{r\cap s}^1 = \mathbf{x}_{r\cap s}^2)1(\mathbf{x}_{s\cap t}^2 = \mathbf{x}_{s\cap t}^3)1(\mathbf{x}_{r\cap t}^1 = \mathbf{x}_{r\cap t}^3) \\ &= 1(\mathbf{x}_{r\cap s}^1 = \mathbf{x}_{r\cap s}^2)1(\mathbf{x}_{s\cap t}^2 = \mathbf{x}_{s\cap t}^3)1(\mathbf{x}_{s'}^1 = \mathbf{x}_{s'}^3) \qquad (4.1)\end{aligned}$$

But we showed earlier that $\mathcal{F}_1 \perp\!\!\!\perp \mathcal{F}_3 \mid \mathcal{F}_2$ iff $A_{\mathbf{x}_s^2}$ is rank-one for all $\mathbf{x}_s^2$, i.e. equation (4.1) factorizes as $f_1(\mathbf{x}_r^1) \cdot f_3(\mathbf{x}_t^3)$. From equation (4.1) above it is obvious that this happens iff $s' = (r \cap t)\backslash s$ is empty, i.e. $r \cap t \subseteq s$. Remember from Section 1.2 that this is precisely the condition for $r - s - t$ to be a (GDL) junction tree.

Now suppose that s' is not empty. Then, after possibly reordering its rows and/or columns, $A_{\mathbf{x}_s^2}$ will be a diagonal block matrix, where each block is a $(q_{r\backslash s'} \times q_{t\backslash s'})$ matrix, and where the diagonal blocks (D) are blocks of all 1's, and off-diagonal blocks (O) are 0 matrices:

$$A_{\mathbf{x}_s^2} = \begin{bmatrix} D & O & \cdots & O \\ O & D & & O \\ \vdots & & \ddots & \vdots \\ O & O & \cdots & D \end{bmatrix} \qquad (4.2)$$

Then the rank-one decomposition of $A_{\mathbf{x}_s^2}$ corresponds to the decomposition where the diagonal blocks are separated:

$$A_{\mathbf{x}_s^2} = \begin{bmatrix} D & O & \cdots & O \\ O & O & & O \\ \vdots & & \ddots & \vdots \\ O & O & \cdots & O \end{bmatrix} + \begin{bmatrix} O & O & \cdots & O \\ O & D & & O \\ \vdots & & \ddots & \vdots \\ O & O & \cdots & O \end{bmatrix} + \cdots + \begin{bmatrix} O & O & \cdots & O \\ O & O & & O \\ \vdots & & \ddots & \vdots \\ O & O & \cdots & D \end{bmatrix} \qquad (4.3)$$

This corresponds to the following decomposition of (4.1):

$$A_{\mathbf{x}_s^2}(\mathbf{x}_r^1, \mathbf{x}_t^3) = \sum_{\mathbf{x}_{s'}} 1(\mathbf{x}_{r\cap s}^1 = \mathbf{x}_{r\cap s}^2)1(\mathbf{x}_{s\cap t}^2 = \mathbf{x}_{x\cap t}^3)1(\mathbf{x}_{s'}^1 = \mathbf{x}_{s'}^3 = \mathbf{x}_{s'})$$

The corresponding lifting is obtained by splitting atom $\mathbf{x}_s^2$ of $\mathcal{F}_2$ by intersecting it with the level-sets of $\mathbf{x}_{s'}$, so the new atoms can be represented by $(\mathbf{x}_s^2, \mathbf{x}_{s'})$. This means that the atoms of $\mathcal{F}_2'$ are the level-sets of $\mathbf{x}_{s\cup s'} = \mathbf{x}_{s\cup(r\cap t)}$. This is precisely what would be done in the GDL framework: in order for the local domains r, s and t

to form a junction chain $r - s - t$, we expand the domain s to contain $r \cap t$. We have therefore shown that when the state space and the σ-fields are represented in terms of orthogonal directions of variables, our condition for independence reduces to that of GDL, and our lifting algorithm will produce the same expansions as in the GDL framework.

Next consider the case when the sample space Ω and the underlying measure μ are arbitrary. For each $i = 1, 2, 3$ let $q_i := |\mathcal{A}(\mathcal{F}_i)|$, and define variables x, y, z taking values in $[q_1], [q_2]$ and $[q_3]$ respectively, corresponding to different atoms of $\mathcal{F}_1, \mathcal{F}_2$ and $\mathcal{F}_3$; this means that $\mathcal{A}(\mathcal{F}_1) = \{\{x = 1\}, \cdots, \{x = q_1\}\}$, $\mathcal{A}(\mathcal{F}_2) = \{\{y = 1\}, \cdots, \{y = q_2\}\}$ and $\mathcal{A}(\mathcal{F}_3) = \{\{z = 1\}, \cdots, \{z = q_3\}\}$. Once again we use the shorthand $\mu(x^1, y^2, z^3)$ for the measure of the intersection of the x^1st atom of $\mathcal{F}_1$ and the y^2nd atom of $\mathcal{F}_2$ and the z^3rd atom of $\mathcal{F}_3$.

As before, for each $y \in [q_2]$ denote by A_y the matrix of the joint measures $\mu(x, z, y)$. Let r_y denote the rank of A_y. Correspondingly, we decompose each A_y as the sum of r_y rank-one matrices, which is equivalent to splitting the yth atom of $\mathcal{F}_2$ into r_y new atoms. To properly index these new atoms, we need to invent a new 'hidden variable' w_y taking value in the set $[r_y]$. Then the new atoms of the lifted σ-field, $\mathcal{F}_2'$ can be indexed by pairs of variables $(y, w_y) \in [q_2] \times [r_y]$. In particular, $\mathcal{F}_2'$ will have $\sum_{y=1}^{q_2} r_y$ atoms, compared to the q_2 atoms of $\mathcal{F}_2$. Note however that in general, the 'directions' corresponding to w_y's are not aligned to those of variables x, y or z, and the pair (y, w_y) does not take value in a product space. Considering some special cases, however, may add some intuition into the process of lifting:

If A_y is full-rank for some y, then w_y is aligned to either x or z: Remember that A_y is a $(q_1 \times q_3)$ matrix. Suppose $r_y = q_1 \leq q_3$, so that A_y has full row rank. We then decompose A_y row-wise, into q_1 single-row matrices. But rows of A_y correspond to the atoms of $\mathcal{F}_1$, which are indexed precisely by the variable x. Therefore for this particular value of y, the variable w_y is identical to the variable x, and the new atoms are indexed by pairs (y, x) where $x \in [q_1]$. This indeed *is* a product space. If A_y is

full-rank for all y's, then creating the desired conditional independence requires a full augmentation of $\mathcal{F}_2$ and $\mathcal{F}_1$, i.e. $\mathcal{F}_2' = \mathcal{F}_1 \vee \mathcal{F}_2$, and atoms of $\mathcal{F}_2'$ correspond to the elements of the product space $[q_1] \times [q_2]$, indexed by pair (x, y).

More generally, if A_y can be rearranged as a block matrix with rank-one blocks such that each row and column of blocks contains at most one nonzero block,– such as in Equation (4.2),– then the 'hidden variables' w_y's will correspond to 'sub-directions' of variables x and z, as determined by the position of the nonzero blocks.

The other trivial case is when A_y is rank-one for some y. In this case no splitting of the yth atom of $\mathcal{F}_2$ is needed, and the corresponding atom of $\mathcal{F}_2$ will be duplicated as an atom of $\mathcal{F}_2'$.

We will close this section by showing that any junction tree obtained using the moralization and triangulation procedure can also be found using Algorithm 4.1.

Suppose we start with the conventional MPF problem discussed in Section 1.1, with $M = |R|$ local domains, $R = \{S_1, \cdots, S_M\}$. Let $\mathbf{G}$ be a junction tree of cliques of the triangulated moral graph for the given MPF problem, as discussed in Section 1.2. For each node i of $\mathbf{G}$, let S_i' be the domain (subset of $[N]$) for the corresponding clique. Then, as discussed before, each S_j of the original GDL local domains is contained in the domain S_i' for a node of $\mathbf{G}$. We can assume then that $\mathbf{G}$ is a tree on $\{1, \cdots, M\}$ by identifying each node of $\mathbf{G}$ with the index i of a local domain it contains; (if an index $i \in \{1, \cdots, M\}$ is left unaccounted for, we will add a node i as a neighbor of a node in $\mathbf{G}$ which also contained local domain S_i, and we set $S_i' = S_i$.) Therefore we will have $S_i \subseteq S_i'$ for each node i of $\mathbf{G}$. Then, as before, we let the state-space Ω to be represented by the GDL variables $\{x_1, \cdots, x_n\}$, equipped with the uniform measure, and identify $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$ as the σ-fields whose atoms are the level-sets of $\{x_{S_1}, \cdots, x_{S_M}\}$ respectively.

We will then run Algorithm 4.1, using the convention that each time a lifting is required to create a conditional independence relation $\mathcal{F}_i \perp\!\!\!\perp \mathcal{F}_c \,|\, \mathcal{F}_t$ (where $c \subset \{1, \cdots, M\}$), we lift $\mathcal{F}_t$ to $\mathcal{F}_t'$, where the atoms of $\mathcal{F}_t'$ are the level-sets of variables

$x_{S'_t}$. Note then that since the atoms of $\mathcal{F}_t$ correspond to x_{S_t} and $S_t \subseteq S'_t$, we have $\mathcal{F}_t \subseteq \mathcal{F}'_t$, confirming that atoms of $\mathcal{F}_t$ are indeed split to obtain those of $\mathcal{F}'_t$. We will now show that with the above choice for the liftings, Algorithm 4.1 can create $\mathbf{G}$ as the junction tree.

We start Algorithm 4.1 with a node i as the root, where i is a leaf-node of $\mathbf{G}$. Let t be the neighbor of i in $\mathbf{G}$ and let $P := \{c_1, \cdots, c_l\}$ be the partition of $\{1, \cdots, M\} \backslash \{t\}$ that is compatible with $\mathbf{G}$. We can assume WLOG that $c_l = \{i\}$. Also let $j_k \in c_k$ be the neighbor of t in $\mathbf{G}$ lying in c_k, for each $k = 1, \cdots, l$. Then the first lifting will be a splitting of atoms of $\mathcal{F}_t$ to ensure $\mathcal{F}_i \perp\!\!\!\perp \bigvee_{k=1}^{l-1} \mathcal{F}_{c_k} \mid \mathcal{F}_t$. We do this, according to the above convention, by splitting atoms of $\mathcal{F}_t$ to form $\mathcal{F}'_t$, whose atoms correspond to the level-sets of $x_{S'_t}$. But since $\mathbf{G}$ is a (GDL) junction tree on domains $\{S'_1, \cdots, S'_M\}$, we must have $S'_{c_j} \bigcap (\cup_{k \neq j} S'_{c_k}) \subseteq S'_t$, for all $j = 1, \cdots, l$. Then, from earlier discussions in this section, $\mathcal{F}_{c_1} \perp\!\!\!\perp \mathcal{F}_{c_2} \perp\!\!\!\perp \cdots \perp\!\!\!\perp \mathcal{F}_{c_l} | \mathcal{F}'_t$ holds, and in particular P is now a valid partition of $\{1, \cdots, M\} \backslash \{t\}$ w.r.t. t (where $\mathcal{F}_t$ has been replaced by its lifting $\mathcal{F}'_t$). This valid partition can be used in the next call to the algorithm to ensure that the neighboring subtrees of t in the junction tree will be identical to those in $\mathbf{G}$. It is then easy to see that following similar choices in each step of Algorithm 4.1, we will end up with $\mathbf{G}$ as the junction tree, while $(\Omega, \{\mathcal{F}_1, \cdots, \mathcal{F}_M\}, \mu)$ is lifted to $(\Omega', \{\mathcal{F}'_1, \cdots, \mathcal{F}'_M\}, \mu')$ where $\Omega' = \Omega$, μ' is the uniform measure on Ω', and $\mathcal{F}'_i$ is the σ-field whose atoms are the level-sets of variables $x_{S'_i}$. Therefore we have shown that using Algorithm 4.1, one can always retrieve a junction tree equivalent to any (GDL) junction tree obtained using the moralization and triangulation procedure.

Chapter 5

Complexity Issues

In this section we discuss the computational complexity of the methods proposed in this dissertation so far. We start by estimating the complexity of computing the conditional expectation of a random variable given a σ-field, with respect to a signed measure.

5.1 Computing Conditional Expectations

Consider the problem of efficiently computing the conditional expectation $\mathbf{E}[X|\mathcal{G}]$. Let $\mathcal{A}'(\mathcal{G}) = \{\beta_1, \cdots, \beta_m\}$ be the atoms of $\mathcal{G}$ with nonzero measure, and let $\mathcal{F}$ denote the sigma field generated by X, with atoms $\mathcal{A}(\mathcal{F}) = \{\alpha_1, \cdots, \alpha_l\}$. Let W denote the $l \times m$ matrix of conditional measures, with (i,j) entry $\mu(\alpha_j|\beta_i)$. Also define $\mathbf{x}$ as an $m \times 1$ column vector with entries $\mathbf{x}_j = X(\alpha_j)$. Then calculating $\mathbf{E}[X|\mathcal{G}]$ is precisely equivalent to computing $\mathbf{y} := W \cdot \mathbf{x}$.

At first look it appears that calculating $W \cdot \mathbf{x}$ requires $l \cdot m$ multiplications and $l \cdot (m-1)$ additions. However, using the fact that matrix W is known a priori, in many cases we can perform the calculations with a lower number of operations, as suggested in Example 2.1. To illustrate this in the context of Algorithm 3.1, note that $\mathbf{E}[X|\mathcal{G}]$ is precisely the message sent from $\mathcal{F}$ to $\mathcal{G}$, in a junction 'tree' in the form $\mathcal{F}$— $\mathcal{G}$. Now let $\mathcal{H} := \{\emptyset, \Omega\}$ be the trivial σ-field on the same state space Ω. We will lift the space so as to create the conditional independence $\mathcal{F}' \perp\!\!\!\perp \mathcal{G}' \mid \mathcal{H}'$. We

will then have a junction chain $\mathcal{F}'$— $\mathcal{H}'$— $\mathcal{G}'$. Then it is easy to see that $\mathbf{E}[X|\mathcal{G}]$ will be the same as the message from $\mathcal{H}'$ to $\mathcal{G}'$ on this junction tree (after message from $\mathcal{F}'$ has been received at $\mathcal{H}'$).

Now to create the desired conditional independence, we need the rank-one decomposition of W (note that W – the matrix of conditional measures,– is rank-one iff A_Ω – the matrix of joint measures $\mu(\alpha_j, \beta_i, \Omega) = \mu(\alpha_j, \beta_i)$ is rank-one).

Suppose now that W has rank $r \leq min(l, m)$. Denote the ith row of W by W_i, and define $z(i)$ to be the number of nonzero elements of W_i. Let $\{W_{i_1}, \cdots, W_{i_r}\}$ be a linearly independent subset of rows of W, chosen so as to minimize the sum $z := \sum_{k=1}^{r} z(i_k)$. Then for each $i = 1, \cdots, l$, $W_i = \sum_{k=1}^{r} C_{i,k} W_{i_k}$ for some $l \times r$ matrix C. Incidentally this matrix has the property that after possibly reordering its rows and/or columns, it has the $r \times r$ identity matrix as a submatrix. Denote by z' the number of nonzero elements of the $(l-r) \times r$ submatrix of C when this $r \times r$ identity submatrix has been removed.

We then have the following rank-one decomposition: $W = \sum_{k=1}^{r} C^k \cdot W_{i_k}$, where C^k is the kth column of C. Corresponding to this decomposition, $W \cdot \mathbf{x}$ can be computed as $\mathbf{y} = \sum_{k=1}^{r} C^k \cdot W_{i_k} \cdot \mathbf{x}$. Calculating $W_{i_k} \cdot \mathbf{x}$ requires $z(i_k)$ multiplications and $(z(i_k)-1)$ additions. Note that, as discussed before, while the vector $\mathbf{x}$ is arbitrary, the matrix W is known a priori and hence we can exclude multiplications and additions by the 0's. To compute $\mathbf{y} = \sum_{k=1}^{r} C^k \cdot (W_{i_k} \cdot \mathbf{x})$ given scalars $(W_{i_k} \cdot \mathbf{x})$ we need an additional z' multiplications and $(z' - l + r)$ additions, where z' was defined above for matrix C.

Therefore, to compute $W \cdot \mathbf{x}$ we need a total of $(z'+z)$ multiplications and $(z'-l+z)$ additions. Define $\chi := 2(z' + z) - l$ as the *edge complexity* associated with the hypothetical directed edge from $\mathcal{F}$ to $\mathcal{G}$, which is the number of additions and multiplications required for computing $\mathbf{E}[X|\mathcal{G}]$. Note from definitions that $r \leq z \leq r\,m$ and $l - r \leq z' \leq (l-r)m$, and therefore $l \leq z' + z \leq l \cdot m$. Therefore using this technique, as compared to the straightforward multiplications and summations suggested

by equation 2.1, depending on the structure of the matrix W, there is potential for a saving by a factor of m in the total operations required to carry out the computations.

Also note that one can always use a row-wise rank-one decomposition (i.e. $W = \sum_{k=1}^{l} e^k W_k$, where e^k is an $(l \times 1)$ column vector with a 1 at the kth position and 0's everywhere else). Then an upper-bound on χ is $(2\,\mathrm{nz} - l)$, where variable nz is defined to be the total number of nonzero elements in matrix W.

Next we will discuss the complexity of the junction tree algorithm 3.1.

5.2 Complexity of the PJT Algorithm

First notice that, as discussed in Section 4.3, any junction tree obtained using the moralization and triangulation (possibly after introduction of hidden variables), corresponds to a junction tree on a lifting of the original σ-fields, – where liftings are done in the whole-variable directions,– which can also be discovered by Algorithm 4.1. Algorithm 3.1 on this junction tree will then be equivalent to GDL on the junction tree of cliques of the triangulated graph. In this sense then, using our framework we can always find a marginalization algorithm which is at least as good as GDL.

With that point noted, in this section we will discuss the complexity of Algorithm 3.1 in terms of the sizes of the σ-fields on the junction tree, rather than those of the original σ-fields before possible lifting. It is clear that the lifted σ-fields can be substantially larger than the original ones. This is also the case in the original GDL framework, where triangulation can produce enormous cliques.

Let $\mathbf{G}$ be a junction tree with σ-fields $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$, as defined above, and let $\{X_1, \cdots, X_M\}$ be arbitrary random variables with $X_i \in \mathcal{F}_i$. Denote by q_i the number of atoms of the σ-field $\mathcal{F}_i$, so $q_i = |\mathcal{A}(\mathcal{F}_i)|$.

It can be seen that, in general, the sample space Ω can have as many as $\prod_{i=1}^{M} q_i$ elements and thus full representation of σ-fields and the measure function requires exponentially large storage resources. Fortunately, however, a full representation is not required. Along each edge (i, j) on the tree, Algorithm 3.1 only requires local

computation of $\mathbf{E}[X|\mathcal{F}_i]$ for a random variable $X \in \mathcal{F}_j$. This only requires a $q_i \times q_j$ table of the joint measures of the atoms of $\mathcal{F}_i$ and $\mathcal{F}_j$. For an arbitrary edge (i,j), let $\mathcal{A}(\mathcal{F}_i) = \{a_1, \cdots, a_{q_i}\}$ and $\mathcal{A}(\mathcal{F}_j) = \{b_1, \cdots, b_{q_j}\}$ be the sets of atoms of $\mathcal{F}_i$ and $\mathcal{F}_j$. Define $W(i,j)$ to be the $q_i \times q_j$ matrix with (r,s) entry equal to $\mu(a_s|b_r)$; note that from Lemma A.1, (possibly after trivial simplification of the problem by eliminating the events with measure zero,) no atom of $\mathcal{F}_j$ can have measure 0, so $\mu(a_s|b_r)$ is defined for all atoms of $\mathcal{F}_j$. Then once a junction tree has been found, we need only keep $2(M-1)$ such matrices (corresponding to the $(M-1)$ edges of the tree) to fully represent the algorithm, for a total of $2\sum_{(i,j)\text{ an edge}} q_i q_j$ storage units.

As defined in the previous section, for each directed edge (i,j) in $\mathbf{G}$ let $\chi(i,j)$ be the corresponding edge complexity, i.e. the arithmetic complexity of computing $\mathbf{E}[X_i|\mathcal{F}_j]$. From the Algorithm 3.1, calculation of the conditional expectation of the product given a single σ-field $\mathcal{F}_i$ with the most efficient schedule requires updating of the messages from the leaves towards the node i. Each edge is activated in one direction, and at each non-leaf node l the messages need to be multiplied to update the message from l to its neighbor in the direction of i. This requires, for each edge (k,l), an additional q_l multiplications. Thus the grand total arithmetic operations needed to calculate $\mathbf{E}[\prod_j X_j|\mathcal{F}_i]$ is $\sum_{(k,l)} (\chi(k,l)+q_l)$, where the summation is taken over all the directed edges (k,l) approaching i.

The complexity of the full algorithm, in which $\mathbf{E}[\prod_j X_j|\mathcal{F}_i]$ is calculated for all $i = 1, \cdots, M$, can also be found using similar ideas. For each node k, let $d(k)$ denote the number of the neighbors of k on the tree. Then for each directed edge (k,l), first the $d(k)-1$ messages from other neighbors of k must be multiplied by X_k and then the conditional expectation given $\mathcal{F}_l$ must be taken. For each non-leaf node k, calculating the product of X_k and all the products of $d(k)-1$ out of $d(k)$ messages incoming to k requires $(3d(k)-4)q_k$ multiplications, using a method similar to the one described in Section 5 of [2]. So the total number of operations required for the

full algorithm is

$$\sum_{(k,l) \text{ a dir. edge}} \chi(k,l) + \sum_{k \text{ non-leaf}} (3d(k) - 4)q_k$$

In particular, if $\mathbf{G}$ is a chain $\mathcal{F}_1 - \mathcal{F}_2 - \cdots - \mathcal{F}_M$, the complexity of Algorithm 3.1 becomes $\sum_{i=1}^{M-1} \left(\chi(i,i+1) + \chi(i+1,i)\right) + 2\sum_{i=2}^{M-1} q_i$.

Using the upper-bound $\chi(i,i+1) \leq (2\,\mathrm{nz}(i,i+1) - q_{i+1})$ derived in the previous section, the above complexity of Algorithm 3.1 on a chain is upper-bounded by $4\sum_{i=1}^{M-1} \mathrm{nz}(i,i+1)$. Here we have used the fact that $\mathrm{nz}(i,i+1) = \mathrm{nz}(i+1,i)$, i.e. the number of nonzero elements of matrices $W(i,i+1)$ and $W(i+1,i)$ are equal.

5.3 Complexity of Lifting

In this section discuss the complexity of Algorithm 4.1. Recall that the lifting process at each step of Algorithm 4.1 can increase not only the number of atoms of the σ-field which is being processed, but also the overall number of states in the state space. As mentioned before, in general it is not possible to a priori get a reasonable estimate of these figures, since the rank-one decompositions can be arbitrarily irregular and unpredictable. Rather, the complexity will depend on the choices made in each lifting.

Consider again the setup used above, with σ-fields $\mathcal{F}_1, \mathcal{F}_2$ and $\mathcal{F}_3$, where lifting is done to create $\mathcal{F}_1' \perp\!\!\!\perp \mathcal{F}_3' \mid \mathcal{F}_2'$. Also for each $c_k \in \mathcal{A}(\mathcal{F}_2)$, let A_k be the matrix of joint measures $\mu(a_i, b_j, c_k)$ where $\mathcal{A}(\mathcal{F}_1) = \{a_i\}$ and $\mathcal{A}(\mathcal{F}_3) = \{b_j\}$. Then clearly, as noted before, the number of atoms of the lifted σ-field, $\mathcal{F}_2'$ is minimized by using a decomposition of A_k as the sum of r_k rank-one matrices, where $r_k = rank(A_k)$. However, this is not the only factor determining the complexity of the algorithm. Each time a nonzero entry of A_k is decomposed as sum of nonzero numbers, in effect new states are added to the state space. It is therefore preferable to choose decompositions that minimize splitting of nonzero entries. For example, the splitting in equation (4.3) does not create any new states, since the nonzero blocks are non-overlapping. As another example, suppose $A_k = \left(\begin{smallmatrix} 2 & 3 \\ 3 & 4 \end{smallmatrix}\right)$. Then decomposition $A_k = \left(\begin{smallmatrix} 2 & 3 \\ 0 & 0 \end{smallmatrix}\right) + \left(\begin{smallmatrix} 0 & 0 \\ 3 & 4 \end{smallmatrix}\right)$ is

preferable to $A_k = \binom{1\,2}{1\,2} + \binom{1\,1}{2\,2}$, since the latter results in creating four new states in the state space.

An interesting observation is that in the GDL framework, once a representation of the space is decided by the choice of variables, the operations to create independencies (e.g. moralization and triangulation) will never expand the state space; all expansions of local domains are achieved by addition of whole variables, equivalent to splitting of non-overlapping rank-one blocks in our framework.

The complexity of each lifting is related (polynomially) to the number of states in the lifted state space; in the above setup this number is $|\mathcal{A}(\mathcal{F}'_1 \vee \mathcal{F}'_2 \vee \mathcal{F}'_3)|$. Algorithm 4.1 consists of liftings at different stages, so the overall complexity of Algorithm 4.1 is on the order of that of the most complex lifting in the process. Now consider the case when the algorithm is run on σ-fields $\mathcal{F}_1, \cdots, \mathcal{F}_M$, each with q atoms, and when the σ-fields are processed in the order of their index to form a junction chain. It is clear that the original state space can have as many as q^M states, indicating that the general algorithm is exponentially complex in M. Once the lifting has been done to create the independence condition $\mathcal{F}'_1 \perp\!\!\!\perp \mathcal{F}'_{\{3,\cdots,M\}} \mid \mathcal{F}'_2$, the number of atoms of $\mathcal{F}'_2$ can grow to q^2. The next lifting, however, will only involve σ-fields $\mathcal{F}'_2$ through $\mathcal{F}'_M$. Thus the number of relevant states cannot exceed $|\mathcal{A}(\mathcal{F}'_{\{2,\cdots,M\}})|$. But $|\mathcal{A}(\mathcal{F}'_{\{2,\cdots,M\}})| \leq |\mathcal{A}(\mathcal{F}'_2)| \cdot |\mathcal{A}(\mathcal{F}'_{\{3,\cdots,M\}})| \leq q^2 \cdot q^{(M-2)} = q^M$.

In other words, the size of the relevant state space will not exceed q^M, and hence the complexity of none of the consequent liftings will exceed that of the first lifting. An upper bound on the complexity of Algorithm 4.1 is therefore $k\,M\,q^{\frac{3}{2}M}$ for a constant k, where $k\,q^{\frac{3}{2}M}$ is an upper bound on the complexity of rank-one decomposition of a matrix with q^M elements.

However, even though the general form of Algorithm 4.1 is complex, there are justifications for why it can be a useful practical tool. Firstly, note that as mentioned in Section 5.2, a full description of the marginalization algorithm requires storing only the $W(i,j)$ matrices of the conditional measures of the atoms of the neighboring

σ-fields, and a full representation of the exponentially large state space is unnecessary. Secondly, remember that Algorithm 4.1 can be executed off-line once and for all. Once a junction tree has been created, the corresponding marginalization algorithm can be used for all possible inputs, namely the random variables $X_1, \cdots, X_M$. Therefore the cost of creating an efficient algorithm is amortized over many future uses.

As mentioned before, our framework is general enough to be able to take advantage of partial independencies in the objective functions; together with ideas of lifting, we have an automatic algorithm for exploiting these structures.

Chapter 6

Examples

In this section we consider several examples where GDL-based algorithms are applicable. We have created a MATLAB library containing the necessary functions to set up a general marginalization problem and create a junction tree. The examples in this section are processed using that code. To explain the algorithms resulting from our code, we have described each example in detail. Note however that to generate the marginalization algorithms, no calculations were done manually and the entire process was automatic. The details were presented here simply to clarify the end result. In each example we further compare the complexity of our message-passing algorithm with the naive implementation of GDL.

We note that there do exist other methods and techniques that deal with each specific class of problems. Such methods are tailored for a single class of problems and are not applicable to the general case. For instance, the paper of Zhang and Poole [55] gives a procedure that is somewhat more general than the conventional GDL for generation low complexity algorithms, when the local functions are of specific form called 'causally independent.' Our measure-theoretic framework, on the other hand is general in handling all marginalization problems. It is also worth noting that in one of the examples considered,– that of decoding linear block codes,– our procedure is able to discover the minimal complexity trellis-based decoding algorithm.

Our Example 2.1 from Section 2 can be generalized as follows:

Example 6.1. Let A be a finite set of size n, and for each $i = 1, \cdots, M$, let X_i be a real function on A. Let μ be a real (weight) function on A^M. We would like to compute the weighted average of the product of X_i's, namely

$$E := \sum_{a_1 \in A} \cdots \sum_{a_M \in A} \prod_{i=1}^{M} X_i(a_i) \mu(a_1, \cdots, a_M)$$

Suppose that the weight function is in the following form:

$$\mu(a_1, \cdots, a_M) = \prod_{i=1}^{M} f_i(a_i) + \prod_{i=1}^{M} g_i(a_i)$$

As long as the weight function μ is not in product form, the naive version of the GDL algorithm will prescribe

$$E = \sum_{a_1 \in A} X_1(a_1) \cdots \sum_{a_M \in A} X_M(a_M) \mu(a_1, \cdots, a_M)$$

requiring $O(n^M)$ additions and multiplications, corresponding to the junction tree in Figure 6.1.

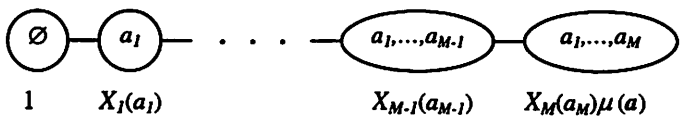


Figure 6.1: GDL junction tree for Example 6.1

Now, Let Ω be the product space A^M with signed measure μ, and for $i = 1, \cdots, M$, let $\mathcal{F}_i$ be the σ–field containing the planes $a_i = c$, so $X_i \in \mathcal{F}_i$. Let $\mathcal{F}_0 = \{\emptyset, \Omega\}$ be the trivial σ–field. Then the problem is to find the conditional expectation of the product of the X_i's given $\mathcal{F}_0$.

The best junction tree is obtained by lifting the space so that given $\mathcal{F}_0'$ all other σ–fields are mutually conditionally independent. To do this, we split the atom Ω of $\mathcal{F}_0$ into two atoms Ω_1 and Ω_2. In effect, for each element $(a_1, \cdots, a_M)$ of Ω, the new space Ω' has two elements $(a_1, \cdots, a_M) \cap \Omega_1$ and $(a_1, \cdots, a_M) \cap \Omega_2$. The new weight function is defined on Ω' as $\mu'((a_1, \cdots, a_M^1) \cap \Omega_1) = \prod_{i=1}^{M} f_i(a_i)$ and

$\mu'((a_1, \cdots, a_M) \cap \Omega_1) = \prod_{i=1}^{M} g_i(a_i)$.

Then there is a star-shaped junction tree on $\{0, \cdots, M\}$ with node 0 at the center. The message passing algorithm on this tree is:

$$\begin{aligned} E &= \mathbf{E}[\prod X_i | \mathcal{F}_0](\Omega) \\ &= \mathbf{E}[\prod X_i | \mathcal{F}_0'](\Omega_1) + \mathbf{E}[\prod X_i | \mathcal{F}_0'](\Omega_2) \\ &= \prod_{i=1}^{M} \mathbf{E}[X_i | \mathcal{F}_0'](\Omega_1) + \prod_{i=1}^{M} \mathbf{E}[X_i | \mathcal{F}_0'](\Omega_2) \\ &= \prod_{i=1}^{M} \sum_{a_i \in A} X_i(a_i) f_i(a_i) + \prod_{i=1}^{M} \sum_{a_i \in A} X_i(a_i) g_i(a_i) \end{aligned}$$

Note that this requires only $O(Mn)$ additions and multiplications. □

Example 6.2. Partition Function in a Circuit-Switched Network. In a circuit-switched network, one is interested in finding the invariant distribution of calls in progress along routes of the network. It can be shown (see [45]) that the invariant distribution has the form

$$\pi(a_1, \cdots, a_M) = \frac{X_1(a_1) \cdots X_M(a_M)}{Z} \prod_{j=1}^{L} 1(\sum_{i \in R_j} a_i < n_j)$$

Here a_i is the number of calls along route i; M is the total number of routes; $X_i(a_i)$ is a known function (invariant distribution of a_i if the links had an infinite number of circuits); L is the number of links in the network; n_j is the capacity of link j and $R_j \subset \{1, \cdots, M\}$ is the index set of routes that use link j. Finally Z is a normalizing factor called the partition function, and is defined by

$$Z := \sum_{a_1, \cdots, a_M} \prod_{i=1}^{M} X_i(a_i) \prod_{j=1}^{L} 1(\sum_{i \in R_j} a_i < n_j)$$

Therefore in order to calculate the invariant distribution, one only needs to calculate the partition function Z. Having this in mind, we consider the following simplest case:

Let $X_1(a_1), \cdots, X_M(a_M)$ be arbitrary functions with integer variables $a_1, \cdots, a_M$. Let $f(s) = 1(0 \leq s < n) \cdot g(s)$ be an arbitrary function of the integer variable s, which vanishes when $s \notin \{0, \cdots, n-1\}$. We would like to calculate the following weighted marginalization

$$Z = \sum_{a_1} \cdots \sum_{a_M} f(a_1 + \cdots + a_M) \prod_{i=1}^{M} X_i(a_i)$$

Relying on orthogonal directions of independent variables to represent the state space, a GDL algorithm will suggest

$$Z = \sum_{a_1} X_1(a_1) \sum_{a_2} X_2(a_2) \cdots \sum_{a_M} f(a_1 + \cdots + a_M) X_M(a_M)$$

Define $s_j = \sum_{i=1}^{j} a_i$. Then, noting that $s_1 \leq s_2 \leq \cdots \leq s_M$ and that $f(s_M) = 0$ for $s_M \geq n$, one can interpret the above sum as the following:

$$Z = \sum_{a_1=0}^{n-1} X_1(a_1) \sum_{a_2=0}^{n-1-s_1} X_2(a_2) \cdots \sum_{a_M=0}^{n-1-s_{M-1}} f(a_1 + \cdots + a_M) X_M(a_M)$$

Note however that even this simplified version requires $O(n^M)$ arithmetic operations.

We will now show that our Algorithm 4.1 creates a simple junction tree, and implementation of Algorithm 3.1 on that junction tree substantially simplifies the marginalization problem.

Define a sample space $\Omega = \{(a_1, \cdots, a_M) : 0 \leq a_i,\ 0 \leq s_M < n\}$ with uniform measure, $\mu(\omega) = 1$ for all $\omega \in \Omega$; here for ease of notation, we denote elements of Ω by ω, and define $a_i(\omega)$ to be the ith coordinate of ω. We also define, as before, $s_j(\omega) = \sum_{i=1}^{j} a_i(\omega)$. When there is no risk of confusion, we drop the explicit dependence of a_i's and s_j's on ω.

For each $i = 1, \cdots, M$, let $\mathcal{F}_i$ be the σ-field with atoms $\mathcal{A}(\mathcal{F}_i) = \{\{\omega \in \Omega : a_i = j\} : j = 0, \cdots, n-1\}$. Define also σ-field $\mathcal{G}$ with atoms $\mathcal{A}(\mathcal{G}) = \{\{\omega \in \Omega : s_M = j\} : j = 0, \cdots, n-1\}$. Next we view local functions as random variables measurable in the corresponding σ-field: $X_i \in \mathcal{F}_i$ and $g \in \mathcal{G}$. The problem is then to find $\mathbf{E}[g \prod_{i=1}^{M} X_i]$.

We now follow Algorithm 4.1 step by step to create a junction tree. We pick $\mathcal{G}$ as the root node and $\mathcal{F}_M$ as the neighboring node, and lift to create conditional independence $\mathcal{G} \perp\!\!\!\perp \mathcal{F}_{\{1:M-1\}} \mid \mathcal{F}_M$: For each atom $f_k := \{\omega \in \Omega : a_M = k\}$ of $\mathcal{F}_M$, let A_k be the $(n \times n^{M-1})$ matrix with (i,j) entry $\mu(g_i, h_j, f_k)$, where $g_i := \{\omega \in \Omega : s_M = i\}$ and $h_j := \{\omega \in \Omega : a_{l+1} = j_l \text{ for } l = 0, \cdots, M-2\}$ are atoms of $\mathcal{G}$ and $\mathcal{F}_{\{1:M-1\}}$ respectively, and $(j_{M-2} j_{M-3} \cdots j_0)$ is the n-ary expansion of j, so that $j = \sum_{l=0}^{M-2} j_l n^l$. Then it can be seen that the (i,j) entry, $\mu(g_i, h_j, f_k) = 1(i = k + \sum_{l=0}^{M-2} j_l)$ where $1(\cdot)$ is the indicator function.Clearly A_k has rank $(n-k)$ since it has precisely $(n-k)$ nonzero (linearly independent) rows, namely rows k through $n-1$. Hence we can split atom f_k into $(n-k)$ new atoms, corresponding to row-wise decomposition of A_k. Let $\mathcal{F}'_M$ be the σ-field whose atoms are these split atoms. Then

$$\begin{aligned}\mathcal{A}(\mathcal{F}'_M) &= \{f'_{l,m} : 0 \le l \le m \le n-1\} \\ &:= \left\{\{\omega \in \Omega : a_M = l \ \& \ s_M = m\} : 0 \le l \le m \le n-1\right\}\end{aligned}$$

In particular, $\mathcal{F}'_M$ has $n(n+1)/2$ atoms.

Next we try to lift to create conditional independence $\mathcal{F}'_M \perp\!\!\!\perp \mathcal{F}_{\{1:M-2\}} \mid \mathcal{F}_{M-1}$: For each atom $f_k := \{\omega \in \Omega : a_{M-1} = k\}$ of $\mathcal{F}_{M-1}$, let A_k be the $(\frac{n(n+1)}{2} \times n^{M-2})$ matrix with (i,j) entry $\mu(f'_{l_i,m_i}, h_j, f_k)$. Here $f'_{l,m}$'s are the atoms of $\mathcal{F}'_M$ and (l_i, m_i) is a map from $\{0, \cdots, n(n+1)/2 - 1\}$ to $\{0, \cdots, n-1\} \times \{0, \cdots, n-1\}$ such that $l_i \le m_i$. Also $h_j := \{\omega \in \Omega : a_{l+1} = j_l \text{ for } l = 0, \cdots, M-3\}$ are atoms of $\mathcal{F}_{\{1:M-2\}}$, and $(j_{M-3} j_{M-4} \cdots j_0)$ is the n-ary expansion of j, so that $j = \sum_{l=0}^{M-3} j_l n^l$. Again it can be seen that the (i,j) entry, $\mu(f'_{l_i,m_i}, h_j, f_k) = 1(m_i = l_i + k + \sum_{l=0}^{M-3} j_l)$. It is evident that this function only depends on row index i through $m_i - l_i$; for a given $r \in \{0, \cdots, n-1\}$, all the rows with $m_i - l_i = r$ are identical, and hence can be grouped together for rank-one decomposition. Also notice that

$$\begin{aligned}&\{f'_{l,m} : 0 \le l \le m \le n-1, m-l = r\} = \\ &\quad \left\{\{\omega \in \Omega : a_M = l \ \& \ s_{M-1} = m-l\} : 0 \le l \le m \le n-1, m-l = r\right\} \\ &\qquad = \left\{\{\omega \in \Omega : s_{M-1} = r\}\right\}\end{aligned}$$

Further, each A_k has $(n-k)$ such groups of rows, corresponding to $k \leq r \leq n-1$. Hence we can split atom f_k into $(n-k)$ new atoms. Let $\mathcal{F}'_{M-1}$ be the σ-field whose atoms are these split atoms. Then

$$\mathcal{A}(\mathcal{F}'_{M-1}) = \left\{\{\omega \in \Omega : a_{M-1} = l \ \& \ s_{M-1} = m\} : 0 \leq l \leq m \leq n-1\right\}$$

Note that $\mathcal{F}'_{M-1}$ has $n(n+1)/2$ atoms.

This procedure will be repeated and it can be seen that the lifted σ-fields will have atoms:

$$\mathcal{A}(\mathcal{F}'_i) = \left\{\{\omega \in \Omega : a_i = l \ \& \ s_i = m\} : 0 \leq l \leq m \leq n-1\right\}$$

The corresponding junction tree is the chain pictured in Figure 6.2.

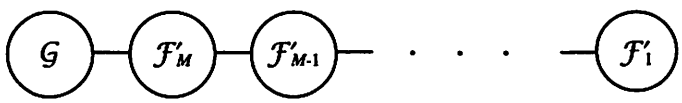


Figure 6.2: Junction tree for Example 6.2

Examining the matrices of joint measures along edges of this chain, it can be verified that the corresponding junction tree algorithm is equivalent to the following:

$$Z = \sum_{s_M=0}^{n-1} \mu(s_M) \sum_{s_{M-1}=0}^{s_M} X_M(s_M - s_{M-1}) \cdots \sum_{s_2=0}^{s_3} X_3(s_3 - s_2) \sum_{s_1=0}^{s_2} X_2(s_2 - s_1) X_1(s_1) \quad (6.1)$$

This requires only $O(Mn^2)$ arithmetic operations.

The form given in equation (6.1) suggests that, after introduction of variables s_i, GDL can also produce (6.1). However, variables s_i as defined above are not independent, so GDL can never come up with (6.1) exactly. To account for this problem in GDL, one can take s_i's to be free and independent variables taking value in $\{0, \cdots, n-1\}$, and then redefine functions $X_i(s_i, s_{i-1})$ to equal $X_i(s_i - s_{i-1})$ when

$0 \leq s_{i-1} \leq s_i$, and be zero otherwise. Then indeed GDL will come up with a form similar to (6.1) (note that upper limits in the sums cannot depend on variable s_i; a postprocessor can however realize that the terms that correspond to $s_{i-1} > s_i$ will vanish.)

Notice however, the *automatic* nature of our procedure: although we have introduced the auxiliary variables s_i to analytically express the atoms of each σ-field and to express the algorithm of equation (6.1) in closed form, the automatic procedure of Algorithm 4.1 does not rely on these variables. There is no need for pre- or post-processing of data or introduction of variables. Given a representation of the σ-fields $\mathcal{G}$ and $\mathcal{F}_1, \cdots, \mathcal{F}_M$, a computer program will automatically come up with algorithm of equation (6.1) without any assistance. This is the essence of our method. □

In the next example we show that Pearl's treatment of the belief propagation algorithm in the case of a node with *disjunctive interaction* or *noisy-or-gate* causation ([33], Section 4.3.2) can be viewed as a special case of our algorithm:

Example 6.3. Bayesian Network with Disjunctive Interaction. Let the binary inputs $U = (U_1, U_2, \cdots, U_n)$ be the parents of the binary node X in the Bayesian network of Figure 6.3, interacting on X through a noisy-or-gate.

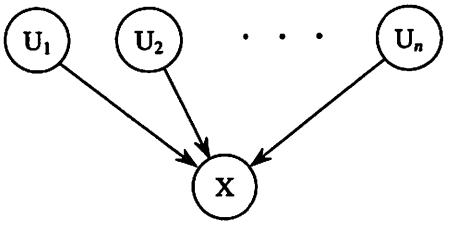


Figure 6.3: Bayesian network of Example 6.3

This means that there are parameters $q_1, \cdots, q_n \in [0, 1]$ so that

$$P(X = 0 \mid U) = \prod_{i \in T_u} q_i$$

$$P(X = 1 \mid U) = 1 - \prod_{i \in T_u} q_i$$

where $T_u := \{i \,:\, U_i = 1\}$.

Normal moralization and triangulation technique applied to this graph will give a single clique with all the variables. However, because of the structure in the problem, a better solution exists.

Let $\mathcal{F}_X, \mathcal{F}_1, \cdots, \mathcal{F}_n$ be the σ-fields generated by the (independent) variables x, $u_1, \cdots, u_n$ respectively. All variables are binary so each of the σ-fields has precisely two atoms. In our framework, let the 'random variables' be (the function) $1 \in \mathcal{F}_X$, and $\pi_X(u_i) = P(U_i = u_i) \in \mathcal{F}_i$ for $i = 1, \cdots, n$, with the underlying joint measure on $x, u_1, \cdots, u_n$ defined to be $\mu(x, u_1, \cdots, u_n) = P(X = x \mid U = (u_1, \cdots, u_n))$. Then F_i's are *not* mutually conditionally independent given $\mathcal{F}_X$, however the following simple lifting of space will create the independence: Let a variable x' be defined to take value in $0, 1, 2$, where the event $\{x' = 0\}$ corresponds to $\{x = 0\}$, and $\{x' = 1\} \cup \{x' = 2\}$ correspond to $\{x = 1\}$. Extend the measure as follows:

$$\mu'(x', u_1, \cdots, u_n) = \begin{cases} \prod_{i \in T_u} q_i & \text{if } x' = 0 \\ -\prod_{i \in T_u} q_i & \text{if } x' = 1 \\ 1 & \text{if } x' = 2 \end{cases}$$

Then we see that in this *lifted* spaces, the σ-fields $\mathcal{F}'_i$ (generated by variables u_i respectively) are mutually conditionally independent given $\mathcal{F}'_{X'}$ (the σ-field generated by variable x'.) Then we have a junction tree in the shape of a star, with $\mathcal{F}'_{X'}$ corresponding to the central node. The junction tree algorithm will calculate the following marginalized random variable at the node corresponding to $\mathcal{F}'_{X'}$:

$$\beta(x') = \begin{cases} \prod_{i=1}^n \left(P(U_i = 0) + P(U_i = 1)\, q_i\right) & \text{if } x' = 0 \\ -\prod_{i=1}^n \left(P(U_i = 0) + P(U_i = 1)\, q_i\right) & \text{if } x' = 1 \\ 1 & \text{if } x' = 2 \end{cases}$$

Then the *belief* at X is the function

$$\mathrm{BEL}(x) = \begin{cases} \prod_{i=1}^n \left(P(U_i = 0) + P(U_i = 1)\, q_i\right) & \text{if } x = 0 \\ 1 - \prod_{i=1}^n \left(P(U_i = 0) + P(U_i = 1)\, q_i\right) & \text{if } x = 1 \end{cases}$$

where we have merged the atoms $x' = 1$ and $x' = 2$ of $\mathcal{F}'_{X'}$ to get back $x = 1$. This is essentially the same as Equation (4.57) in [33]. □

Example 6.4. Hadamard Transform. Let $x_1, \cdots, x_n$ be binary variables and let $f(x_1, \cdots, x_n)$ be a real function of x_i's. The Hadamard transform of f is defined as:

$$g(y_1, \cdots, y_n) := \sum_{x_1, \cdots, x_n} \prod_{i=1}^{n} (-1)^{x_i y_i} f(x_1, \cdots, x_n)$$

where $y_1, \cdots, y_n$ are binary variables.

Since our framework is particularly useful when the underlying functions are structured, we consider the case when f is a symmetric function of $x_1, \cdots, x_n$, i.e. f depends only on the sum of the x_i's. Then it is easy to verify that when f is symmetric, its Hadamard transform, g is also a symmetric function.

We now set up the problem in our framework. Let Ω be $\{0,1\}^{2n}$ with elements $\omega = (x_1, \cdots, x_n, y_1, \cdots, y_n)$. Let $\mathcal{F}$ and $\mathcal{G}$ be the σ–fields in which respectively f and g are measurable; in our case of symmetric f and g, $\mathcal{A}(\mathcal{F}) = \{\alpha_k \text{ for } k = 0, \cdots, n\} = \{\{\omega : \sum_i x_i = k\} \text{ for } k = 0, \cdots, n\}$ and $\mathcal{A}(\mathcal{G}) = \{\beta_k \text{ for } k = 0, \cdots, n\} = \{\{\omega : \sum_i y_i = k\} \text{ for } k = 0, \cdots, n\}$.

Next we note that all the factors involving terms $(-1)^{x_i y_i}$ can be summarized as a signed measure μ on $\mathcal{F} \vee \mathcal{G}$ as follows:

$$\begin{aligned}
\mu(\alpha_j, \beta_k) &= \sum_{\omega \in \alpha_j \cap \beta_k} (-1)^{\sum_i x_i y_i} \\
&= \sum_{\substack{\omega: \sum_i x_i = j, \\ \sum_i y_i = k}} (-1)^{\sum_i x_i y_i} \\
&= \sum_{(x_1, \cdots, x_n): \sum_i x_i = j} (-1)^{\sum_{i=1}^{k} x_i}
\end{aligned}$$

Note that μ can be stored in a $(n+1) \times (n+1)$ table.

Now we have a junction tree with only two nodes, corresponding to $\mathcal{F}$ and $\mathcal{G}$, and the marginalization is done as follows:

$$\begin{aligned}
g(\beta_k) &= \mathbf{E}[f|\mathcal{G}] \\
&= \sum_{j=0}^{n} f(\alpha_j) \mu(\alpha_j, \beta_k)
\end{aligned}$$

where $f(\alpha_j) = f\big((x_1, \cdots, x_n) : \sum_i x_i = j\big)$ and $g(\beta_k) = g\big((y_1, \cdots, y_n) : \sum_i y_i = k\big)$.

This requires only n additions and $(n+1)$ multiplications for each of $(n+1)$ possible values of g. □

Example 6.5. Probabilistic State Machine. Consider the Bayesian network depicted in Figure 6.5, where u_i, s_i and y_i denote inputs, hidden states and the outputs of a chain of length n. Let m be the memory of the state machine, so that each state s_i can be taken to be $(u_{i-m}, \cdots, u_{i-1})$, where for ease of notation we fix $u_{-1} = u_{-2} = \cdots := 0$ so we will not have to worry about stages $i < m$.

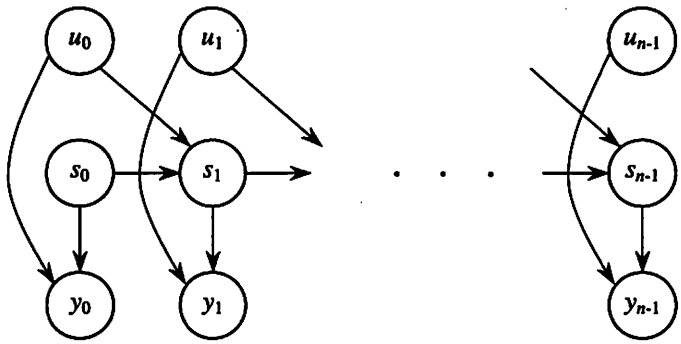


Figure 6.4: Bayesian network for a probabilistic state machine

A specific device used to find the maximum-likelihood input symbols given y_i's, is a trellis, which is equipped with an efficient marginalization algorithm, namely the BCJR algorithm on the trellis (see [5]). As mentioned before however, this is not a general method, but rather a method tailored only for a specific class of marginalization problems. Description of that algorithm in GDL format with variables requires introduction of complicated hidden variables (see Example 6.8).

The general and automatic GDL solution for this problem is the BCJR algorithm on a junction chain (rather than a trellis). The functions involved are $P(s_0)$,$P(u_i)$, $P(y_i^*|u_i, s_i)$ and $P(s_i|u_{i-1}, s_{i-1})$, so the GDL local domains for this chain are $\{s_0\}$, and $\{u_i\}$ and $\{u_{i-m}, \cdots, u_i\}$ for $i = 1, \cdots, n-1$. Assuming binary inputs and outputs, the BCJR algorithm will require about $3 \cdot \sum_v d(v)\, q_v$ operations, where v ranges over the GDL local domains, $d(v)$ is the number of neighbors of v, and $q_v = 2^{|v|}$ is the size

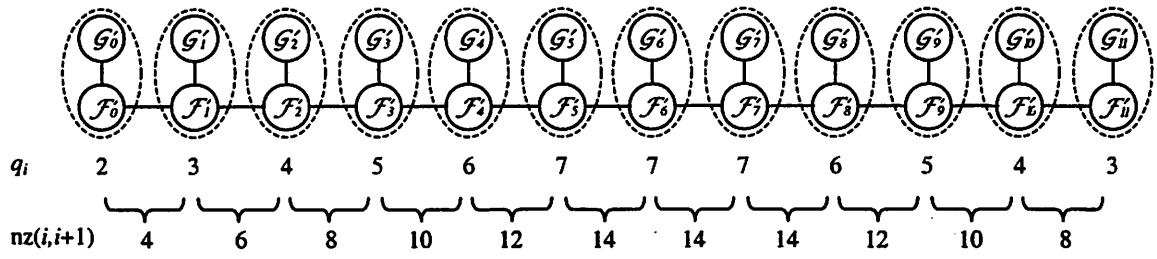


Figure 6.5: Junction tree created for chain of length 12 and memory 6

of the set of possible values for the variables in domain v (see [2]). The complexity of GDL then grows roughly as $3(n-m)2^{m+2}$ (the exact formula used below in Table 6.1 is $3(2^{m+1}(2n-2m+1)-6)$).

Now consider a case when the output of the state machine depends on the input and state in a simple, but non-product form. For the purposes of this example, we have chosen the output y_i to be the outcome of an 'OR' gate on the state s_i and input u_i, passed through a binary symmetric channel, i.e.

$$P(y_i|u_i,s_i) = (1-p)1(y_i = \vee_{j=0}^{m} u_{i-j}) + p \cdot 1(y_i \neq \vee_{j=0}^{m} u_{i-j})$$

where '$\vee$' indicates a logical 'OR', and $1(\cdot)$ is the indicator function.
We formed the Ω space as $\{0,1\}^n$, with elements $\omega = (u_0, \cdots, u_{n-1})$. Then each functions $P(y_i^*|u_i,s_i)$ is measurable in a σ–field $\mathcal{F}_i$, with two atoms $\{\omega : \vee_{j=0}^{m} u_{i-j} = 0\}$ and $\{\omega : \vee_{j=0}^{m} u_{i-j} = 1\}$. Since we like to calculate the posterior probabilities on each input u_i, we also include σ–fields $\mathcal{G}_i$ each with two atoms $\{\omega : u_i = 0\}$ and $\{\omega : u_i = 1\}$.

We then run our algorithm to create a junction tree on the $\mathcal{F}_i$'s and the $\mathcal{G}_i$'s, lifting the space whenever needed. The result is a chain consisting of the $\mathcal{F}_i'$'s with each $\mathcal{G}_i'$ hanging from its corresponding $\mathcal{F}_i'$ (see Figure 6.5).

We have run the algorithm on chains with various values of n and m. Table 6.1 compares the complexity of the message-passing algorithm on the probabilistic junction tree and the normal GDL (BCJR) algorithm. GDL complexity was estimated as discussed above. The complexity of probabilistic algorithm on the lifted chain, as discussed in Section 5, is at most (4 nz) where 'nz' is the total number of nonzero en-

(n, m)	(9,5)	(9,6)	(10,5)	(10,6)	(10,7)	(11,5)	(11,6)	(11,7)	(12,5)	(12,6)	(12,7)
nz	70	70	82	84	82	94	98	98	106	112	114
GDL ops	1710	2670	2094	3438	5358	2478	4206	6894	2862	4974	8430
PGDL ops	292	292	343	352	343	394	412	412	445	472	481

Table 6.1: Comparison between complexity of GDL and probabilistic GDL. Here nz denotes the total number of nonzero entries in the tables of pairwise joint measures. 'GDL ops' and 'PGDL ops' are the estimated total number of arithmetic operations required by GDL and our probabilistic algorithm respectively.

tries in the tables of pairwise joint measures. We add to this the number of additions required to get the desired marginals at the level of atoms of the original σ-fields to obtain the figures listed in the table.

The details of the case $n = 12, m = 6$ have been portrayed in Figure 6.5. The number underneath each $\mathcal{F}_i'$ is q_i, the number of atoms of $\mathcal{F}_i' \vee \mathcal{G}_i'$ after lifting has been done. Note that with our setup, originally $\mathcal{F}_0 \vee \mathcal{G}_0$ has 2 atoms, and all other $\mathcal{F}_i \vee \mathcal{G}_i$'s have 3 atoms. The numbers under the brackets denote $\mathrm{nz}(i, i+1)$, the number of nonzero elements in the matrix of joint measures between the atoms of adjacent nodes. Here we note that the pattern that can be seen in these sequences of numbers is not a coincidence. We will reexamine this general problem at the end of this chapter. □

Example 6.6. Exact Decoding of Low Density Parity Check Codes. In this example we apply our method to some LDPC codes of small block size over a memoryless channel. The codes used in this example are depicted in Figure 6.6 as bipartite graphs, in which the upper nodes corresponds to the bits and the lower nodes correspond to the parity checks (see [20]). In each case we will obtain an exact algorithm to find the *a posteriori* probabilities for each bit. We then compare these algorithms with the exact algorithms obtained under GDL using the triangulation method (see [38] and [2]). As we will see, for a randomly generated LDPC code, the cliques of

the triangulated graph are almost as big as the whole variable set, resulting in poor algorithms. On the other hand, using our framework we are able to find exact algorithms that are much less complex. In fact, as we will show later, the algorithms derived using our method in this case are equivalent to the best known exact decoding algorithms for linear block codes, i.e. the BCJR algorithm on the minimal trellis of the code, see e.g. [21; 5].

Let $\mathbf{H} \in \mathrm{GF}(2)^{m \times n}$ be the parity-check matrix for a binary linear block code with m check-nodes and block size n. Then the column vector $\mathbf{x} = [x_1 \cdots x_n]^T$ in $\mathrm{GF}(2)^n$ is a codeword iff it satisfies $\mathbf{H}\mathbf{x} = \mathbf{0}$. Given observations $\mathbf{y}^*$ of $\mathbf{x}$, the *a posteriori* joint probability of $\mathbf{x}$ factors as:

$$P^*(x_1, \cdots, x_n) = \frac{1}{Z} \prod_{i=1}^{n} P(x_i) P(y_i^* | x_i) \prod_{j=1}^{c} 1(H_j \cdot \mathbf{x} = 0)$$

Here again, Z is a normalizing factor and H_j is the jth row of the matrix $\mathbf{H}$. We are interested in finding $P^*(x_i)$'s, the marginals of P^* for $i \in \{1, \cdots, n\}$.

To set up the problem in our framework, we define the sample space Ω to be the set of codewords, i.e. $\Omega := \{(x_1, \cdots, x_n) : \mathbf{H}\mathbf{x} = \mathbf{0}\}$. We choose the uniform measure on Ω, so that $\mu(\mathbf{x}) = 1$ for all $\mathbf{x} \in \Omega$. For $i = 1, \cdots, n$, define σ-field $\mathcal{F}_i$ with atoms $\mathcal{A}(\mathcal{F}_i) = \{\{\mathbf{x} \in \Omega : x_i = j\} : j = 0, 1\}$. Finally we define random variables $X_i \in \mathcal{F}_i$ to equal $P(x_i = j) P(y_i^* | x_i = j)$. Then the marginals $P^*(x_i)$ correspond to the conditionals expectations $\mathbf{E}[\prod_j X_j | \mathcal{F}_i]$.

We ran our MATLAB lifting code for a number of randomly chosen LDPC codes with small block size (Figure 6.6), and in each case formed a junction chain with lifted σ-fields $\mathcal{F}_1', \cdots, \mathcal{F}_n'$. Table 2 summarizes each chain.

Each code is presented with block length n and parity checks m, as well as code rate and parameter c, the number of 1's per each column of $\mathbf{H}$ (checks-per-bit). For each code, we have listed q_i's, the number of atoms of the lifted σ-fields $\mathcal{F}_i'$. We have also reported functions $\mathrm{nz}(i, i+1)$, the number of nonzero entries of the matrix

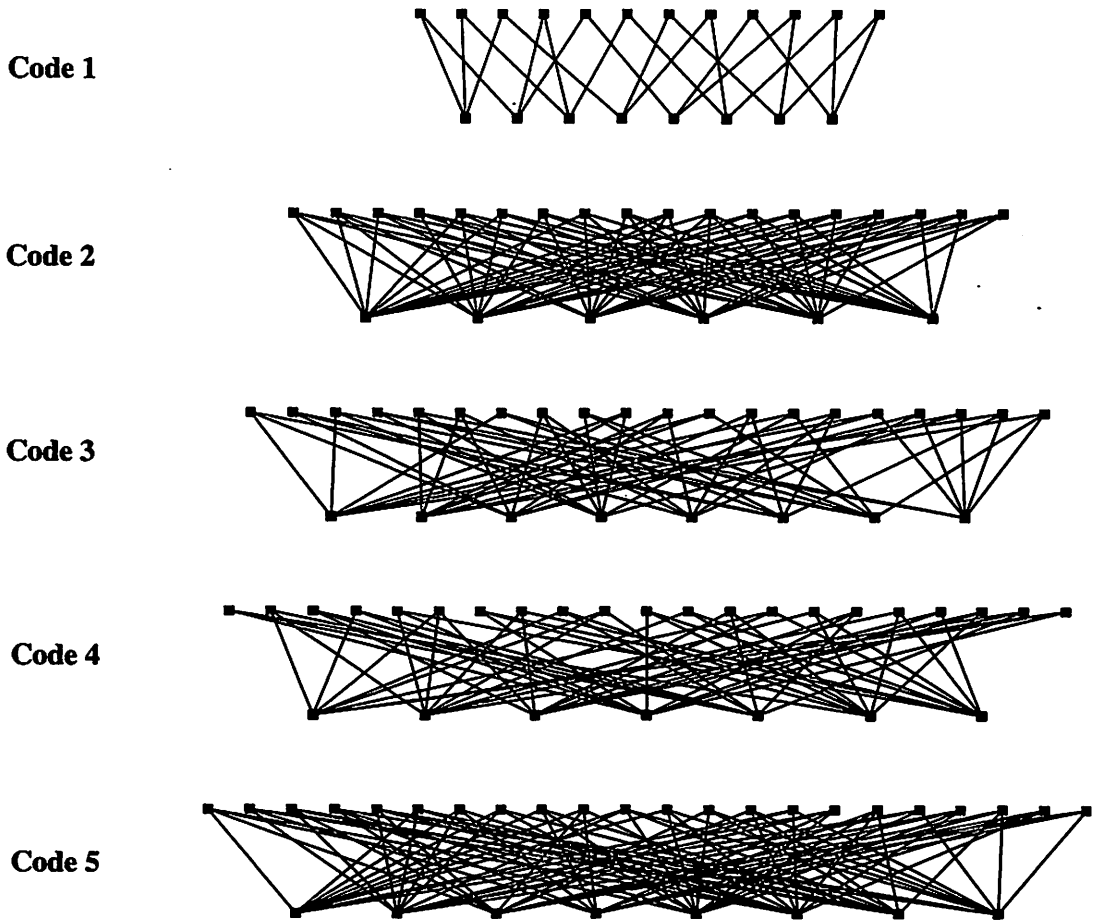


Figure 6.6: LDPC Codes of Example 6.6

of joint measures of the atoms of neighboring σ-fields; as discussed in Section 5, total arithmetic complexity of Algorithm 3.1 on the chain is at most $4\sum_i \mathrm{nz}(i, i+1)$. Finally, to get the marginal $P^*(x_i)$ from the conditional expectation $\mathbf{E}\left[\prod_j X_j \middle| \mathcal{F}_i\right]$, we need $(q_i - 2)$ additions. We have therefore calculated and reported the total arithmetic complexity of our exact algorithm.

For each code, we also triangulated the moral graph and found the junction tree of the cliques suitable for GDL-type algorithm. Specifically, for each LDPC code we form the moral graph; this is a graph with n nodes corresponding to the bits of a codeword, and where an edge (i, j) exists iff bits i and j are involved in a common parity constraint. We then triangulate this graph using the algorithm given in Section 3.2.4 of [33], by adding edges (chords) in each unchorded cycle of the graph with length

at least 4. The cliques of this graph can be put on a junction tree. In Table 6.2 for each code we report the size of the cliques of the triangulated graph; the clique sizes should be compared with $\log_2$ of the number of atoms of the lifted σ-fields in the left column. We also report the 'edge sizes', i.e. the size of the intersection of the cliques that are connected by an edge in the junction tree of the cliques; these in turn should be compared with $\log_2$ of the functions $\mathrm{nz}(i, i+1)$. Finally we use $4\sum_{v \text{ clique}} d(v)q_v$ as the (approximate) arithmetic complexity of GDL algorithm, where $d(v)$ is the number of neighbors of clique v in the junction tree and $q_v = 2^{|v|}$ is the cardinality of the set of possible values for variables in v (see [2]).

It can be seen that the triangulation method is incapable of recognizing the structure that exists within each parity check term $1(H_j \cdot \mathbf{x} = 0)$. It is therefore forced to treat each indicator function as a complete function of the bit variables that appear in that check. As a result of this and the interconnection between the bits, clique sizes are almost as big as the block size n, resulting in very inefficient algorithms.

On the other hand, avoiding representation with variables, our measure-theoretic approach is able to discover minimal liftings that render the σ-fields independent, and hence come up with much more efficient marginalization algorithms. Since the general complexity bounds we have given on the lifting procedure in Section 5.3 are exponential, it is not clear for how large a block size it will be practicable to have a complete implementation of the lifting algorithm. Our MATLAB implementations were far from optimized, and were run on a 850MHz Pentium III Laptop, with 256MB of memory. It took about half an hour for the largest reported code size, namely $n = 22$. This suggests that a highly optimized implementation on a state of the art computing cluster could carry out lifting for much larger sized codes.

We will now show that applying Algorithm 4.1 to the problem of decoding a linear block code will produce a junction tree which is equivalent to the minimal trellis representing the code. Let Ω be a binary linear (n, k) code, with codewords $\mathbf{x} \in \Omega$

Code 1: $n = 12, m = 8, c = 2$, rate=5/12	
Lifting (Algorithm 4.1)	Moralization/Triangulation
q_i: 2, 4, 4, 8, 4, 4, 8, 8, 8, 8, 4, 2	Clique sizes: 5, 6, 6, 5, 5, 5
$\mathrm{nz}(i, i+1)$: 4, 4, 8, 8, 4, 8, 8, 16, 8, 8, 4	Edge sizes: 4, 5, 4, 3, 4
Total PGDL complexity: **360**	Total GDL complexity: $\mathbf{1.8 \times 10^3}$
Code 2: $n = 18, m = 6, c = 4$, rate=2/3	
Lifting (Algorithm 4.1)	Moralization/Triangulation
q_i: 2, 4, 8, 16, 16, 32, 64, 64, 64, 64, 32, 64, 64, 32, 16, 8, 4, 2	Clique sizes: 18
$\mathrm{nz}(i, i+1)$: 4, 8, 16, 32, 32, 64, 128, 128, 128, 64, 64, 128, 64, 32, 16, 8, 4	Edge sizes: –
Total PGDL complexity: $\mathbf{5.95 \times 10^3}$	Total GDL complexity: $\mathbf{1.0 \times 10^6}$
Code 3: $n = 20, m = 8, c = 3$, rate=3/5	
Lifting (Algorithm 4.1)	Moralization/Triangulation
q_i: 2, 4, 8, 16, 32, 64, 128, 256, 256, 512, 512, 256, 256, 128, 64, 32, 16, 8, 4, 2	Clique sizes: 14, 16, 16, 16
$\mathrm{nz}(i, i+1)$: 4, 8, 16, 32, 64, 128, 256, 512, 512, 1024, 512, 512, 256, 128, 64, 32, 16, 8, 4	Edge sizes: 13, 15, 14
Total PGDL complexity: $\mathbf{2.02 \times 10^4}$	Total GDL complexity: $\mathbf{1.4 \times 10^6}$
Code 4: $n = 21, m = 7, c = 3$, rate=2/3	
Lifting (Algorithm 4.1)	Moralization/Triangulation
q_i: 2, 4, 8, 16, 32, 64, 128, 256, 256, 256, 128, 128, 64, 64, 64, 64, 32, 16, 8, 4, 2	Clique sizes: 16, 17, 17
$\mathrm{nz}(i, i+1)$: 4, 8, 16, 32, 64, 128, 256, 512, 512, 256, 256, 128, 128, 128, 128, 64, 32, 16, 8, 4	Edge sizes: 14, 15
Total PGDL complexity: $\mathbf{2.14 \times 10^4}$	Total GDL complexity: $\mathbf{1.8 \times 10^6}$
Code 5: $n = 22, m = 8, c = 4$, rate=7/11	
Lifting (Algorithm 4.1)	Moralization/Triangulation
q_i: 2, 4, 8, 16, 32, 64, 64, 128, 128, 256, 256, 256, 256, 128, 128, 128, 64, 32, 16, 8, 4, 2	Clique sizes: 21, 21
$\mathrm{nz}(i, i+1)$: 4, 8, 16, 32, 64, 128, 128, 256, 256, 256, 512, 512, 256, 256, 256, 128, 64, 32, 16, 8, 4	Edge sizes: 20
Total PGDL complexity: $\mathbf{2.40 \times 10^4}$	Total GDL complexity: $\mathbf{8.4 \times 10^6}$

Table 6.2: LDPC Results from Example 6.6 (see also Figure 6.6)

defined by $\mathbf{H} \cdot \mathbf{x} = \mathbf{0}$. As before, we define μ to be the uniform measure on Ω, so that $\mu(\mathbf{x}) = 1$ for all $\mathbf{x} \in \Omega$, and for each $i = 1, \cdots, n$, define $\mathcal{F}_i$ as the σ-field whose atoms are the level sets of the ith bit of $\mathbf{x}$, i.e. $\mathcal{A}(\mathcal{F}_i) = \big\{\{\mathbf{x} \in \Omega : x_i = j\} : j = 0, 1\big\}$.

Let T be a rank n trellis with binary labels. This is defined as a graph whose vertices are partitioned into $n+1$ *time stages*, $V_0, V_1, \cdots, V_n$, with $|V_0| = |V_n| = 1$, such that any edge in T connects vertices from neighboring time stages. Further, each edge is labelled with a symbol from the set $\{0, 1\}$. We denote by $E_{i-1,i}$ the set of edges between V_{i-1} and V_i, and for each edge e, we denote by $\lambda(e)$ the label associated with edge e. More generally, we define $E_{m,m+d}$ to be the collection of all paths $(e_{m,m+1}, \cdots, e_{m+d-1,m+d})$ between V_m and V_{m+d}, with $e_{j,j+1} \in E_{j,j+1}$. Also for each path $p \in E_{m,m+d}$, we denote by $\lambda(p)$ the $(d \times 1)$ vector of the labels of the edges of p. We will further assume that if p and p' are two different paths in $E_{0,n}$, then $\lambda(p) \neq \lambda(p')$.

A trellis T is said to represent code Ω if the labelled paths of T from V_0 to V_n are precisely the codewords of Ω, i.e. $\Omega = \{\lambda(p),\ p \in E_{0,n}\}$. Therefore we identify codewords of Ω by the paths $p \in E_{0,n}$ in T. With this representation, we can view measure function μ as a function of n-tuples of edges, such that $\mu(p) = 1$ iff p is a path in $E_{0,n}$. Also, using this representation for Ω, the two atoms of $\mathcal{F}_i$ are precisely $\{(e_{0,1}, \cdots, e_{n-1,n}) \in E_{0,n} : \lambda(e_{i-1,i}) = j\}$ for $j \in \{0, 1\}$.

Now note that each trellis representing Ω can be viewed as a lifting $(\Omega, \{\mathcal{F}'_i\}, \mu)$ of $(\Omega, \{\mathcal{F}_i\}, \mu)$, where for each $i = 1, \cdots, n$, $\mathcal{F}'_i$ is defined as the σ-field whose atoms correspond to *all* the edges in $E_{i-1,i}$. Specifically $\mathcal{A}(\mathcal{F}'_i) = \big\{\{(e_{0,1}, \cdots, e_{n-1,n}) \in E_{0,n} : e_{i-1,i} = e^0\},\ e^0 \in E_{i-1,i}\big\}$. Then the atoms of $\bigvee_{i=m}^{m+d} \mathcal{F}'_{i+1}$ correspond in similar way to the paths in $E_{m,m+d}$.

We now claim that for every trellis representing code Ω, the chain $\mathcal{F}'_1 — \mathcal{F}'_2 — \cdots — \mathcal{F}'_n$ is a junction chain. To see this, let e be an edge in $E_{m-1,m}$, corresponding to an atom of $\mathcal{F}'_m$. Also let $p^1 := (e^1_{0,1}, \cdots, e^1_{m-2,m-1}) \in E_{0,m-1}$ and $p^2 := (e^2_{m,m+1}, \cdots, e^2_{n-1,n}) \in E_{m,n}$ be representing atoms of $\bigvee_{i=1}^{m-1} \mathcal{F}'_i$ and $\bigvee_{i=m+1}^{n} \mathcal{F}'_i$

respectively. Then from definitions above, $\mu(p^1, e, p^2)$ is 1 precisely if (p^1, e, p^2) is a path in $E_{0,n}$, and is zero otherwise. In other words,

$$\mu(p^1, e, p^2) = 1\big(\text{fin}(e^1_{m-2,m-1}) = \text{init}(e)\big) \cdot 1\big(\text{fin}(e) = \text{init}(e^2_{m,m+1})\big),$$

where for an edge $e \in E_{i-1,i}$, we define $\text{init}(e) \in V_{i-1}$ and $\text{fin}(e) \in V_i$ as the initial and final vertices of e, respectively. This means precisely that for a fixed $e \in E_{m-1,m}$, $\mu(p^1, e, p^2)$ factorizes as a product of functions of p^1 and p^2, and hence $\bigvee_{i=1}^{m-1} \mathcal{F}'_i \perp\!\!\!\perp \bigvee_{i=m+1}^{n} \mathcal{F}'_i \mid \mathcal{F}'_m$, proving that $\mathcal{F}'_1 — \mathcal{F}'_2 — \cdots — \mathcal{F}'_n$ is a junction chain.

Finally, recall that for any linear block code there is a minimal representing trellis T_{min} that has the smallest number of vertices and edges at each time stage. The minimal trellis has the property that for any two initial (or terminal) paths $p^1, p^2 \in E_{0,m}$ (or $p^1, p^2 \in E_{m-1,n}$), we have $p^1 = p^2$ iff $\lambda(p^1) = \lambda(p^2)$. This means that the atoms of $\bigvee_{i=1}^{m} \mathcal{F}_i$ can be precisely identified by the paths in $E_{0,m}$ (which are also the atoms of $\bigvee_{i=1}^{m} \mathcal{F}'_i$). Similarly, the atoms of $\bigvee_{i=m}^{n} \mathcal{F}_i$ can be precisely identified by the paths in $E_{m-1,n}$ (which are also the atoms of $\bigvee_{i=m}^{n} \mathcal{F}'_i$). Starting Algorithm 4.1 with the original σ-fields $\{\mathcal{F}_i\}$, the lifting of $\mathcal{F}_i$ will then amount to rank-one decomposition of a matrix of 0's and 1's with non-overlapping blocks of 1's, of the type mentioned in Section 4.3, where a 1 in the matrix of joint measures corresponds precisely to the event that an initial path of T_{min} (representing an atom of $\bigvee_{j=1}^{i-1} \mathcal{F}'_i$) and a terminal path of T_{min} (representing an atom of $\bigvee_{j=i+1}^{n} \mathcal{F}_i$) can be joined, with the appropriate value for the ith code-bit (representing an atom of $\mathcal{F}_i$) to form a complete codeword in Ω. From linearity of the code and minimality of trellis T_{min}, the lifted atoms will be none other than the edges in $E_{i-1,i}$ of T_{min}, and the lifted σ-field will be precisely $\mathcal{F}'_i$. Therefore Algorithm 4.1 in this case will produce a junction chain which is equivalent to the minimal trellis of the code, for the given order of processing of the nodes. It is also easy to verify that Algorithm 3.1 on this junction tree is the same as the BCJR algorithm on the minimal trellis. □

Example 6.7. CH-ASIA. Our next example is CH-ASIA from [10], pp. 110-111.

The chain graph of Figure 6.7 describes the dependencies between the variables.

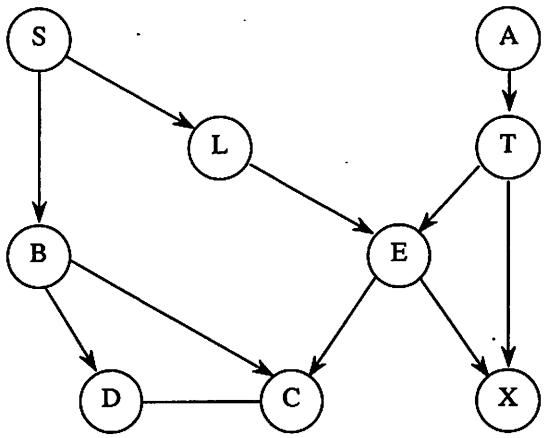


Figure 6.7: Graph of CH-ASIA Example

The problem is to marginalize $P(S, A, L, T, B, E, D, C, X)$, which is the product of the following functions: $P(S)$, $P(A)$, $P(L|S)$, $P(T|A)$, $P(B|S)$, $1(E = L \vee T)$, $P(X|E)$, $f(C, D, B)$, $g(C, B, E)$ and $h(B, E)$.

Again we set up measurable spaces, with σ–fields corresponding to each of the above functions. We then ran the lifting algorithm to find a junction tree in form of a chain, as in the previous example. This time, however, due to lack of structure at the level of the marginalizable functions, (i.e. the aforementioned conditional probabilities,) the algorithm produced exactly a junction tree that one could obtain by the process of moralization and triangulation at the level of original variables. In other words, all liftings were done by addition of one or more 'whole' orthogonal directions (i.e. GDL variables) of the Ω space to the σ–fields. After reconverting σ–fields to 'variables', the junction tree we obtained was the following:

Figure 6.8: Junction tree for CH-ASIA Example

In this case, our algorithm has reduced to GDL. □

Example 6.8. Probabilistic State Machine Revisited. In the automatic treatment of Example 6.5 it was seen that the lifted junction chains exhibit strong struc-

tures, suggesting that there is a simple closed-form solution for the general problem. This in fact is the case, and as will be seen shortly our marginalization algorithm is equivalent to the trellis BCJR algorithm. Here we simply report the closed-form representation of the general junction chain, and give the corresponding marginalization algorithm. We have the same underlying sample-space Ω as in Example 6.5. For compact representation, we continue to use variables, viewing each specific value of a variable as an event, i.e. a subset of Ω. Now for each $i = 0, \cdots, n-1$ we define a variable v_i taking value in $\{0, \cdots, q_i - 1\}$, as follows:

- $v_0 := u_0$
- For $i = 1, \cdots, m-1$:

$$v_i := \begin{cases} v_{i-1} & \text{if } u_i = 0 \\ i+1 & \text{if } u_i = 1 \end{cases}$$

  Note that here $q_i = i + 2$.

- For $i = m, \cdots, n-m+1$:

$$v_i := \begin{cases} \max(0, v_{i-1} - 1) & \text{if } u_i = 0 \\ m & \text{if } u_i = 1 \end{cases}$$

  with $q_i = m + 1$.

- For $i = n-m+2, \cdots, n-1$:

$$v_i := \begin{cases} \max(0, v_{i-1} - 2) & \text{if } u_i = 0 \\ n-i+1 & \text{if } u_i = 1 \end{cases}$$

  with $q_i = n - i + 2$.

Once the lifting has been done, the resulting junction tree is a chain similar to that of Figure 6.5 where atoms of the σ-fields are given by the events $\{v_i = j\}$; more specifically, the jth atom of $\mathcal{F}_i' \vee \mathcal{G}_i'$ is $\{\omega \in \Omega : v_i = j\}$. Then q_i, the number

of atoms of $\mathcal{F}_i' \vee \mathcal{G}_i'$ is the number of possible values for v_i as listed above. Also in this case $\mathrm{nz}(i, i+1)$, the number of nonzero entries in the matrix of joint measures of atoms of $\mathcal{F}_i' \vee \mathcal{G}_i'$ and $\mathcal{F}_{i+1}' \vee \mathcal{G}_{i+1}'$ is simply $2q_i$. The total arithmetic complexity of implementing Algorithm 3.1 on this chain to solve the original marginalization problem is $(9nm+6n+18m-7m^2-32)$, which should be compared to $3(n-m)2^{m+2}$ operations required by naive implementation of GDL.

Lastly, we give the explicit message update rules of Algorithm 3.1 on this chain. Note that the original form of Algorithm 3.1 will involve multiplying each term below by a fixed weight, which we have simplified throughout the expressions. Also for compactness, here we are defining $f_i(v_i)$ to equal the product $P(u_i)P(y_i^*|u_i, \cdots, u_{i-m})$, noting that both these functions are measurable given v_i: for a given value of v_i, the corresponding value for u_i is 1 if $v_i = q_i - 1$, and is 0 otherwise; also the corresponding value for the term $\vee_{j=0}^{m} u_{i-j}$ is 0 if $v_i = 0$, and is 1 otherwise.

- For $i = 1, \cdots, m-1$:

$$Y_{i-1,i}(v_i) = \begin{cases} f_{i-1}(v_i) Y_{i-2,i-1}(v_i) & \text{if } v_i < q_i - 1 \\ \sum_{j=0}^{q_{i-1}-1} f_{i-1}(j) Y_{i-2,i-1}(j) & \text{if } v_i = q_i - 1 \end{cases}$$

  and

$$Y_{i,i-1}(v_{i-1}) = f_i(q_i - 1) Y_{i+1,i}(q_i - 1) + f_i(v_{i-1}) Y_{i+1,i}(v_{i-1})$$

- For $i = m, \cdots, n-m+1$:

$$Y_{i-1,i}(v_i) = \begin{cases} \sum_{j=0}^{1} f_{i-1}(j) Y_{i-2,i-1}(j) & \text{if } v_i = 0 \\ f_{i-1}(v_i + 1) Y_{i-2,i-1}(v_i + 1) & \text{if } 0 < v_i < q_i - 1 \\ \sum_{j=0}^{q_{i-1}-1} f_{i-1}(j) Y_{i-2,i-1}(j) & \text{if } v_i = q_i - 1 \end{cases}$$

  and

$$Y_{i,i-1}(v_{i-1}) = f_i(q_i - 1) Y_{i+1,i}(q_i - 1) + f_i\big(\max(0, v_{i-1} - 1)\big) Y_{i+1,i}\big(\max(0, v_{i-1} - 1)\big)$$

- For $i = n-m+2, \cdots, n-1$:

$$Y_{i-1,i}(v_i) = \begin{cases} \sum_{j=0}^{2} f_{i-1}(j) Y_{i-2,i-1}(j) & \text{if } v_i = 0 \\ f_{i-1}(v_i + 2) Y_{i-2,i-1}(v_i + 2) & \text{if } 0 < v_i < q_i - 1 \\ \sum_{j=0}^{q_{i-1}-1} f_{i-1}(j) Y_{i-2,i-1}(j) & \text{if } v_i = q_i - 1 \end{cases}$$

and

$$Y_{i,i-1}(v_{i-1}) = f_i(q_i-1)Y_{i+1,i}(q_i-1)+f_i\big(\max(0,v_{i-1}-2)\big)Y_{i+1,i}\big(\max(0,v_{i-1}-2)\big)$$

Careful examination of these expressions reveals that the above algorithm is equivalent to the BCJR algorithm on the trellis tailored for this problem. Figure 6.9 shows the corresponding trellis for $n = 9$ and $m = 4$, and the connection with variables v_i above. It is evident that the original general version of GDL relying on variables is

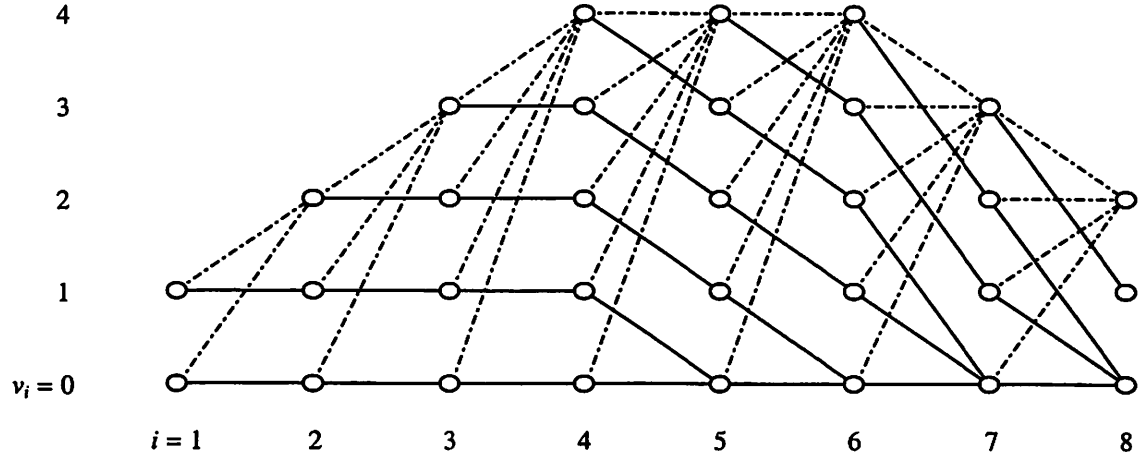


Figure 6.9: Trellis of the state machine with $n = 9$ and $m = 4$.
Solid and dashed lines correspond to 0 and 1 inputs respectively.

neither natural nor adequate in dealing with this problem. Although, as shown above, a description of the optimal marginalization algorithm in terms of some 'variables' exists, it requires a careful preprocessing phase to discover such variables. Of course in practice this inadequacy is addressed by invention of trellises, but that method is not a general method that is applicable to all classes of problems. Our lifting algorithm, on the other hand, is general and automatic in detecting the structures in any marginalization problem, and producing an efficient algorithm, without the need to discover hidden variables or to have any knowledge of trellises. Similarly, the eventual marginalization algorithm does not run on any trellis nor does it require complicated descriptions as shown in this example. □

Chapter 7

Summary and Discussion

We have developed a measure-theoretic version of the junction tree algorithm. We have generalized the notions of independence and junction trees at the level of σ–fields, and have produced algorithms to find or construct a junction tree on a given set of σ–fields. By taking advantage of structures at the atomic level of sample space Ω, our lifting algorithm is capable of automatically producing solutions less complex than the GDL-type algorithms. Although one can typically introduce new variables and redefine the local functions in a way that GDL will also come up with the same efficient algorithm as our method, this requires an intelligent processor to examine the data and determine a good way of introducing these variables. This process, then, remains more of an art than science. As we saw through example, our framework is capable of automating this process.

The cost of generating a junction tree with Algorithm 4.1 is exponential in the size of the problem, and so is the size of any complete representation of the sample space Ω. Once a junction tree has been constructed, however, the algorithm will only depend on the joint measure of the atoms of adjacent pairs of σ–fields on the tree. This means that an algorithm which was build by considering an Ω space with exponentially many elements, can be stored compactly and efficiently, and be used for all combinations of input functions. Therefore the cost of generating an efficient marginalization algorithm is amortized over many uses.

Using our framework, the tradeoff between the construction complexity of junc-

tion trees and the overall complexity of the marginalization algorithm can be made with an appropriate choice for the representation of the measurable spaces; at one extreme, one considers the complete sample space, taking advantage of all the possible structures, and at the other, one represents the sample space with independent variables (i.e. orthogonal directions), in which case our framework reduces to GDL, both in concept and in implementation

The validity of this theory for the *signed* measures is of enormous convenience; it allows for introduction of atoms of negative weight in order to create independencies. This greatly simplifies the task of lifting, which now can be done using standard techniques such as singular value decomposition. By contrast, the problem of finding a *positive* rank-one decomposition of a positive matrix (which would arise if one confined the problem to the positive measures functions) is a hard problem (see [7]). Meanwhile, the complexities and abstractions due to use of signed measure theory are transparent to the end-user: the eventual marginalization algorithm will only consist of additions and multiplications of the original data.

The measure-theoretic framework is the natural choice that is capable of discovering all the structure inherent in a problem. The extension of the GDL to this framework is quite natural and mathematically elegant. Together with the lifting algorithm, we feel that this approach can have strong practical as well as theoretical significance.

Part II

Kikuchi Approximation Method

Chapter 8

Introduction and Setup

8.1 Motivation

It was shown recently in [49] that there is a close connection between loopy belief propagation and certain approximations to the *variational free energy* in statistical physics. Specifically, as we will also discuss later in this document, the fixed points of the belief propagation algorithm were shown to coincide with the stationary points of *Bethe free energy* subject to consistency constraints. Here, Bethe free energy is a simple class of approximations to the variational free energy. This is while the minima of the exact form of the variational free energy correspond to the true marginals of the underlying product distribution.

The significance of this observation goes far beyond the mathematical connection between fundamental problems from two seemingly unrelated fields; more importantly, an entirely new perspective is suggested, through which to view the empirically-successful loopy BP algorithm, and to improve on existing approximate estimation methods.

It is well-known that the Bethe free energy is only a special case of a more general class of approximations called *Kikuchi free energy*, see [15]. Kikuchi free energy functionals are in general much more powerful in approximating the variational free energy, although they are also more complex. Similar to the loopy BP algorithm, a class of iterative message-passing algorithms can be introduced as in [49], which at-

tempt to find the constrained minima of the Kikuchi free energy. Using such message-passing algorithms is expected to result in approximations that are closer to the true marginals than are the ones given by belief propagation.

In the rest of this document we will explore a wide range of ideas related to the Kikuchi approximation method. In particular, we discuss necessary conditions for uniqueness of the minimizers of the Kikuchi free energy, introduce graphical representations for the problem, and define *minimal graphical representations*, which result in iterative solutions that are often significantly less complex than the algorithms discussed in [49], [50] and [23]. Furthermore, we will show that, for generic problems, Kikuchi approximation yields the exact marginals if and only if this minimal graphical representation of the Kikuchi problem is loop-free, where by the 'Kikuchi problem' we mean the problem of minimizing the Kikuchi free energy, subject to some consistency constraints. We will also address the more general problem of approximating the entropy of a product distribution in terms of the entropies of its marginals.

Other researchers have developed various techniques based on related ideas, each with specific advantages over traditional loopy belief propagation; we will briefly review concepts from some basic mean field methods in Appendix G. Yuille [54] derived a 'double-loop,' free-energy minimizing algorithm that is guaranteed to converge, unlike loopy belief propagation, see Appendix H. Welling and Teh [47] formulate an algorithm of gradient descent type, which is guaranteed to find a fixed point of Bethe free energy. Wainwright and Jordan [44] discuss convex relaxations of the variational principle, resulting in efficient algorithms which yield upper bounds to the partition function.

8.2 Setup and Notation

Recall from Section 1.1 that the marginalize a product distribution (MPD) problem 1.2 is defined as that of finding the partition function and/or the marginals of an R-decomposable Boltzmann distribution, $B(\mathbf{x}) := \frac{1}{Z}\prod_{r\in R}\alpha_r(\mathbf{x}_r)$ for the given

collection $\{\alpha_r(\mathbf{x}_r), r \in R\}$ of kernel functions.

The methods developed in the rest of this document to solve this problem are best described in the language of *partially ordered sets* or *posets*, see e.g. [40]. Specifically, the collection R of regions can be viewed as a poset with set inclusion as its partial ordering relation. This is because inclusion is reflexive ($\forall r \in R, r \subseteq r$), antisymmetric ($r \subseteq s$ and $s \subseteq r$ implies $r = s$), and transitive ($r \subseteq s$ and $s \subseteq t$ implies $r \subseteq t$). We write $r \subset t$ to denote strict inclusion. We say t *covers* u in R and write $u \prec t$, if $u, t \in R$, $u \subset t$ and $\nexists v \in R$ s.t. $u \subset v \subset t$.

Definition 8.1. Given a poset R, its *Hasse diagram* G_R is a directed acyclic graph (DAG)[1], whose vertices are the elements of R, and whose edges correspond to cover relations in R, i.e. an edge $(t \rightarrow u)$ exists in G_R iff $u \prec t$. □

It follows that for any two distinct nodes $r, s \in R$, we have $r \subset s$ iff there is a directed path from s to r in G_R.

Throughout this document we will need the following definitions. Let R be a poset of subsets of $[N]$ with the partial ordering of inclusion. For each subset $r \subseteq [N]$ we define:

Ancestors:	$\mathcal{A}(r) := \{s \in R \ : \ r \subset s\}$
Descendants	$\mathcal{D}(r) := \{s \in R \ : \ s \subset r\}$
Forebears (Up-set)	$\mathcal{F}(r) := \{s \in R \ : \ r \subseteq s\}$

Further for $r \in R$ we define

Parents	$\mathcal{P}(r) := \{s \in R \ : \ r \prec s\}$
Children	$\mathcal{C}(r) := \{s \in R \ : \ s \prec r\}$

[1] Traditionally the Hasse diagram is drawn as an 'undirected graph, with an implied upward direction' (see [40]). This is indeed equivalent to a DAG, which will be the view used in this dissertation.

Note that in each of these definitions, the collection of subsets being defined is comprised of regions, even though the argument r of $\mathcal{A}(r), \mathcal{D}(r)$ and $\mathcal{F}(r)$ need not be a region itself. For a collection S of subsets of $[N]$, we define $\mathcal{F}(S) := \bigcup_{s \in S} \mathcal{F}(s)$. Finally we define the *depth* of each region $r \in R$ as:

$$d(r) := \begin{cases} 0 & \text{if } r \text{ is maximal} \\ 1 + \max_{s \in \mathcal{P}(r)} d(s) & \text{otherwise} \end{cases}$$

Example 8.1. Let $R := \{\{1\}, \{4\}, \{1,2\}, \{1,3\}, \{1,2,3\}, \{1,2,4\}, \{1,2,5\}, \{1,3,4\}\}$ be a poset on $\{1,2,3,4,5\}$. The Hasse diagram corresponding to this poset is pictured in Figure 8.1 below.

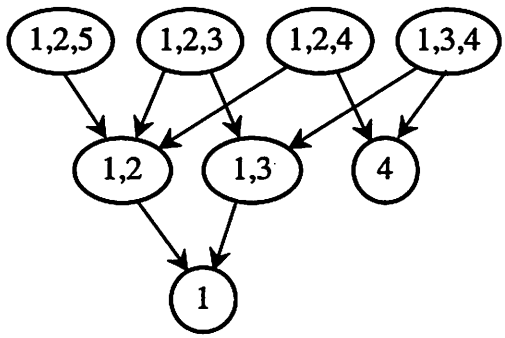


Figure 8.1: Hasse diagram of the poset of Example 8.1

Then, e.g. we have $\mathcal{A}(\{1\}) = \{\{1,2\}, \{1,3\}, \{1,2,3\}, \{1,2,4\}, \{1,2,5\}, \{1,3,4\}\}$; $\mathcal{P}(\{1\}) = \{\{1,2\}, \{1,3\}\}$; $\mathcal{F}(\{1,3\}) = \{\{1,3\}, \{1,2,3\}, \{1,3,4\}\}$; $\mathcal{D}(\{1,3\}) = \{\{1\}\}$; and $\mathcal{C}(\{4\}) = \emptyset$. Also, e.g. $\mathcal{F}(\{1,4\}) = \{\{1,2,4\}, \{1,3,4\}\}$ is defined, even though $\{1,4\} \notin R$. Finally, in this example regions $\{1,2,3\}, \{1,2,4\}, \{1,2,5\}$ and $\{1,3,4\}$ have depth 0, while $\{4\}, \{1,2\}$ and $\{1,3\}$ have depth 1, and $\{1\}$ has depth 2. $\square$

Chapter 9

Kikuchi Approximation Method

9.1 Connection with Statistical Physics

In the problem setup described in Sections 1.1 and 8.2, we can view x_i as the 'spin' of the particle at position i in a system of N particles. Let $b(\mathbf{x})$ denote a probability distribution on the configuration of spins, and consider a function $E(\mathbf{x})$ called the *energy function.* Suppose the energy function is R-decomposable, i.e. $E(\mathbf{x}) = \sum_{r \in R} E_r(\mathbf{x}_r)$ for certain functions $\{E_r(\mathbf{x}_r), r \in R\}$.

In statistical physics one defines (*Helmholtz*) *variational free energy* as the following functional of the distribution:

$$F(b(\mathbf{x})) := U(b(\mathbf{x})) - H(b(\mathbf{x})) \tag{9.1}$$

where $U := \sum_{\mathbf{x}} b(\mathbf{x}) E(\mathbf{x})$ is the average energy and $H := -\sum_{\mathbf{x}} b(\mathbf{x}) \log(b(\mathbf{x}))$ is the entropy of the system. We make the connection with the problem formulation of Section 8.2 by setting $E_r(\mathbf{x}_r) := -\log(\alpha_r(\mathbf{x}_r))$. We can then write

$$\begin{aligned} E(x) &= \sum_{r \in R} E_r(\mathbf{x}_r) \\ &= -\sum_{r \in R} \log(\alpha_r(\mathbf{x}_r)) \\ &= -\log(\frac{1}{Z} \prod_{r \in R} \alpha_r(\mathbf{x}_r)) - \log(Z) \\ &= -\log(B(\mathbf{x})) - \log(Z) \end{aligned}$$

where $B(\mathbf{x})$ is the Boltzmann distribution of (1.2). Then the variational free energy can be rewritten as follows:

$$F(b) = \sum_{\mathbf{x}} b(\mathbf{x})\big(-\log(B(\mathbf{x})) - \log(Z)\big) + \sum_{\mathbf{x}} b(\mathbf{x})\log(b(\mathbf{x})) \tag{9.2}$$

$$= \sum_{\mathbf{x}} b(\mathbf{x})\log(\frac{b(\mathbf{x})}{B(\mathbf{x})}) - \log(Z) \tag{9.3}$$

$$= \mathrm{KL}(b||B) - \log(Z) \tag{9.4}$$

where $\mathrm{KL}(b||B)$ is the Kullback-Leibler divergence between $b(\mathbf{x})$ and $B(\mathbf{x})$, see e.g. [9]. It is then clear that $F(b)$ is uniquely minimized when $b(\mathbf{x})$ equals the Boltzmann distribution $B(\mathbf{x})$ of (1.2), and we have

$$F_0 := \min_{b(\mathbf{x})} F(b(\mathbf{x})) = F(B(\mathbf{x})) = -\log(Z). \tag{9.5}$$

As mentioned in the introduction, equation (9.5) is of great interest in science and engineering. Physicists are interested in finding the *log-partition function* F_0, as a function of a temperature variable, which we have omitted here, since thermodynamical properties of physical systems can be derived from it. In estimation problems in engineering, one is interested in finding the marginals of the Boltzmann distribution $B(\mathbf{x})$. This is called the probabilistic inference problem. However, equation (9.5), viewed as an optimization problem, does not prescribe a practical way for computing these quantities, as it involves minimization over the exponentially large domain of distributions $b(\mathbf{x})$.

Given that the energy function is R-decomposable, to simplify the minimization problem (9.5) one may try to reformulate it in a way that is, loosely speaking, also R-decomposable. A natural way to do this is to try to represent the free energy as a functional of the R-marginals of the distribution $b(\mathbf{x})$.

Definition 9.1. We will call a collection $\{b_r(\mathbf{x}_r), r \in R\}$ of probability functions, which may or may not be the marginals of a single distribution, a collection of *R-pseudo-marginals.*

A collection of R-pseudo-marginals that are further the marginals of a probability distribution $b(\mathbf{x})$ are called the *R-marginals* of $b(\mathbf{x})$. □

Define Δ_R to be the family of the R-marginals of all probability distributions on $\mathbf{x}$, i.e. a collection $\{b_r(\mathbf{x}_r), r \in R\}$ belongs to Δ_R if and only if there exists a distribution $b(\mathbf{x})$ s.t. $\forall r \in R$, $b_r(\mathbf{x}_r) = \sum_{\mathbf{x}_{[N]\setminus r}} b(\mathbf{x})$. Then we can rewrite (9.5) as

$$\begin{aligned} F_0 &= \min_{\{b_r(\mathbf{x}_r)\}\in\Delta_R} F_R(\{b_r(\mathbf{x}_r)\}) \\ \{b_r^*(\mathbf{x}_r)\} &= \arg \min_{\{b_r(\mathbf{x}_r)\}\in\Delta_R} F_R(\{b_r(\mathbf{x}_r)\}) \end{aligned} \tag{9.6}$$

where

$$F_R(\{b_r\}) := \min_{b(\mathbf{x}):\{b_r\}\ R\text{-marginals of } b} F(b(\mathbf{x})).$$

Since $E(\mathbf{x})$ is R-decomposable, the average energy decomposes as

$$U(b(\mathbf{x})) = \sum_{r\in R}\sum_{\mathbf{x}_r} b_r(\mathbf{x}_r)E_r(\mathbf{x}_r) \tag{9.7}$$

where $b_r(\mathbf{x}_r)$'s are the marginals of distribution $b(\mathbf{x})$. In general however, the entropy term in the free energy (9.1) cannot be decomposed in terms of the R-marginals of $b(\mathbf{x})$. The key component of the Kikuchi approximation method is to use an approximation of the form

$$H(b(\mathbf{x})) \simeq \sum_{r\in R} k_r\, H_r(b_r(\mathbf{x}_r)) \tag{9.8}$$

where $H_r(b_r(\mathbf{x}_r)) := -\sum_{\mathbf{x}_r} b_r(\mathbf{x}_r)\log(b_r(\mathbf{x}_r))$ is the regional entropy associated with a region $r \in R$, and k_r's are suitable constants to be determined.

A suitable choice for the factors $\{k_r\}$ is achieved using the Möbius inversion formula, see Appendix E. In particular, we can recursively define a collection of factors $\{c_r, r \in R\}$ using the following equations:

$$c_r = 1 - \sum_{s\in\mathcal{A}(r)} c_s \tag{9.9}$$

where $\mathcal{A}(r)$ is the set of ancestors of r, as defined in Section 8.2. Following [49] we call the c_r's defined in this manner the *overcounting factors*. As it turns out, c_r's are the natural choice for the constants $\{k_r\}$ in (9.8), as we show in Proposition 9.1:

Proposition 9.1. *The only choice of factors* $\{k_r\}$ *which can result in exactness of* (9.8) *for all R-decomposable Boltzmann distributions ,- i.e. distributions* $b(\mathbf{x}) := \frac{1}{Z}\prod_{r\in R}\alpha_r(\mathbf{x}_r)$ *for all choices of* $\{\alpha_r, r \in R\}$ *- is the overcounting factors* $\{c_r\}$.

We will prove this proposition in Section 10. In fact the original choice of $\{k_r\}$ in the Kikuchi approximation method [15] was also $\{k_r\} = \{c_r\}$. It will also be shown that this exactness happens if and only the collection R of regions is 'loop-free' in an appropriate sense, which will be defined in Section 10.1.

The Kikuchi approximation method, which will be defined more formally in Section 9.2, proposes to solve a constrained minimization problem of the following form (cf. equation (9.6)):

$$\{B_r(\mathbf{x}_r)\} \simeq \{b_r^*(\mathbf{x}_r)\} := \arg \min_{\{b_r(\mathbf{x}_r)\}\in\Delta_R^K} F_R^K(\{b_r(\mathbf{x}_r)\}) \tag{9.10}$$

Here $F_R^K(\{b_r\})$, known as the *Kikuchi free energy*, see e.g. [15], is defined as (cf. equation (35) in [49])

$$F_R^K(\{b_r(\mathbf{x}_r)\}) := \sum_{r\in R}\sum_{\mathbf{x}_r} b_r(\mathbf{x}_r)E_r(\mathbf{x}_r) + \sum_{r\in R}\sum_{\mathbf{x}_r} c_r\, b_r(\mathbf{x}_r)\log(b_r(\mathbf{x}_r)) \tag{9.11}$$

and Δ_R^K is a set of constraints to enforce consistency between the b_r's, defined as

$$\Delta_R^K := \Big\{\{b_r(\mathbf{x}_r),\, r\in R\} \;:\; \forall t,u\in R \text{ s.t. } t\subset u, \sum_{\mathbf{x}_{u\setminus t}} b_u(\mathbf{x}_u) = b_t(\mathbf{x}_t) \text{ and } \forall u\in R,\ \sum_{\mathbf{x}_u} b_u(\mathbf{x}_u) = 1\Big\} \tag{9.12}$$

Note that in general the constraints of Δ_R^K are not enough to guarantee that every collection of pseudo-marginals $\{b_r,\, r\in R\} \in \Delta_R^K$ is in fact the collection of the marginals of a single distribution function $b(\mathbf{x})$; a collection may very well satisfy all the consistency constraints of (9.12) and not be the marginals of any distribution.

In Chapter 10 we discuss conditions on R that guarantee that the free energy $F(b)$ can be viewed as a functional of the marginals of $b(\mathbf{x})$, i.e. $\{b_r, r\in R\}$, and, as such a functional, equals the Kikuchi functional F_R^K. Further we discuss conditions on R under which the constraint set Δ_R^K equals the family of R-marginals Δ_R.

9.2 Kikuchi Approximation Method

In this section we formulate the Kikuchi approximation method for solving the MPD problem posed in Section 1.1. We will further describe conditions on the collection of regions R, which are expected to improve the quality of the approximations.

Let R_0 be a collection of regions, and $\{\alpha_r^0(\mathbf{x}_r),\ r \in R_0\}$ be a collection of kernel functions. We are interested in solving the MPD problem 1.2 posed in Section 8.2 for the collection R_0 of regions and the corresponding kernels $\{\alpha_r^0\}$.

Let R be another collection of regions obtained from R_0 in such a way that $\forall r' \in R_0$, $\exists r \in R$ s.t. $r' \subseteq r$. Then one can always form[1] a collection of R-kernels $\{\alpha_r(\mathbf{x}_r),\ r \in R\}$ so that $-\sum_{r\in R}\log(\alpha_r(\mathbf{x}_r)) = -\sum_{r\in R_0}\log(\alpha_r^0(\mathbf{x}_r)) =: E(\mathbf{x})$.

Now for each $r \in R$, define $\beta_r(\mathbf{x}_r) := \prod_{s\subseteq r}\alpha_s(\mathbf{x}_s)$. Then the Boltzmann distribution of equation (1.2) takes the following product forms:

$$B(\mathbf{x}) = \frac{\prod_{r'\in R_0}\alpha_{r'}^0(\mathbf{x}_{r'})}{Z} = \frac{\prod_{r\in R}\alpha_r(\mathbf{x}_r)}{Z} = \frac{\prod_{r\in R}\beta_r(\mathbf{x}_r)^{c_r}}{Z} \tag{9.13}$$

where the last equality follows from the fact that, by (9.9), $\sum_{r\in\mathcal{F}(s)} c_r = 1$ for all $s \in R$.

Using approximations (9.11) and (9.12) we are now interested in solving the following:

Problem 9.1 (Kikuchi Approximation).

$$\begin{aligned} -\log(Z) \simeq F^* &:= \min_{\{b_r(\mathbf{x}_r)\}\in\Delta_R^K} F_R^K(\{b_r(\mathbf{x}_r)\}) \\ \text{and}\quad \{B_r(\mathbf{x}_r)\} \simeq \{b_r^*(\mathbf{x}_r)\} &:= \arg\min_{\{b_r(\mathbf{x}_r)\}\in\Delta_R^K} F_R^K(\{b_r(\mathbf{x}_r)\}) \end{aligned} \tag{9.14}$$

□

[1]Note however that the way this assignment is done can impact the quality of the approximations to (9.6) provided by (9.14).

Note now that by equation (9.6), if $F(b(\mathbf{x})) = F_R^K(\{b_r(\mathbf{x}_r)\})$ for all $b(\mathbf{x})$, and $\Delta_R = \Delta_R^K$, then the minimizer collection $\{b_r^*(\mathbf{x}_r)\}$ of (9.14) would correspond exactly to the collection of the marginals of the product function $B(\mathbf{x})$ of equation (9.13); hence, if the Kikuchi approximate free energy $F_R^K(\{b_r\})$ is close to $F(b)$, and local consistency constraint set Δ_R^K is also close to Δ_R, the minimizers $\{b_r^*\}$ of equation (9.14) are expected to be close approximations to these marginals.

Our focus in the rest of this dissertation shifts to the above problem and to the relation between the solution to this problem and the original one in Section 8.2. An important question which we address in detail is when the b_r^*'s are equal to the marginals B_r of the Boltzmann distribution. We also address in detail in Section 10 message-passing algorithms on graphs which solve equation (9.14) which are more efficient than the ones known to date, such as the *generalized belief propagation* [50] and *poset belief propagation* [23].

The collection R of regions effectively specifies both the Kikuchi approximation (9.11), and the constraint set (9.12). It is also evident that (9.14) as an approximation method can be applied for any given F_R^K and Δ_R^K; better choices of R simply result in better approximations. Therefore we can define the Kikuchi approximation method as the general class of constrained minimization problems given by (9.14), which are parameterized by the poset[2] R of regions, and local kernel functions $\alpha_r(\mathbf{x}_r)$ for each $r \in R$.

It remains to specify which choices of R yield good approximations of the marginals. In the remainder of this dissertation we only consider collections of regions R that have the same maximal regions as R_0. Expansion of the maximal regions corresponds to 'clustering' methods, as discussed in [33]. The techniques developed here to derive low complexity message-passing algorithms to solve the Kikuchi approximation problem can also be applied after clustering.

[2]Note that although 'inclusion' is certainly the most natural partial ordering for R, the problem is well-defined for any arbitrary partial ordering.

It certainly seems that minimization with more local consistency constraints on $\{b_r(\mathbf{x}_r)\}$ should result in better approximations, since the true marginals would satisfy all such constraints. At the same time, the entropy approximations of the type given in equation (9.8) are also expected to improve if more regions are included. Therefore one might conclude that for a given collection of maximal regions of R_0, augmenting them by introducing additional subregions to form R, – where the α_r's corresponding to the augmented subregions are taken to be 1 – should improve the approximation (at the expense of increasing the complexity of the underlying minimization).

Let G be a labelled graph whose vertices are identified with subsets of $[N]$. We define the following *connectivity conditions* on G:

$$\forall i \in [N], \text{the subgraph of } G \text{ consisting of the regions in } \mathcal{F}(\{i\}) \text{ is connected.} \quad (\mathbf{A1})$$

Generalizing this, we can devise condition $(\mathbf{A}n)$ on G, for each $n \in \{1, \cdots, N\}$ as follows:

$$\forall s \subset [N],\ |s| \leq n, \quad \text{the subgraph of } G \text{ on regions in } \mathcal{F}(s) \text{ is connected.} \qquad (\mathbf{A}n)$$

We say a poset R has property $(\mathbf{A}n)$ iff its Hasse diagram G_R satisfies condition $(\mathbf{A}n)$. Note that in the context of Kikuchi problem (9.14), property $(\mathbf{A}n)$ guarantees that the beliefs at all regions will be consistent at the level of any subset $\mathbf{x}_r$ of the variables of cardinality up to n. It is therefore natural to require that R satisfies at least condition $(\mathbf{A}1)$. We call a poset R satisfying $(\mathbf{A}n)$ for all n, a *totally connected* poset.

Inspired by [3], one might insist that acceptable approximations of the entropy term (9.8) are those in which each variable x_i appears the same number of times on the two sides of the equality sign, i.e.

$$\sum_{r \in \mathcal{F}(\{i\})} c_r = 1 \qquad \text{for each } i = 0, \cdots, N-1 \qquad (\mathbf{B1})$$

We can extend this condition also, as follows:

$$\sum_{r \in \mathcal{F}(s)} c_r = 1 \qquad \text{for each } s \subset [N],\ |s| \leq n \text{ s.t. } \mathcal{F}(s) \neq \emptyset \qquad (\mathbf{B}n)$$

Conditions ($\mathbf{B}n$) are called the *balance condtions*, and we call a poset R satisfying ($\mathbf{B}n$) for all n, a *totally balanced* poset.

These conditions are expected to give progressively better approximate solutions, although they will not in general guarantee an exact solution.

The original cluster variation method of Kikuchi as defined in [24] and [49] in effect chooses R to be the smallest collection of regions including R_0 which is closed under non-empty intersection of regions. The following proposition shows that the choice of R made in the cluster variation method is expected to give a reasonable Kikuchi approximation.

Proposition 9.2. *Any collection of regions R which is closed under non-empty intersection of regions is totally connected and totally balanced.*

Proof. Note first that if $u, v \in R$ and $u \subset v$, then there is a directed path from v to u in G_R, where all the nodes in the path contain u. Let $t \neq \emptyset$ be any subset of $[N]$, and let $r, s \in \mathcal{F}(t)$ be any two regions containing t. Then $r \cap s$ must lie in R, since R is closed under non-empty intersections. Therefore r and s each are connected in G_R to $r \cap s$, where all the vertices on the paths from r to $r \cap s$ and from s to $r \cap s$ contain $r \cap s$, which in turn contains t. This proves that R is totally connected.

Now let $r \subseteq [N]$ be a subset such that $\mathcal{F}(r) \neq \emptyset$. If $r \in R$, then by definition of the overcounting factors $\sum_{t \in \mathcal{F}(r)} c_t = 1$ and we are done. Suppose then that $r \notin R$. We will show that there is a unique minimal $s \in \mathcal{F}(r)$, so that $\mathcal{F}(r) = \mathcal{F}(s)$. If not, then there must be at least two minimal regions t_1 and t_2 in $\mathcal{F}(r)$, with $t_1 \not\subseteq t_2$ and $t_2 \not\subseteq t_1$. Then $t_1 \cap t_2$ is a region, strictly smaller than both t_1 and t_2, which lies in $\mathcal{F}(r)$ since it contains r. This would contradict t_1 and t_2 each being minimal

in $\mathcal{F}(r)$. Therefore there exists an $s \in R$ such that $\mathcal{F}(s) = \mathcal{F}(r)$, and therefore $\sum_{t \in \mathcal{F}(r)} c_t = \sum_{t \in \mathcal{F}(s)} c_t = 1$. This proves that R is totally balanced. □

The special case when the Hasse diagram G_R has depth 2, i.e. there are no distinct $r, s, t \in R$ such that $r \subset s \subset t$, is called the *Bethe case* in this document. In this case G_R can be thought of as a hypergraph in which the maximal regions of R are the vertices and the minimal regions are the hyperedges. If we insist, as assumed in [49], that the maximal regions be pairs $\{i, j\}$ of indices for $i, j \in [N]$, and that the minimal regions be all the singletons $\{i\}$ for $i \in [N]$, then we will in fact have a poset R of depth 2 which is closed under intersection; this is what was called the Bethe case in [49]. Our notion of *Bethe case* is more general than that of [49], since no restriction on the size of the regions is necessary, and we allow for R not to be closed under intersection.

On the other hand, [3] considers only the case when the aforementioned 'hypergraph' view of G_R is a graph, i.e. the minimal elements of R are covered by at most two regions, so the hyperedges are in fact edges. It can be immediately verified that he 'junction graph' condition given in [3] is simply the intersection of conditions (A1) and (B1) above. It can also be shown that the 'junction graph' condition of [3] does *not* imply either (A2) or (B2).

We now give an example to illustrate some of the notions defined in this section.

Example 9.1. Consider the $(16, 8)$ linear code represented by the bipartite graph of Figure 9.1, where the top nodes correspond to parity checks, and the bottom nodes correspond to symbol bits. This graph can be interpreted as the Hasse diagram of a two-level poset, where the regions associated with the 'bit-nodes' are $\{1\}, \{2\}, \cdots, \{16\}$ respectively, and the region associated with each 'check-node' is the subset of $\{1, \cdots, 16\}$ corresponding to the bits that constitute that parity check. This is an example of the Bethe case, where the regions corresponding to the check-

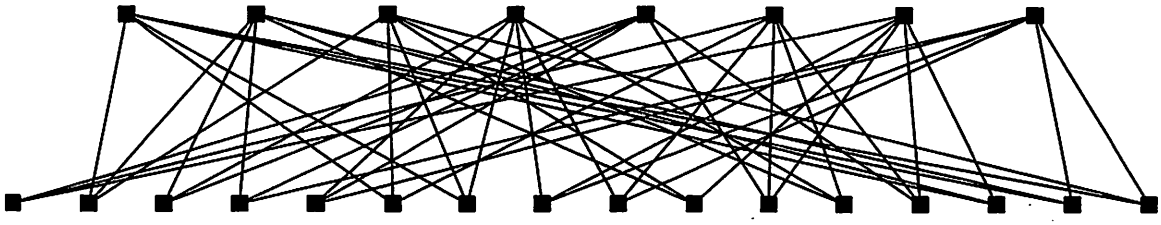


Figure 9.1: Tanner graph of a linear code

nodes are maximal and those corresponding to the bit-nodes are minimal. The overcounting factors corresponding to the check-nodes are equal to 1, while those corresponding to the bit-nodes equal "one minus the number of check-nodes connected to that bit-node". In this case each bit-node is connected to three check-nodes, so that the overcounting factors for all bit-nodes equal $1 - 3 = -2$. In this case, the GBP algorithm we discuss in Section 11 will reduce to the original Gallager-Tanner decoding algorithm for LDPC codes, see [13], [41].

This poset has property (**A1**), but not (**A2**): note for example that the regions corresponding to the first and third check-nodes are $\{2, 6, 7, 14, 15, 16\}$ and $\{2, 6, 7, 10, 12, 16\}$ respectively, both containing $\{2, 6\}$, but they are not connected through regions that contain $\{2, 6\}$.

Also, this poset satisfies (**B1**), but not (**B2**): for $s := \{2, 6\}$, $\mathcal{F}(s)$ is precisely the first and third check-node regions. Then $\sum_{r \in \mathcal{F}(s)} c_r = 1 + 1 = 2 \neq 1$.

On the other hand, one can throw in all the intersections of the check-node regions to create the poset whose Hasse diagram is shown in Figure 9.2.

Here the nodes in the middle row correspond to the intersections of the check-node regions, in the first row, to which they are connected; e.g. the second node in the middle row corresponds to region $\{2, 6, 7, 16\}$, which is the intersection of the first and third check-node regions.

It is easy to verify that this poset is totally connected and totally balanced. □

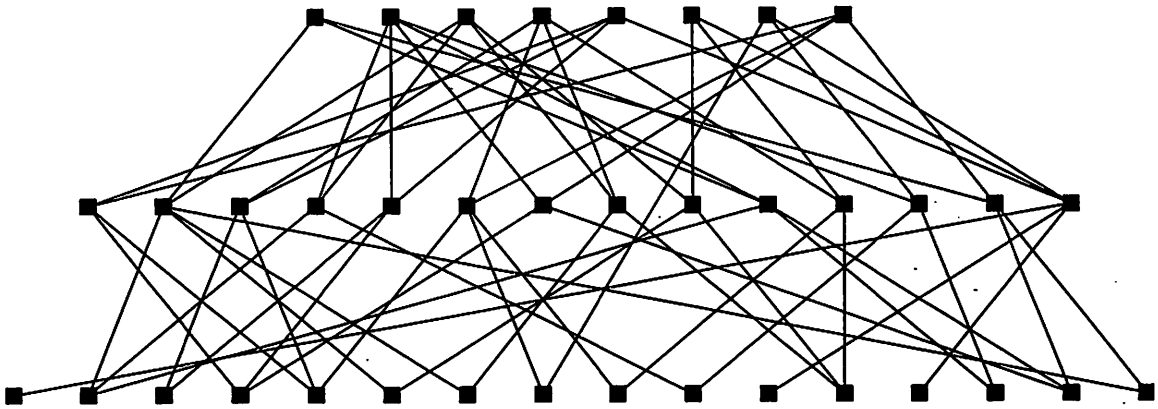


Figure 9.2: Alternative poset of linear code of Example 9.1

9.3 Lagrange Multipliers and Iterative Solutions

Lagrange's method can be used to solve the constrained minimization problem (9.14). We form the Lagrangian:

$$\begin{aligned}\mathcal{L} := \sum_{r\in R}\sum_{\mathbf{x}_r}(-b_r(\mathbf{x}_r)\log(\alpha_r(\mathbf{x}_r)) + c_r\, b_r(\mathbf{x}_r)\log(b_r(\mathbf{x}_r))) \\ + \sum_{r\in R}\sum_{t\prec r}\sum_{\mathbf{x}_t}\lambda_{rt}(\mathbf{x}_t)\big(b_t(\mathbf{x}_t) - \sum_{\mathbf{x}_{r\setminus t}} b_r(\mathbf{x}_r)\big) \\ + \sum_{r\in R}\kappa_r\big(\sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) - 1\big) \quad (9.15)\end{aligned}$$

where coefficients $\lambda_{rt}(s_t)$ enforce consistency constraints, and coefficients κ_r enforce normalization constraints, and as before $t \prec r$ means that r covers t. Note that since the edge-constraints of G_R are a sufficient representation of Δ_R^K as discussed before, we need only define λ_{rt} for pairs $r, t \in R$ with $t \prec r$, i.e. along the edges of G_R.

Setting partial derivative $\partial\mathcal{L}/\partial b_r(\mathbf{x}_r) = 0$ for each $r \in R$ gives an equation for $b_r(\mathbf{x}_r)$ in terms of λ_{ur}'s and λ_{rt}'s. The consistency constraints give update rules for each λ_{rt} in terms of other λ multipliers. Once a set of messages m_{rt} (from r to t, for each edge $(r \to t)$ of G_R) has been defined in terms of the Lagrange multipliers λ_{rt}'s, these update rules define an iterative algorithm whose fixed points are the stationary points of the given constrained minimization problem.

In Section 11 we will give detailed derivation for a nice such algorithm called the

'generalized belief propagation' (GBP) algorithm, see also [49], and we will also see that the belief propagation algorithm of [33] is the restriction of the above algorithm in the Bethe case. It should be noted that the algorithm we derive in Section 11, although called GBP, can be considerably less complex than the one called GBP in [49]. This is because of certain systematic complexity-reducing transformations we carry out in Section 10, which constitute the major practical contribution of this work.

9.4 Convexity Conditions

In this section we describe our results regarding the convexity of the optimization problem (9.14), which we first reported in [30].

Kikuchi free energy (9.11) constrained on $\{b_r\} \in \Delta_R^K$ is bounded below and hence the constrained minimization problem (9.14) always has a global minimum. Therefore, as discussed in Section 9.3, the message passing algorithms derived from Lagrangian (9.15) always possess at least one fixed point (see [54] for an algorithm that is guaranteed to find a minimum of F_R^K).

The following result gives sufficient conditions on R for the problem (9.14) to have precisely one minimum:

Theorem 9.3. *Kikuchi free energy functional* (9.11) *is strictly convex on* Δ_R^K *(and hence the constrained minimization problem has a unique solution) if the overcounting factors* $c_r, r \in R$ *satisfy:*

$$\forall S \subseteq R, \quad \sum_{s \in \mathcal{F}(S)} c_s \geq 0 \tag{9.16}$$

where, as defined in Section 8.2, $\mathcal{F}(S) := \cup_{s \in S} \mathcal{F}(s) = \{r \in R : \exists\, s \in S \text{ s.t. } r \subseteq s\}$ *is the set of forebears of* S.

Proof. Note that Kikuchi approximate free energy, as a functional of the pseudo-marginals $\{b_r(\mathbf{x}_r)\} \in \Delta_R^K$ consists of an energy term – which is linear, – and a linear

combination of entropy terms, with both positive and negative coefficients. We will show that if the hypothesis of the theorem holds, there is a matching between the negative and the positive terms such that the overall entropy term will be a positive linear combination of KL divergence terms which are strictly convex, see e.g. [9]. We will prove the existence of such matching using results from the bipartite graph theory.

Form a bipartite graph $G(V^+, V^-, E)$ with vertex sets V^+ and V^- and the edge set E as follows:

- For each $r \in R$ with $c_r < 0$, create $|c_r|$ nodes $\{v_r^1, \cdots, v_r^{|c_r|}\}$ in V^-.
- For each $s \in R$ with $c_s > 0$, create c_s nodes $\{u_s^1, \cdots, u_s^{c_s}\}$ in V^+.
- To form the edge set E, connect each $v_i^r \in V^-$ to each $u_j^s \in V^+$ iff $r \subset s$.

For a subset $S \subseteq V^-$, denote by $N(S)$ the subset of nodes in V^+ that are connected to a node in S. Then graph G has the following property:

$$\forall S \subseteq V^-, \ \ |S| \leq |N(S)| \tag{9.17}$$

To see this, let $S = \{v_s^i \ : \ (s,i) \in \mathcal{I}\}$ where the index set $\mathcal{I}$ consists of some pairs of the form (s,i) with $c_s < 0$ and $0 < i \leq |c_s|$. Now create another index set $\overline{\mathcal{I}}$ as $\overline{\mathcal{I}} := \{(s,j) \ : \ (s,i) \in \mathcal{I} \ \text{ for some } i \ , \ 0 < j \leq |c_s|\}$, and let $\overline{S} := \{v_s^i \ : \ (s,i) \in \overline{\mathcal{I}}\}$. Then clearly $S \subseteq \overline{S}$ and hence $|S| \leq |\overline{S}|$, but notice that $N(S) = N(\overline{S})$. Also note that $|\overline{S}| = -\sum_{t \in T} c_t$, where $T := \{t \in R \ : \ (t,1) \in \overline{\mathcal{I}}\}$. Further,

$$\begin{aligned}
\sum_{t \in \mathcal{A}(T)} c_t &= \sum_{t \in \mathcal{A}(T); c_t > 0} c_t + \sum_{t \in \mathcal{A}(T); c_t < 0} c_t \\
&= |N(\overline{S})| + \sum_{t \in \mathcal{A}(T); c_t < 0} c_t \\
&\leq |N(\overline{S})|
\end{aligned}$$

where the second equality follows from the definitions of $|N(\overline{S})|$ and $\mathcal{A}(T)$. But by the hypothesis of the theorem, $-\sum_{t \in T} c_t \leq \sum_{t \in \mathcal{A}(T)} c_t$. Putting these all together, we get $|S| \leq |\overline{S}| = -\sum_{t \in T} c_t \leq \sum_{t \in \mathcal{A}(T)} c_t \leq |N(\overline{S})| = |N(S)|$ as claimed.

Then the bipartite graph satisfies the hypothesis of Hall's Matching Theorem (see [14]), and hence there is a matching on G that saturates every vertex of V^-. In other words, there is matching $M = \{(v_r^i, u_s^j)\}$ such that every $v_r^i \in V^-$ is uniquely matched with a $u_s^j \in V^+$. Denote by U the subset of vertices in V^+ that are left unmatched.

We now rewrite the entropy term of the Kikuchi free energy, i.e. the second summation in (9.11), using the matching M. For each $\{b_r\} \in \Delta_R^K$:

$$\begin{aligned}
\sum_{r\in R} c_r \sum_{\mathbf{x}_r} b_r \log(b_r) &= \sum_{r:c_r<0} c_r \sum_{\mathbf{x}_r} b_r \log(b_r) + \sum_{s:c_s>0} c_s \sum_{\mathbf{x}_s} b_s \log(b_s) \\
&= -\sum_{r:c_r<0} \sum_{i=1}^{-c_r} \sum_{\mathbf{x}_r} b_r \log(b_r) + \sum_{s:c_s>0} \sum_{j=1}^{c_s} \sum_{\mathbf{x}_s} b_s \log(b_s) \\
&= -\sum_{v_r^i \in V^-} \sum_{\mathbf{x}_r} b_r \log(b_r) + \sum_{u_s^j \in V^+} \sum_{\mathbf{x}_s} b_s \log(b_s) \\
&= \sum_{(v_r^i, u_s^j)\in M} \Big(\sum_{\mathbf{x}_s} b_s \log(b_s) - \sum_{\mathbf{x}_r} b_r \log(b_r)\Big) + \sum_{u_s^j \in U} \sum_{\mathbf{x}_s} b_s \log(b_s) \\
&= \sum_{(v_r^i, u_s^j)\in M} \sum_{\mathbf{x}_s} b_s \log(\frac{b_s}{b_r}) + \sum_{u_s^j \in U} \sum_{\mathbf{x}_s} b_s \log(b_s) \qquad (9.18)
\end{aligned}$$

Notice that for each $(v_r^i, u_s^j) \in M$, by definition of the bipartite graph G, we have $r \subset s$. Further we have taken $\{b_r\} \in \Delta_R^K$, and so that $\sum_{\mathbf{x}_{s\setminus r}} b_s(\mathbf{x}_s) = b_r(\mathbf{x}_r)$ which implies the last equality.

Now note that the first term in (9.18) is a sum of KL-divergences[3], which are strictly convex as functions of their arguments, and the second term is a sum of negative entropy functions which are also strictly convex, see e.g. [9]. On the other hand, as mentioned earlier, the average energy term of the Kikuchi free energy, i.e. the first summation in (9.11), is linear in $\{b_r\}$. Since, constrained by Δ_R^K, the Kikuchi free energy is in effect a functional only of the pseudo-marginals associated to the *maximal* regions in R, and since each maximal region contributes such a KL-divergence term in (9.18), the Kikuchi functional as a whole is also strictly convex. □

Corollary 9.4. (cf. Theorem 3 in [3]) *In the Bethe case, the constrained minimization*

[3]To be precise, each term differs from a true KL-divergence by a constant.

problem (9.14) *has a unique solution if the graphical representation G_R of R has at most one loop.*

Proof. Let $S \subseteq R$ be a subset of regions, and consider the sum $A_S := \sum_{r \in \mathcal{F}(S)} c_r$. Note that in the Bethe case, $c_r = 1 - (\#$ of parents of $r)$ for all $r \in R$. This means that for each region $r \in R$, the contribution of c_r to the sum A_S can be broken up as a contribution of $(+1)$ for the vertex r, and a contribution of (-1) for each edge of the Hasse graph ending in r. Therefore, A_S is precisely equal to the number of vertices minus the number of edges of the Hasse graph of $\mathcal{F}(S)$, which is a subgraph of G_R. Therefore the sum is nonnegative iff this subgraph has at most one loop.

By Theorem 9.3, a sufficient condition for uniqueness of solution to the constrained minimization problem (9.14) is that the sum A_S above be nonnegative for all subsets $S \subseteq R$. In particular, choosing $S = R$ implies, by above, that G_R has at most one loop. On the other hand, if G_R has at most one loop, then any of its subgraphs will have no more than one loop, and by the above argument A_S will be nonnegative for each $S \subseteq R$. It follows that the nonnegativity of A_S for all $S \subseteq R$ is equivalent to the statement that G_R have at most one loop. Therefore the sufficient condition (9.16) for the uniqueness of solution is that G_R have no more than one loop. □

Once we define a suitable notion of graphical representation for a general collection of regions in the next section, we will generalize the result of Corollary 9.4.

Chapter 10

Graphical Representations of the Kikuchi Problem

In this section we define the notion of graphical representations, for a Kikuchi approximation problem. The algorithms of the type discussed in Section 9.3 can then be viewed as message-passing algorithms along the edges of such graphs. We will discuss this in detail in Section 11.

We will further introduce *minimal graphical representations* for a given collection R of regions, which are graphical representations with the fewest number of edges. Our motivation for introduction of such minimal graphs is two-fold.

First, note that the results of Section 9.4 refer to the uniqueness of solution of the constrained minimization problem (9.14). However, one is further interested in the conditions under which these solutions are the exact marginals of the product distribution (9.13). As we will show in this section, the exactness of approximations obtained using (9.14) corresponds directly to non-existence of loops in the minimal graphs. In fact, we will show that in the loop-free case, this graph is a junction tree and the message-passing algorithms of type discussed in Section 9.3 correspond to a variation of junction tree algorithm.

Second, as we will discuss in detail in Section 11, the message-passing algorithms of the type mentioned in Section 9.3 on minimal graphs will be the most compact among all graphical representations of the same problem, and can result in algorithms

that are significantly less complex than such algorithms as the GBP of [50] and the poset-BP of [23].

Let G be a directed acyclic graph with vertex set $V(G)$ and edge set $\mathcal{E}(G)$. Parallel to our definitions in Section 8.2, for each vertex $r \in V(G)$ define:

$$\begin{aligned}
\text{Ancestors:} \quad & \mathcal{A}_G(r) := \{s \in V \ : \ \exists \text{ a directed path from } s \text{ to } r\} \\
\text{Descendants} \quad & \mathcal{D}_G(r) := \{s \in V \ : \ r \in \mathcal{A}_G(s)\} \\
\text{Parents} \quad & \mathcal{P}_G(r) := \{s \in V \ : \ (s \to r) \in \mathcal{E}(G)\} \\
\text{Children} \quad & \mathcal{C}_G(r) := \{s \in V \ : \ (r \to s) \in \mathcal{E}(G)\} \\
\text{Forebears} \quad & \mathcal{F}_G(r) := \{r\} \cup \mathcal{A}_G(r)
\end{aligned}$$

As in Section 8.2, for a subset $S \subseteq V(G)$ we define $\mathcal{F}_G(S) := \bigcup_{s \in S} \mathcal{F}_G(s)$.

Also define *depth* of each vertex $r \in V(G)$ as:

$$d_G(r) := \begin{cases} 0 & \text{if } \mathcal{P}_G(r) = \emptyset \\ 1 + \max_{s \in \mathcal{P}_G(r)} d_G(s) & \text{otherwise} \end{cases}$$

Similarly we define the depth of each edge $(t \to u)$ of G, as the depth of the child vertex u:

$$d_G(t \to u) := d_G(u)$$

Note that given a poset R of regions, the above definitions for the Hasse diagram G_R are consistent with the corresponding definitions for the poset from Section 8.2, i.e. for all $r \in V(G_R)$, $\mathcal{A}_{G_R}(r) = \mathcal{A}(r)$ and so on.

Back to the problem at hand, let R be a collection of regions as before, and let G be a directed acyclic graph whose nodes correspond to the regions $r \in R$. We will further assume that an edge $(s \to t)$ exists in G only if $t \subset s$.

Definition 10.1. The *edge-constraint* for an edge $(s \to t)$ of G is defined as the

following functional of the pseudo-marginals $\{b_r\,,\, r \in R\}$:

$$\mathrm{EC}_{(s\to t)}(\{b_r\,,\, r\in R\}) := \sum_{\mathbf{x}_{s\setminus t}} b_s(\mathbf{x}_s) - b_t(\mathbf{x}_t) \tag{10.1}$$

When the arguments are clear from the context, we abbreviate this as $\mathrm{EC}_{(s\to t)}$. □

Definition 10.2. We call G a *graphical representation* of Δ_R^K if Δ_R^K can be represented using the edge-constraints of G, i.e.

$$\Delta_R^K = \Big\{\, \{b_r(\mathbf{x}_r),\, r\in R\} \;:\; \forall (s\to t)\in \mathcal{E}(G),\ \mathrm{EC}_{(s\to t)} = 0 \\ \text{and } \forall r\in R,\ \sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) = 1\Big\} \tag{10.2}$$

□

As mentioned in the previous sections, a poset R is most naturally represented by its Hasse diagram G_R; Hasse diagram uses the transitivity of partial ordering to represent a poset in the most compact form. Note that our local consistency constraints also have the transitivity property:
If $(r\to s), (s\to t)$ and $(r\to t)$ are edges in graph G, then

$$\big(\mathrm{EC}_{(r\to s)} = 0\big) \text{ and } \big(\mathrm{EC}_{(s\to t)} = 0\big) \implies \big(\mathrm{EC}_{(r\to t)} = 0\big)$$

Therefore the last edge (between 'grandfather' and 'grandchild') is redundant. This is why the Hasse diagram G_R is a graphical representation of Δ_R^K.

On the other hand, local consistency relations satisfy a property other than transitivity which can be used to further reduce the representation of Δ_R^K:

Suppose $(r\to s), (r\to u), (s\to t)$ and $(u\to t)$ are edges in graph G, then

$$\big(\mathrm{EC}_{(r\to s)} = 0\big) \text{ and } \big(\mathrm{EC}_{(r\to u)} = 0\big) \text{ and } \big(\mathrm{EC}_{(u\to t)} = 0\big) \implies \big(\mathrm{EC}_{(s\to t)} = 0\big) \tag{$\Diamond$}$$

Then a graph obtained by removing the edge $(s\to t)$ of G is still graphical representation of Δ_R^K since the edge-constraint of $(s\to t)$ is implied by other edge-constraints. We will refer to this property as the ($\Diamond$) property, because of the underlying diamond-shaped structures of the Hasse diagram, on which this property can be applied.

We now make precise the reductions in the graphical representation which are implied by the anti-transitivity property.

Definition 10.3. Edges $(u \rightarrow r)$ and $(v \rightarrow r)$ are said to be *Equivalent Edges for Removal* (*EER*), and denoted $(u \rightarrow r) \sim (v \rightarrow r)$ if there exists a sequence $(t_0 \rightarrow r), \cdots, (t_k \rightarrow r)$ of edges in G_R, with $t_0 = u$ and $t_k = v$ and with the property that $\forall i = 1, \cdots, k$, $\mathcal{A}(t_{i-1}) \cap \mathcal{A}(t_i) \neq \emptyset$, i.e. $\exists w_i \in R$ s.t. $t_{i-1} \subset w_i$ and $t_i \subset w_i$. □

Then it is easy to verify that this relation '$\sim$' is reflexive, symmetric and transitive and is hence indeed an equivalence relation. Therefore for each region $r \in R$, the collection of all the edges leading to r can be partitioned into equivalence classes of edges for removal (*EER classes* of region r). In the example of figure 10.1(a), $\{t, u, v\}$ and $\{w, x, y\}$ are the EER classes of z.

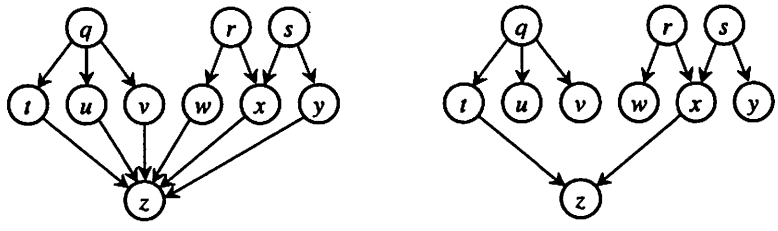


Figure 10.1: Equivalence of edges for removal
In the Hasse diagram (a) of R the EER classes of z are $\{t, u, v\}$ and $\{w, x, y\}$. (b) is a realization of S_R, the minimal graphical representation of R.

Definition 10.4. From each EER class $\{(t_1 \rightarrow r), \cdots, (t_m \rightarrow r)\}$, remove all but one representative edge from the Hasse graph G_R. Denote the resulting graph by S_R. □

Figure 10.1(b) shows one realization of S_R for the Hasse diagram of part (a).

Note that graph S_R is not unique, since the representative edge of each equivalence class can be arbitrarily chosen. However, the number of the edges of any choice of S_R is unique and equals the total number of EER classes of all regions. As we will see shortly, all choices of S_R result in equivalent, minimal graphical representations of R. *All results in the remainder of this dissertation apply to every choice of* S_R.

Lemma 10.1. *For every pair $u, s \in R$ such that $s \subset u$, there is a path in S_R between s and u consisting only of nodes that contain s.*

Proof. Clearly the Hasse diagram G_R has the claimed property; in fact if $s \subset u$, then there is a *directed* path from u to s consisting only of nodes that contain s.

Also note that if the claim is true for all pairs $s \prec u$ with $d(s) \leq l$, then it is also true for all pairs $s \subset u'$ with $d(s) \leq l$; this is because $s \subset u'$ implies that there exists a sequence of regions $s = u_0 \prec u_1 \prec \cdots \prec u_k = u'$, with $d(u_i) \leq l$ for $i = 0, \cdots k-1$. Then there is a path in S_R between each pair u_i and u_{i+1} consisting only of nodes that contain u_i and hence s.

Now to prove the claim for pairs $s \prec u$ we proceed by induction. Suppose first that $s \prec u$ and $d(s) = 1$. Then the edge $(u \rightarrow s)$ of G_R remains in S_R, since edges of depth 1 cannot be EER with other edges. Next assume inductively that we have proven the claim for all pairs $s' \prec u'$ with $d(s') \leq l$, and suppose $s \prec u$ is a pair of regions of R with $d(s) = l + 1$. Then from the definition of S_R there remains an edge $(v \rightarrow s)$ in S_R, a subset $\{t_0 = u, t_1, \cdots, t_k = v\}$ of parents of s and a sequence $\{w_1, \cdots, w_k\}$ of regions of R, s.t. $w_i \in \mathcal{A}(t_{i-1}) \cap \mathcal{A}(t_i)$. Each t_i has a depth of at most l, so for each $i = 1, \cdots, k$ there are paths in S_R from w_i to t_{i-1} and to t_i consisting only of nodes that contain t_{i-1} and t_i respectively. But $s \subset t_i$ for all i, and hence there is a path $(s, v, \cdots, w_k, \cdots, t_{k-1}, \cdots, w_{k-1}, \cdots, \cdots, w_1, \cdots, u)$ in S_R consisting only of nodes that contain s. This completes the proof. □

Now suppose S_R^1 and S_R^2 are two instances of S_R. As mentioned earlier, the number of edges of S_R^1 and S_R^2 are the same. Also by Lemma 10.1, the connected components of any S_R correspond one-to-one to those in G_R. Therefore S_R^1 is loop-free iff S_R^2 is loop-free, and in fact the number of loops of S_R^1 is equal to that of S_R^2. With this justification and based on the next proposition, we call S_R *the minimal graphical representation*, or *the minimal graph*, of R, and freely talk about existence of loops in S_R, as if S_R were unique.

Lemma 10.2. *Let $T \subset R$, and view $\mathcal{F}(T)$ as a sub-poset of R. Let S_R be a minimal graphical representation of R, and let G denote the subgraph of S_R on $\mathcal{F}(T)$. Then G is a minimal graphical representation of $\mathcal{F}(T)$. Furthermore, for each $t \in \mathcal{F}(T)$, the overcounting factors of t w.r.t. posets R and $\mathcal{F}(T)$ are the same.*

Proof. For each $t \in \mathcal{F}(T)$, the EER classes of t in poset $\mathcal{F}(T)$ are identical to those in R. Hence G by definition has one edge from each EER class, making it a minimal graphical representation of $\mathcal{F}(T)$, as claimed. Similarly, the overcounting factor c_t w.r.t. R depends only on the regions in $\mathcal{F}(t)$. These are all included in $\mathcal{F}(T)$. Therefore a simple inductive argument shows that the overcounting factors of t w.r.t. R and $\mathcal{F}(T)$ are identical. □

Based on this result, in the rest of this dissertation, when talking about a specific choice of S_R, we write $S_{\mathcal{F}(T)}$ to denote the subgraph of S_R on $\mathcal{F}(T)$, as the choice of the minimal graphical representation of $\mathcal{F}(T)$.

Proposition 10.3. *S_R is indeed a 'minimal' graphical representation of Δ_R^K, i.e. a collection of pseudo-marginals $\{b_r, r \in R\}$ lies in Δ_R^K iff it satisfies all the edge-constraints of S_R, and further, removal of any of the edges of S_R results in misrepresentation of Δ_R^K.*

Proof. To show that the collection of edge-constraints of S_R is a sufficient representation of Δ_R^K, note that by Lemma 10.1, for each $r \subset t$ there is a path between r and t with only nodes that contain r. Then the collection of edge-constraints of this path imply that $\sum_{\mathbf{x}_{t \setminus r}} b_t(\mathbf{x}_t) = b_r(\mathbf{x}_r)$. Therefore any collection $\{b_r, r \in R\}$ of pseudo-marginals satisfying all the edge-constraints of S_R belongs to Δ_R^K.

To prove minimality, let S'_R be a graph created from S_R by removing an edge $(t_1 \rightarrow u)$ of S_R. Let $\{(t_1 \rightarrow u), \cdots, (t_k \rightarrow u)\}$ be the corresponding EER class – all of these edges are now removed in creating S'_R. Define $T := \bigcup_{i=1}^{k} \bigcup_{r \in \mathcal{F}(t_i)} r$.

Let $b(\mathbf{x}) \geq \epsilon > 0$ be a positive distribution, with marginals $b_r(\mathbf{x}_r), r \subseteq [N]$. Then $\{b_r(\mathbf{x}_r), r \in R\} \in \Delta_R^K$. Now define another collection $\{b'_r(\mathbf{x}_r)\}$ as follows:

$$b'_T(\mathbf{x}_T) := \begin{cases} b_T(\mathbf{x}_T) + \epsilon \cdot (-1)^{\sum_{i \in u} x_i} & \text{if } x_i \in \{1,2\}\ \forall i \in u \\ b_T(\mathbf{x}_T) & \text{else} \end{cases}$$

Therefore $b'_T(\mathbf{x}_T)$ differs from $b_T(\mathbf{x}_T)$ by a small perturbation of $\pm\epsilon$ introduced only on the cube $\{1,2\}^{|u|}$, which is a subset of the range $\prod_{i \in u}[q_i]$ of $\mathbf{x}_u$.

We extend this to define $b'_r(\mathbf{x}_r)$ for all the regions $r \in R$ as follows:

$$\begin{aligned} b'_r(\mathbf{x}_r) &:= \sum_{\mathbf{x}_{T \setminus r}} b'_T(\mathbf{x}_T) && \text{if } r \in \bigcup_{i=1}^{k} \mathcal{F}(t_i) \\ b'_r(\mathbf{x}_r) &:= b_r(\mathbf{x}_r) && \text{else} \end{aligned}$$

We claim that (1) $b'_r = b_r$ for all $r \in R \setminus (\bigcup_{i=1}^k \mathcal{F}(t_i))$; (2) $b'_r \neq b_r$ for any $r \in (\bigcup_{i=1}^k \mathcal{F}(t_i))$, and (3) all the edge-constraints of S'_R are satisfied.

Part (1) is obvious from the definition of b' above.

Part (2) can be seen easily since for any $r \in \bigcup_{i=1}^k \mathcal{F}(t_i)$, $u \subset r$ and by definition, marginalizations of b'_T are different from those of b_T on the subsets that contain u.

To see part (3), let $(r_1 \rightarrow r_2)$ be any edge of S'_R. If both $r_1, r_2 \in R \setminus (\bigcup_{i=1}^k \mathcal{F}(t_i))$ or both $r_1, r_2 \in \bigcup_{i=1}^k \mathcal{F}(t_i)$, then by definition, b'_{r_1} is consistent with b'_{r_2}. If $r_2 \in \bigcup_{i=1}^k \mathcal{F}(t_i)$, then its parent, r_1 must also lie there. Therefore we need only check the case when r_1 lies in $(\bigcup_{i=1}^k \mathcal{F}(t_i))$ and r_2 does not. Note that in this case, r_2 cannot contain u.Also from definition of $b'_T(\mathbf{x}_T)$, the marginals of b'_T at the level of any subset that does not contain u coincide with the marginals of b_T since the ϵ terms cancel out. Hence b'_{r_1} marginalizes to b_{r_2} and since $b'_{r_2} = b_{r_2}$, the edge-constraint in S'_R corresponding to $(r_1 \rightarrow r_2)$ is also satisfied in this case.

We have therefore shown explicitly that there exist collections $\{b'_r\}$ of pseudo-marginals that satisfy all the edge-constraints of S'_R but not the edge-constraint corresponding to edge $(t_1 \rightarrow u)$ of S_R (in particular, $\sum_{\mathbf{x}_{t_1 \setminus u}} b'_{t_1}(\mathbf{x}_{t_1}) \neq b'_u(\mathbf{x}_u) = b_u(\mathbf{x}_u)$.) Therefore $\{b'_r\} \notin \Delta_R^K$. $\square$

As we have seen, to solve the constrained minimization problem one forms the Lagrangian, introducing multipliers $\lambda_{tr}(\mathbf{x}_r)$ for each edge $(t \to r)$ of S_R. Since S_R has fewer edges than any other graphical representation of R, algorithms based on S_R require the fewest message updates per each iteration.

10.1 Connection with Junction Trees

In this section we show that there is a close connection between the minimal graphical representation of a collection R of Kikuchi regions, and the junction trees on R.

Definition 10.5. Let $\{r_1, \cdots, r_M\}$ be a collection of subsets of the index set $[N]$. A tree/forest G with vertices $\{r_1, \cdots, r_M\}$ is called a *junction* tree/forest if it satisfies condition (A1) of Section 9.2, i.e. that for each $i \in [N]$, the subgraph consisting of all the vertices that contain i be connected.[1] □

Although junction trees are traditionally defined as undirected trees, in the above definition we do not make distinction between directed and undirected graphs; we call a directed graph a junction tree if replacing all the directed edges with undirected ones yields a junction tree in the usual sense.

Let $\{r_1, \cdots, r_M\}$ be the maximal elements of R. For the rest of this dissertation we assume that R is totally connected, i.e. it has property (An) for all $n = 1, \cdots, N$. Then it is easy to see from the definition above that the following proposition holds:

Proposition 10.4. *If S_R has no loops, then it is a junction forest and hence $\{r_1, \cdots, r_M\}$ can be put on a junction tree.*

Proof. The Hasse graph G_R satisfies (A1). Let $(u_0, u_1, \cdots, u_n)$ form a path in G_R where $i \in u_j$ for all $j = 0, \cdots, n$. Then for all $j = 1, \cdots, n$ either $u_{j-1} \prec u_j$ or $u_j \prec u_{j-1}$. Then by Lemma 10.1 there is a path between u_{j-1} and u_j for all

[1] Note that given that G has no loops, condition (A1) implies (An) for all n.

$j = 1, \cdots, n$ consisting only of nodes that contain i. This proves that S_R satisfies (A1). Therefore S_R is a junction forest on R.

Now starting with undirected version of S_R, successively absorb each node $r \in R \backslash \{r_1, \cdots, r_M\}$ together with all its connecting edges into one of its neighbors. The resulting graph will be a junction forest on $\{r_1, \cdots, r_M\}$, the maximal elements of R. □

Interestingly, the converse to this is also true:

Proposition 10.5. *If the maximal elements $\{r_1, \cdots, r_M\}$ can be put on a junction tree then S_R has no loops.*

Proof. First note that the existence of a junction tree on the maximal elements of R is equivalent to the existence of a junction tree on R.

Recall from Section 1.2 that we define the *local domain graph*, G_{LD} for the collection R, as a weighted complete graph with vertices corresponding to the regions $r \in R$, with the weight of the edge (r, s) defined as

$$w_{(r,s)} = |r \cap s|$$

Then from Theorem 4.1 of [2], any junction tree on R must be a maximal-weight spanning tree of G_{LD}. A maximal-weight spanning tree can be obtained using a greedy algorithm such as Kruskal's algorithm, see e.g. [8]: Start with an empty graph, H. Identify an edge of G_{LD} with the largest weight whose addition does not create a cycle in graph H, and add that edge to H. Repeat the preceding step until no more edges can be added.

Let G be the undirected version of the Hasse graph G_R, where each directed edge $(r \rightarrow s)$ is replaced by an undirected edge (r, s). Notice that at each stage of the above algorithm, one can choose an edge from the edge-set of G. To see this, suppose (t, r) is an edge in G_{LD} with maximal weight whose addition does not create a cycle. Then, since G is totally connected, there is a path in G where each vertex on the

path contains $t \cap r$; every edge in this path then has weight at least $|t \cap r| \geq w_{t,r}$. Now note that there must be an edge (t', r') in this path (in G), such that t' and r' are not connected in H (otherwise t and r would already be connected in H and so the addition of the edge (t, r) would create a cycle.) Therefore at this stage of the algorithm we can choose the edge (t', r') instead of (t, r). This shows that there is a junction tree on R whose edge-set is a subset of the edge-set of G.

Next note that the junction tree on R constructed in the preceding paragraph, under the hypothesis that a junction tree exists on R, must include at least one edge from each EER class of G. Specifically let u_0 be a parent of r in R, and let $(u_0, u_1, u_2, \cdots, u_m = r)$ be any path between u_0 and r in G such that each u_i includes r. Then it is easy to see that $(u_0 \rightarrow r)$ and $(u_{m-1} \rightarrow r)$ must be in the same EER class. This proves that any path between u_0 and r in G includes one of the edges in the same EER class with $(u_0 \rightarrow r)$, and therefore any junction subtree of G must have at least one edge from each EER class. Now remember that S_R was defined to be a graph with precisely one edge remaining from each EER class, so we could have chosen S_R as a subgraph of the junction tree we constructed. However, if S_R has a loop, it cannot be a subgraph of a tree. □

Recall that in Section 9.2 we originally defined the Kikuchi problem in terms of a collection R_0 of regions.

Definition 10.6. The original collection R_0 of regions is called *loop-free* if there exists a junction tree on its maximal elements, and is called *loopy* if no such junction tree exists. □

Now as before, let R be any poset of regions with the same maximal regions as R_0, which is totally connected. Then by Propositions 10.4 and 10.5 and the definition above, R_0 is called loopy iff S_R has a loop.

10.2 Necessary and Sufficient Conditions for Exactness of Kikuchi Method

It is well-known that the belief propagation algorithm converges to the exact marginals,– in finite time, – if the 'underlying graph' is loop-free, see e.g. [33], [10]. Likewise, the message-passing algorithms of the type discussed in Section 9.3, will converge to yield the exact marginals if the Hasse diagram is loop-free, and in fact the value of the Kikuchi functional equals the variational free energy. However this is a rather weak result, since only very rarely will the Hasse diagram be loop-free. In fact many collections of regions that can be put on a junction tree result in Hasse diagrams that have loops. For example the poset $R = \{\{123\}, \{234\}, \{345\}, \{23\}, \{34\}, \{3\}\}$ will have a loop in the Hasse diagram as displayed in Figure 10.2(a), but can be easily handled as a junction tree 10.2(b).

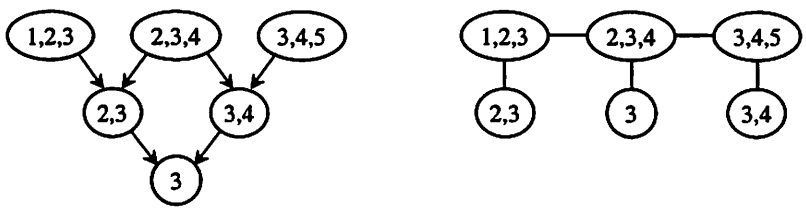


Figure 10.2: Hasse diagram vs. junction tree

Also, in the example of Figure 10.1, even though the Hasse diagram has loops, the solution to the Kikuchi approximation problem equals the exact marginals. This is because not all the loops of G_R are 'bad' loops that cause trouble for the message-passing algorithm. In fact these 'bad' loops are precisely the loops that cannot be broken when one creates S_R. In the examples of both Figures 10.1 and 10.2, S_R is loop-free. One can therefore run a message passing algorithm on S_R, which converges to yield the solution for the Kikuchi approximation (9.14) which is identical to the exact marginals. In fact in these examples, the Kikuchi free energy functionalequals the variational free energy. The results in this subsection are aimed at making these

observations precise.

Let R be an arbitrary collection of regions. Note that the following lemma does *not* assume that the poset R is totally connected.

Lemma 10.6. *If S_R is a tree, then the overcounting factor for each region $r \in R$ satisfies*

$$c_r = 1 - |\mathcal{P}_{S_R}(r)|,$$

where $|\mathcal{P}_{S_R}(r)|$ is the number of parents of r in S_R, as defined earlier in Section 10. Furthermore, the sum of the overcounting factors of all regions equals 1.

Proof. We will show this by induction on the maximum depth of regions of R.

If R has maximum depth of 0, and given that S_R is a tree, R must necessarily consist only of one region. Then the claim then holds immediately for the single region of R.

Now suppose the lemma holds for all posets R' with maximum depth l. Suppose a poset R has maximum depth $l+1$ and S_R is a tree. Let $\{(s_1^1 \to r), \cdots, (s_{k_1}^1 \to r)\}, \cdots, \{(s_1^m \to r), \cdots, (s_{k_m}^m \to r)\}$ be all the EER classes of a given region $r \in R$. Let $T_1 := \cup_{i=1}^{k_1} s_i^1, \cdots, T_m := \cup_{i=1}^{k_m} s_i^m$ be the parents of r corresponding to each EER class. Then $\mathcal{F}(T_1), \cdots, \mathcal{F}(T_m)$ must be disjoint or else two of the EER classes could be merged into a bigger class. Then by induction hypothesis, the sum of overcounting factors for each sub-poset $\mathcal{F}(T_1), \cdots, \mathcal{F}(T_m)$ is 1, and by Lemma 10.2 these are the same as the overcounting factors w.r.t. R. Then

$$c_r = 1 - \sum_{s \in \mathcal{A}(r)} c_s = 1 - \sum_{i=1}^{m} \sum_{u \in T_i} c_u = 1 - \sum_{i=1}^{m} 1 = 1 - |\mathcal{P}_{S_R}(r)|$$

since the number of parents of r in S_R is precisely the number of EER classes of r.

Now consider the sum of the overcounting factors of all nodes:

$$\begin{aligned}\sum_{r\in R} c_r &= \sum_{r\in R} \left(1 - |\mathcal{P}_{S_R}(r)|\right) \\ &= \sum_{r\in R} 1 - \sum_{(s\to r)\in\mathcal{E}(S_R)} 1 \\ &= |V(S_R)| - |\mathcal{E}(S_R)| \\ &= 1\end{aligned}$$

where the last equality follows from the fact that S_R is assumed to be a tree, so the number of its vertices is one more than the number of its edges. This completes the inductive step of the proof. □

The following theorem states sufficient conditions for the Kikuchi approximate free energy and the consistency constraint set of pseudo-marginals to be exact:

Theorem 10.7. *(Exactness of Kikuchi approximates, Δ_R^K and F_R^K)*

A) $\Delta_R^K = \Delta_R$ *if* S_R *is loop-free.*

B) Let $b(\mathbf{x})$ *be a distribution with marginals* $b_r(\mathbf{x}_r)$. *Then*
$F_R^K(\{b_r,\, r \in R\}) = F(b)$ if $b(\mathbf{x}) = \prod_{r\in R} b_r(\mathbf{x}_r)^{c_r}\ \ \forall \mathbf{x}$

Proof. Part A): Suppose S_R is loop-free, and let $\{b_r(\mathbf{x}_r),\, r \in R\} \in \Delta_R^K$. Since S_R is a junction forest, there is a distribution $b(\mathbf{x}) := \prod_{r\in R} b_r(\mathbf{x}_r) / \prod_{(t\to u)\in\mathcal{E}(S_R)} b_u(\mathbf{x}_u)$ that marginalizes to $\{b_r(\mathbf{x}_r),\, r \in R\}$; this is a well-known result on the junction trees, which can be verified by marginalizing $b(\mathbf{x})$ in stages, from the leaves (of the undirected version of S_R) towards an arbitrary region r as the root, where at each step, by local consistency there will be cancellation. Therefore $\{b_r(\mathbf{x}_r),\, r \in R\} \in \Delta_R$, and so $\Delta_R^K \subseteq \Delta_R$. But clearly $\Delta_R \subseteq \Delta_R^K$ since the true marginals of any distribution are locally consistent. Therefore $\Delta_R^K = \Delta_R$.

Part B): From discussion of Section 9.1, $F_R^K(\{b_r, r \in R\}) = F(b)$ if the entropy approximation of equation (9.8) is exact. Now

$$
\begin{aligned}
& b(\mathbf{x}) = \prod_{r \in R} b_r(\mathbf{x}_r)^{c_r} \\
\Longrightarrow \quad & \mathrm{KL}\left(b(\mathbf{x}) \Big|\Big| \prod_{r \in R} b_r(\mathbf{x}_r)^{c_r}\right) = 0 \\
\Longrightarrow \quad & \sum_{\mathbf{x}} b(\mathbf{x}) \Big(\log(b(\mathbf{x})) - \log(\prod_{r \in R} b_r(\mathbf{x}_r)^{c_r}) \Big) = 0 \\
\Longrightarrow \quad & \sum_{\mathbf{x}} b(\mathbf{x}) \Big(\log(b(\mathbf{x})) - \sum_{r \in R} c_r \log(b_r(\mathbf{x}_r)) \Big) = 0 \\
\Longrightarrow \quad & \sum_{\mathbf{x}} b(\mathbf{x}) \log(b(\mathbf{x})) = \sum_{r \in R} c_r \sum_{\mathbf{x}} b(\mathbf{x}) \log(b_r(\mathbf{x}_r)) \\
\Longrightarrow \quad & \sum_{\mathbf{x}} b(\mathbf{x}) \log(b(\mathbf{x})) = \sum_{r \in R} c_r \sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) \log(b_r(\mathbf{x}_r)) \\
\Longrightarrow \quad & F_R^K(\{b_r(\mathbf{x}_r), r \in R\}) = F(b(\mathbf{x}))
\end{aligned}
$$

□

Corollary 10.8. *If R_0 is loop-free, then the constrained minimization problem* (9.14) *has a unique solution. Further, the solutions $\{b_r^*, r \in R\}$ is the exact marginals of the product function, and the minimum free energy equals the log-partition function, i.e. $b_r^*(\mathbf{x}_r) = B_r(\mathbf{x}_r)$ and $F^* = -\log(Z)$.* [2]

Proof. From Theorem 10.7, if S_R has no loops then $\Delta_R^K = \Delta_R$, and further for each $\{b_r\} \in \Delta_R^K$, the function $\prod_{r \in R} b_r^{c_r}(\mathbf{x}_r)$ is a valid distribution on $\mathbf{x}$ which marginalizes to the b_r's, and therefore $F_R^K(\{b_r\}) = F(\prod_{r \in R} b_r^{c_r})$. On the other hand, as stated in Section 9.1, the minimum of $F(b)$ is achieved uniquely with Boltzmann distribution, $B(\mathbf{x}) := \frac{1}{Z} \prod_{r \in R} \alpha_r(\mathbf{x}_r)$. Now clearly $\{B_r, r \in R\} \in \Delta_R = \Delta_R^K$, since $\{B_r, r \in R\}$ is the set of R-marginals of a valid distribution $B(\mathbf{x})$. Therefore $\{B_r, r \in R\}$ is a minimizer of (9.14), and the corresponding minimum F^* indeed equals the minimum F_0 of $F(b)$, which in turn is equal to $-\log(Z)$.

[2] In fact in the case when R_0 is loop-free, iterative algorithms such as GBP, which we will discuss in Section 11, converge in finite time to the unique solutions b_r^*.

To see the uniqueness, note that if $\{b_r^*,\, r \in R\} \in \Delta_R^K$ is any minimizer of (9.14), then the corresponding distribution $b^*(\mathbf{x}) := \prod_{r\in R} b_r^{*c_r}(\mathbf{x}_r)$ is a distribution, with marginals $\{b_r^*\}$, which minimizes the variational free energy $F(b)$; but $B(\mathbf{x})$ is the unique minimizer of $F(b)$. Therefore $b^*(\mathbf{x})$ must be equal to $B(\mathbf{x})$, and accordingly $\{b_r^*\} = \{B_r\}$. □

The above results show that a sufficient condition for the exactness of the solutions of Kikuchi approximation method of (9.14) is that S_R be loop-free. In the sequel we address the necessary conditions for exactness.

We first pose the following, more abstract question about entropy approximations: *Under what conditions is an entropy approximation in the form of equation* (9.8) *exact for all R-decomposable distributions?* The following theorem, answers this question.

Theorem 10.9. *Let R be a totally connected poset of subsets of* $[N]$, *and let* $\{k_r\,,\, r \in R\}$ *be a collection of constants. Then the following are equivalent:*

(1) $H(B) = \sum_{r\in R} k_r\, H_r(B_r)$ *for all R-decomposable distributions* $B(\mathbf{x})$.

(2) $\sum_{r\in \cup_{i\in s}\mathcal{F}(i)} k_r = 1$ *for all* $s \subseteq [N]$ *such that* $\cup_{i\in s}\mathcal{F}(i)$ *is connected in* S_R.

(3) $\sum_{r\in \cup_{s\in S}\mathcal{F}(s)} k_r = 1$ *for all* $S \subseteq 2^{[N]}$ *such that* $\cup_{s\in S}\mathcal{F}(s)$ *is connected in* S_R.

Further, given the poset R, there exists a collection $\{k_r\,,\, r \in R\}$ *satisfying (1), (2) and (3) above, iff* S_R, *the minimal graph of R, is loop-free. If such collection* $\{k_r\,,\, r \in R\}$ *exists, then it is unique and equals the (Möbius) overcounting factors* $\{c_r\,,\, r \in R\}$.

Proof. See Appendix B.1 □

From this theorem, proof of Proposition 9.1 of Section 9.1 follows:

Proof of Proposition 9.1. Note that although the statement of Theorem 10.9 assumes that R is totally connected, that assumption is only needed to show that (3) implies (1). From the proof of Theorem 10.9, for an arbitrary collection of regions R, any collection of factors $\{k_r, r \in R\}$ satisfying condition (1) of Theorem 10.9 must be the Möbius overcounting factors. □

As mentioned in Section 9.2, given a product distribution with local kernels $\{\alpha_r^0(\mathbf{x}_r),\, r \in R_0\}$, where the regions in R_0 cannot be put on a junction tree, it is expected that expanding the collection R_0 by adding subsets of $r \in R_0$ as further regions would improve the quality of approximation obtained by the iterative algorithms such as GBP. The following result, however, shows that it is improbable that *exact* solutions will be obtained.

Theorem 10.10. *If R_0 is loopy, then except on a set of measure zero of choices of kernels $\{\alpha_r^0(\mathbf{x}_r),\, r \in R_0\}$, the Kikuchi approximation method of* (9.14) *will not produce the exact solutions.*

Proof. Let T denote the set of R_0-kernels $\{\alpha_r^0(\mathbf{x}_r)\}$ for which the Kikuchi entropy approximation of equation (9.8) is exact for the Boltzmann distribution. Specifically, define

$$T := \Big\{\, \{\alpha_r^0(\mathbf{x}_r), r \in R_0\} : \sum_{\mathbf{x}} B(\mathbf{x}) \log(B(\mathbf{x})) = \sum_{r \in R} c_r \sum_{\mathbf{x}_r} B_r(\mathbf{x}_r) \log(B_r(\mathbf{x}_r)) \,\Big\} \tag{10.3}$$

where, as in Section 9.2, $B(\mathbf{x}) = \frac{1}{Z} \prod_{r \in R_0} \alpha_r^0(\mathbf{x}_r)$ and B_r's are its marginals. For each region $r \in R_0$ define $q_r := \prod_{i \in r} q_i$ to be the cardinality of the range of $\mathbf{x}_r$, and let $l := \sum_{r \in R_0} q_r$. Let $f : \mathbb{R}_+^l \longrightarrow \mathbb{R}$ be the error function in approximating entropy as

in (9.8), i.e.

$$f(\{\alpha_r^0(\mathbf{x}_r)\}) := \sum_{\mathbf{x}} B(\mathbf{x}) \log(B(\mathbf{x})) - \sum_{r \in R} c_r \sum_{\mathbf{x}_r} B_r(\mathbf{x}_r) \log(B_r(\mathbf{x}_r))$$
$$= \sum_{\mathbf{x}} \frac{\prod_{s \in R} \alpha_s^0(\mathbf{x}_s)}{\sum_{\mathbf{x}'} \prod_{s \in R} \alpha_s^0(\mathbf{x}'_s)} \log(\frac{\prod_{s \in R} \alpha_s^0(\mathbf{x}_s)}{\sum_{\mathbf{x}'} \prod_{s \in R} \alpha_s^0(\mathbf{x}'_s)}) -$$
$$\sum_{r \in R} c_r \sum_{\mathbf{x}_r} \frac{\sum_{\mathbf{x}_{[N]\setminus r}} \prod_{s \in R} \alpha_s^0(\mathbf{x}_s)}{\sum_{\mathbf{x}'} \prod_{s \in R} \alpha_s^0(\mathbf{x}'_s)} \log(\frac{\sum_{\mathbf{x}_{[N]\setminus r}} \prod_{s \in R} \alpha_s^0(\mathbf{x}_s)}{\sum_{\mathbf{x}'} \prod_{s \in R} \alpha_s^0(\mathbf{x}'_s)})$$

Then $T = f^{-1}(0)$. Function $f(\cdot)$ above is clearly analytic on its domain, $\mathbb{R}_+^l$. Then, as demonstrated in [12] §3.1.24, either $T = \mathbb{R}_+^l$ or $\mu(T) = 0$, where $\mu(\cdot)$ is the Lebesgue measure. The first alternative requires that f be identically zero on $\mathbb{R}_+^l$. But from Theorem 10.9 it is evident that if R_0 is loopy, then the entropy approximation cannot be exact. This completes the proof. □

A stronger version of this result can be derived. Although we have so far focused on positive distributions $B(\mathbf{x}) > 0$, in many applications one is interested in distributions that can be zero at certain points of the state space. As mentioned above, using its continuity, the function $x \log(x)$ can be extended conveniently at the point $x = 0$. This means that we can handle arbitrary (not necessarily positive) distributions in the above framework.

Suppose then we condition on the collections of kernels $\{\alpha_r^0(\mathbf{x}_r), r \in R_0\}$ which are zero at predetermined values. Specifically, for each $r \in R_0$, let $Z_r \subset \prod_{i \in r}[q_i]$ be a subset of the range of values of $\mathbf{x}_r$, on which α_r^0 is zero. We say a collection of kernels $\{\alpha_r^0(\mathbf{x}_r), r \in R_0\}$ is *consistent* with $\{Z_r, r \in R_0\}$ if for each $r \in R_0$, $\alpha_r^0(\mathbf{x}_r) = 0$ for all $\mathbf{x}_r \in Z_r$.

Theorem 10.11. *Suppose R_0 is loopy, and let $\{Z_r, r \in R\}$ be a collection of zeros of the kernels as defined above, chosen such that the following holds:*

$$\forall\, s \subseteq [N],\ \exists\, \tilde{\mathbf{x}}_s^1, \tilde{\mathbf{x}}_s^2, \tilde{\mathbf{x}}_{[N]\setminus s} \text{ such that } \tilde{x}_i^1 \neq \tilde{x}_i^2\ \forall i \in s, \text{ and that } B(\tilde{\mathbf{x}}_s^1, \tilde{\mathbf{x}}_{[N]\setminus s})$$
$$\text{and } B(\tilde{\mathbf{x}}_s^2, \tilde{\mathbf{x}}_{[N]\setminus s}) \text{ can both be nonzero under the restrictions imposed by } \{Z_r\}. \tag{10.4}$$

Then the conditional measure of the set of kernels $\{\alpha_r^0(\mathbf{x}_r), r \in R_0\}$ *conditioned to be consistent with* $\{Z_r, r \in R_0\}$, *for which Kikuchi approximation method of* (9.14) *is exact is zero.*

Proof. See Appendix B.2 □

We conclude this section with a generalization of Corollary 9.4 on sufficient conditions for convexity of the Kikuchi free energy. In particular we make a (purely set-theoretic) connection between the number of loops of S_R and the sufficient conditions of Theorem 10.13 for convexity of Kikuchi free energy.

Definition 10.7. We call a collection R of Kikuchi regions *normal,* if for all $S \subset R$, there exist a largest region (w.r.t. set-inclusion) $m_S \in \bigcap_S \mathcal{D}(s)$, possibly the empty set, that contains any other region $u \in \bigcap_S \mathcal{D}(s)$.

□

Lemma 10.12. *If* R *is normal and* S_R *is connected and has exactly one loop, then* $\sum_{r \in R} c_r \in \{0, 1\}$.

Proof. Let $L \subseteq R$ be the set of nodes in the single loop of S_R, and let $\widetilde{L} := \mathcal{F}(L)$. Note that L cannot have a single minimum region, i.e. a region that is contained in all other regions of L; to see this, suppose to the contrary that r_0 is the minimum region of L. Then all the edges of the Hasse diagram that terminate in r_0 and participate in the loop L would be EER; all but one of these edges would be removed in S_R, therefore r_0 could not be part of the loop L, which is a contradiction.

It follows from this that for each $r \in \widetilde{L}$, the sub-poset $\mathcal{F}(r)$ does not contain the loop L, and is hence loop-free. Then by Lemmas 10.6 and 10.2, $c_r = 1 - |\mathcal{P}_{S_{\mathcal{F}(r)}}(r)| = 1 - |\mathcal{P}_{S_{\widetilde{L}}}(r)|$. Then, as argued in the proof of Lemma 10.6, there is a contribution of $+1$ for each vertex and -1 for each edge of $S_{\widetilde{L}}$ for the sum $\sum_{s \in \widetilde{L}} c_s$. Now $S_{\widetilde{L}}$ is connected and has one loop, so the number of its vertices equal the number of its edges and so $\sum_{s \in \widetilde{L}} c_s = 0$.

Now for each $r \in R \backslash \widetilde{L}$ such that $S_{\widetilde{L} \cup \mathcal{F}(r)}$ is connected, we calculate the contribution of the overcounting factors of regions in $\mathcal{F}(r) \backslash \widetilde{L}$ to the overall sum. After each stage, we will inductively append $\mathcal{F}(r)$ to $\widetilde{L}$.

First suppose that $\mathcal{F}(r)$ does not contain the loop L of R; then

$$\sum_{s \in \widetilde{L} \cup \mathcal{F}(r)} c_s = \sum_{s \in \widetilde{L}} c_s + \sum_{s \in \mathcal{F}(r) \backslash \widetilde{L}} c_s \tag{10.5}$$

We will argue that the second term must equal 0. Remember that from the definition of the overcounting factors,

$$1 = \sum_{s \in \mathcal{F}(r)} c_s = \sum_{s \in \mathcal{F}(r) \cap \widetilde{L}} c_s + \sum_{s \in \mathcal{F}(r) \backslash \widetilde{L}} c_s \tag{10.6}$$

But $S_{\mathcal{F}(r) \cap \widetilde{L}}$ must be a tree, since firstly it does not contain the loop L, and secondly r can have only one parent in $S_{\mathcal{F}(r) \cap \widetilde{L}}$ or else S_R would have a loop containing r. Therefore by Lemma 10.6, $\sum_{s \in \mathcal{F}(r) \cap \widetilde{L}} c_s = 1$. Then from (10.6), $\sum_{s \in \mathcal{F}(r) \backslash \widetilde{L}} c_s = 0$ and hence by (10.5), $\sum_{s \in \widetilde{L} \cup \mathcal{F}(r)} c_s = \sum_{s \in \widetilde{L}} c_s$, i.e. the sum is preserved after appending $\mathcal{F}(r)$.

Next suppose that $\mathcal{F}(r)$ contains the loop L of R. Then since R is normal, there is a largest $m \in R$ such that $\mathcal{F}(m)$ contains the loop. Again we break up the new sum of overcounting factors as in equation (10.5). If r is equal to m, $\sum_{s \in \mathcal{F}(r) \cap \widetilde{L}} c_s$ in (10.6) equals 0 since $S_{\mathcal{F}(r) \cap \widetilde{L}}$ has one loop and has no region r' such that $\mathcal{F}(r')$ contains the loop and hence by what we have shown so far, $\sum_{s \in S_{\mathcal{F}(r) \cap \widetilde{L}}} c_s = 0$. Therefore by (10.6), $\sum_{s \in \mathcal{F}(r) \backslash \widetilde{L}} c_s = 1$, and hence from (10.5), $\sum_{s \in \widetilde{L} \cup \mathcal{F}(m)} c_s = \sum_{s \in \widetilde{L}} c_s + 1 = 1$.

On the other hand, whenever $\mathcal{F}(r)$ contains the loop but $r \neq m$, then $m \subset R$ and hence $\mathcal{F}(m) \subset \widetilde{L} \cap \mathcal{F}(r)$. Then by what has been shown so far, $\sum_{s \in \widetilde{L} \cap \mathcal{F}(r)} c_s = 1$. Then the additional term $\sum_{s \in \mathcal{F}(r) \backslash \widetilde{L}} c_s$ in (10.5) is 0 and hence $\sum_{s \in \widetilde{L} \cup \mathcal{F}(r)} c_s = \sum_{s \in \widetilde{L}} c_s = 1$.

Therefore we have shown that, depending on whether $\bigcap_{s \in L} \mathcal{D}(s)$ is empty or not, the sum $\sum_{r \in R} c_r$ is 0 or 1. □

Theorem 10.13. *Let R be a* normal *collection of Kikuchi regions. Then the Kikuchi free energy functional $F_R^K(\{b_r\})$ is strictly convex if S_R has zero or one loop. In*

particular, the Kikuchi free energy for the cluster variation method of [49] is strictly convex if S_R has zero or one loop.

Proof. For each $S \subset R$, $\mathcal{F}(S)$ contains zero or one loop. Then by Lemmas 10.6 and 10.12, $\sum_{s \in \mathcal{F}(S)} c_s \geq 0$ and hence, from Theorem 9.3 the Kikuchi functional is strictly convex. □

Chapter 11

Generalized Belief Propagation Algorithm

We are now in position to describe a class of iterative message-passing algorithms that try to solve the constrained minimization problem (9.14). Previously described algorithms such as the generalized belief propagation (GBP) algorithms of [49] and [50], and Poset-BP algorithm of [23] are special cases of the class of algorithms we describe. The algorithms proposed earlier work on the full Hasse diagram. The results derived in the earlier chapters on the minimal graphs allow us to propose algorithms for solving (9.14) which are often substantially less complex than the ones proposed in [50] and [23], and which appear to have comparable convergence performance in some examples we have investigated and reported on Section 12.

Let R be the collection of regions for a Kikuchi approximation problem. In Section 9.3 we described how the Lagrange multipliers method can be used to obtain an iterative, message-passing algorithm with fixed points that coincide with the stationary points of (9.14).

Now let G be any graphical representation of Δ_R^K as defined in Section 10. Then the Lagrangian of equation (9.15) can be rewritten in terms of the edge-constraints of G, in which case the 'messages' of the resulting iterative algorithm can be identified precisely with the edges of G. This means that, for each graphical representation

of Δ_R^K there is a distinct message-passing algorithm along the edges of that graph. Clearly all such algorithms have the same set of fixed points, although the dynamics of each algorithm may be different.

So far we have represented the constraint set Δ_R^K using the edge-constraints defined in Section 10. Motivated by an observation made by Yedidia, Freeman and Weiss in [49; 50], we introduce an alternative but essentially equivalent set of edge-constraints; we will then be able to use this alternative representation of the constraint set Δ_R^K to derive an alternative message-passing algorithm to solve (9.14).

Definition 11.1. The *YFW edge-constraint*[1] for an edge $(s \to t)$ of G is defined as the following functional of the pseudo-marginals $\{b_r\,,\, r \in R'\}$:

$$\mathrm{EC}'_{(s\to t)}(\{b_r\,,\, r \in R'\}) := \sum_{u \in \mathcal{F}(t)\setminus\mathcal{F}(s)} c_u \sum_{\mathbf{x}_{u\setminus t}} b_u(\mathbf{x}_u) \tag{11.1}$$

where $R' := \{r \in R,\ c_r \neq 0\}$ is the collection of regions with non-zero overcounting factors. Note that, since $c_u = 0$ for $u \in R'$, $\mathrm{EC}'_{(s\to t)}$ is a function of only $\{b_r, r \in R'\}$ as claimed in (11.1). When the arguments are clear from the context, we abbreviate these edge-constraints as $\mathrm{EC}'_{(s\to t)}$.

□

Proposition 11.1. *The collection of pseudo-marginals represented by the YFW edge-constraints is equal to the restriction of Δ_R^K to R'. Namely, if we define*

$$\Delta'_R := \Big\{\, \{b_r(\mathbf{x}_r),\, r \in R'\} \;:\; \forall (s \to t) \in \mathcal{E}(G),\ \mathrm{EC}'_{(s\to t)}(\{b_r, r \in R'\}) = 0 \\ \text{and } \forall r \in R',\ \sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) = 1 \Big\}, \tag{11.2}$$

then $\Delta'_R = \Delta_R^K\big|_{R'}$, where

$$\Delta_R^K\big|_{R'} := \big\{\, \{b_r(\mathbf{x}_r), r \in R'\} :\ \{b_r(\mathbf{x}_r), r \in R'\} \text{ has an extension } \{b_r(\mathbf{x}_r), r \in R\} \in \Delta_R^K \,\big\}$$

is the restriction of Δ_R^K to R'.

[1] We call these constraints YFW after Yedidia, Freeman and Weiss.

Proof. See Appendix B.3 □.

Remark. Note from (9.11) and (9.13) that Kikuchi free energy can be rewritten as follows:

$$F_R^K(\{b_r(\mathbf{x}_r)\}) = \sum_{r\in R}\sum_{\mathbf{x}_r}\big(-c_r\, b_r(\mathbf{x}_r)\log(\beta_r(\mathbf{x}_r)) + c_r\, b_r(\mathbf{x}_r)\log(b_r(\mathbf{x}_r))\big) \quad (11.3)$$

From (11.3) it is apparent that $F_R^K(\{b_r, r\in R\})$ only depends on $\{b_r, r\in R'\}$, since the terms involving the pseudo-marginals corresponding to the regions with zero overcounting factors are multiplied by zero. Therefore (9.14) can be rewritten as follows

$$\min_{\{b_r, r\in R\}\in\Delta_R^K} F_R^K(\{b_r, r\in R\}) = \min_{\{b_r, r\in R'\}\in\Delta_R'} F_R^K(\{b_r, r\in R'\})$$
$$\text{and}\quad \Big(\arg\min_{\{b_r, r\in R\}\in\Delta_R^K} F_R^K(\{b_r, r\in R\})\Big)\Big|_{R'} = \arg\min_{\{b_r, r\in R'\}\in\Delta_R'} F_R^K(\{b_r, r\in R'\}) \quad (11.4)$$

In other words, the central constrained minimization problem (9.14) is reduced to the following: $\min_{\{b_r, r\in R'\}\in\Delta_R'} F_R^K(\{b_r, r\in R'\})$.

□

We will now write the Lagrangian for (11.4) using the YFW edge-constraints:

$$\mathcal{L} := \sum_{r\in R} c_r\, b_r(\mathbf{x}_r)\log\Big(\frac{b_r(\mathbf{x}_r)}{\beta_r(\mathbf{x}_r)}\Big) + \sum_{(r\to t)\in\mathcal{E}(G)}\sum_{\mathbf{x}_t}\lambda_{rt}(\mathbf{x}_t)\sum_{u\in\mathcal{F}(t)\setminus\mathcal{F}(r)} c_u\sum_{\mathbf{x}_{u\setminus t}} b_u(\mathbf{x}_u) + \sum_{r\in R}\kappa_r\Big(\sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) - 1\Big) \quad (11.5)$$

Setting partial derivative $\partial\mathcal{L}/\partial b_r(\mathbf{x}_r) = 0$ for each region r and each value of $\mathbf{x}_r$, and identifying 'messages,' for each edge $(p\to r)$, as $m_{pr}(\mathbf{x}_r) := e^{-\lambda_{pr}(\mathbf{x}_r)}$ we obtain:

$$b_r(\mathbf{x}_r) = k\,\beta_r(\mathbf{x}_r)\Big(\prod_{p\in\mathcal{P}_G(r)} m_{pr}(\mathbf{x}_r)\Big)\Big(\prod_{d\in\mathcal{D}(r)}\prod_{p'\in\mathcal{P}_G(d)\setminus(\{r\}\cup\mathcal{D}(r))} m_{p'd}(\mathbf{x}_d)\Big) \quad (11.6)$$

where constant k is chosen to normalize b_r so it will sum to 1, and message m_{pr} is

updated to satisfy the original edge-constraint $\sum_{\mathbf{x}_{p\setminus r}} b_p(\mathbf{x}_p) - b_r(\mathbf{x}_r) = 0$:

$$m_{pr}(\mathbf{x}_r) = k' \frac{\sum_{\mathbf{x}_{p\setminus r}} \beta_p(\mathbf{x}_p) \Big(\prod_{s \in \mathcal{P}_G(p)} m_{sp}(\mathbf{x}_p) \Big) \Big(\prod_{d \in \mathcal{D}(p)} \prod_{s' \in \mathcal{P}_G(d) \setminus (\{p\} \cup \mathcal{D}(p))} m_{s'd}(\mathbf{x}_d) \Big)}{\beta_r(\mathbf{x}_r) \Big(\prod_{s \in \mathcal{P}_G(r) \setminus \{p\}} m_{sr}(\mathbf{x}_r) \Big) \Big(\prod_{d \in \mathcal{D}(r)} \prod_{p' \in \mathcal{P}_G(d) \setminus (\{r\} \cup \mathcal{D}(r))} m_{p'd}(\mathbf{x}_d) \Big)} \tag{11.7}$$

where k' is any convenient constant. Note that the common terms from the numerator and denominator of (11.7) can be cancelled, but to avoid even longer formulas we will not write the explicit form here.

The fixed points of equations (11.6) and (11.7) set all the derivatives of the Lagrangian equal to zero, and hence are precisely the stationary points of the Kikuchi free energy F_R^K subject to constraint set Δ_R^K.

The algorithm of equations (11.6) and (11.7) is defined on any graphical representation of Δ_R^K, and has as many messages as the edges of the underlying graph. From results of Section 10 then, using S_R, the minimal graphical representation, yields the least complex such algorithm in this sense. In fact in most cases the algorithm on S_R is substantially less complex than the full version implemented on the Hasse diagram G_R.

Remark. A version of this algorithm was originally labeled GBP in [49]. In [23] also the authors described an algorithm called 'Poset-BP' which is equivalent to the restriction of our results when G is the Hasse diagram. Our result shows that in general there are algorithms with strictly fewer messages, that have the same fixed points. In particular, the messages corresponding to the edges of the Hasse diagram that are removed in forming a more compact graphical representation, can be set to 1 in the entire algorithm. Not only the messages corresponding to the removed edges need not be updated at each iteration of the algorithm, the update rules for the remaining messages are also less complex, since they depend on fewer edges.

It is also noteworthy that the proofs given in [50] and [23] both presume that the poset is first simplified by removing the regions with zero overcounting factors. We

note however that removing the regions with zero overcounting factors can in general alter the problem. This is because a region with zero overcounting factor may still serve to ensure consistency between the pseudo-marginals at other regions (see e.g. the poset in Figure 11.1). We have avoided this restriction, by proving the results for a general poset. □

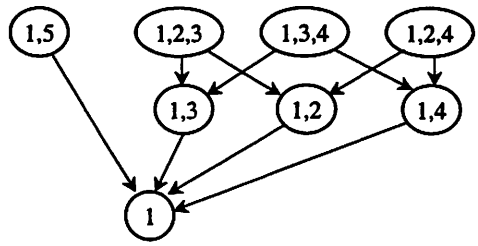


Figure 11.1: A simple poset
Region $\{1\}$ has zero overcounting, but cannot be removed.

Consider now the restriction of the above algorithm in the Bethe case, i.e. each region in R is either maximal or minimal w.r.t. inclusion. Then $\mathcal{P}(r) = \emptyset$ for a maximal region r, and $\mathcal{D}(s) = \emptyset$ for a minimal region s. To demonstrate the connection with the belief propagation algorithm, in addition to the messages $m_{pr}(\mathbf{x}_r)$ for $r \subset p$, we also define messages from a child to parent as follows:

$$n_{rp}(\mathbf{x}_r) := \beta_r(\mathbf{x}_r) \prod_{s \in \mathcal{P}(r) \setminus \{p\}} m_{sr}(\mathbf{x}_r) \tag{11.8}$$

Then, by equation (11.6), for a maximal region $p \in R$,

$$b_p(\mathbf{x}_p) = k\, \beta_p(\mathbf{x}_p) \prod_{d \in \mathcal{D}(p)} n_{pd}(\mathbf{x}_d) \tag{11.9}$$

Similarly, for a minimal region $r \in R$,

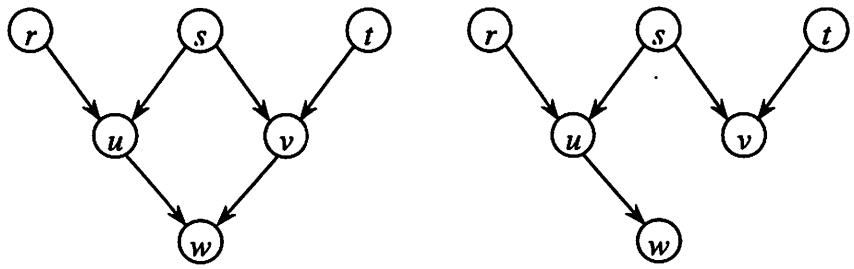
$$b_r(\mathbf{x}_r) = k\, \beta_r(\mathbf{x}_r) \prod_{d \in \mathcal{P}(r)} m_{dr}(\mathbf{x}_r) \tag{11.10}$$

The update equation (11.7) for messages m_{pr} can then be rewritten as

$$\begin{aligned} m_{pr}(\mathbf{x}_r) &= k' \frac{\sum_{\mathbf{x}_{p \setminus r}} \beta_p(\mathbf{x}_p) \prod_{d \in \mathcal{D}(p)} n_{dp}(\mathbf{x}_d)}{\beta_r(\mathbf{x}_r)\, n_{rp}(\mathbf{x}_r)} \\ &= k' \sum_{\mathbf{x}_{p \setminus r}} \frac{\beta_p(\mathbf{x}_p)}{\beta_r(\mathbf{x}_r)} \prod_{d \in \mathcal{D}(p) \setminus \{r\}} n_{dp}(\mathbf{x}_d) \end{aligned} \tag{11.11}$$

It is now easy to see that equations (11.8)–(11.11) precisely define the conventional belief propagation algorithm of [33] applied on G_R.

Example 11.1. Consider a poset $R = \{r, s, t, u, v, w\}$ with the Hasse diagram G_R given in Figure 11.2(a). We will write the explicit form the GBP algorithm on both G_R and S_R.



(a) (b)

Figure 11.2: Graphical representations of Example 11.1
(a) Hasse diagram G_R, and (b) a minimal graphical representation S_R

GBP on G_R:

Messages:

$$m_{ru}(\mathbf{x}_u) = \frac{\sum_{\mathbf{x}_{r\setminus u}} \beta_r(\mathbf{x}_r)}{\beta_u(\mathbf{x}_u)}, \qquad m_{tv}(\mathbf{x}_v) = \frac{\sum_{\mathbf{x}_{t\setminus v}} \beta_t(\mathbf{x}_t)}{\beta_v(\mathbf{x}_v)}$$

$$m_{su}(\mathbf{x}_u) = \frac{\sum_{\mathbf{x}_{s\setminus u}} \beta_s(\mathbf{x}_s)\, m_{tv}(\mathbf{x}_v)}{\beta_u(\mathbf{x}_u)\, m_{vw}(\mathbf{x}_w)}, \qquad m_{sv}(\mathbf{x}_v) = \frac{\sum_{\mathbf{x}_{s\setminus v}} \beta_s(\mathbf{x}_s)\, m_{ru}(\mathbf{x}_u)}{\beta_v(\mathbf{x}_v)\, m_{uw}(\mathbf{x}_w)}$$

$$m_{uw}(\mathbf{x}_w) = \frac{\sum_{\mathbf{x}_{u\setminus w}} \beta_u\, m_{ru}\, m_{su}}{\beta_w(\mathbf{x}_w)}, \qquad m_{vw}(\mathbf{x}_w) = \frac{\sum_{\mathbf{x}_{v\setminus w}} \beta_v\, m_{tv}\, m_{sv}}{\beta_w(\mathbf{x}_w)}$$

Beliefs:

$$b_r(\mathbf{x}_r) = \beta_r(\mathbf{x}_r)\, m_{su}(\mathbf{x}_u)\, m_{vw}(\mathbf{x}_w), \qquad b_t(\mathbf{x}_t) = \beta_t(\mathbf{x}_t)\, m_{sv}(\mathbf{x}_v)\, m_{uw}(\mathbf{x}_w)$$

$$b_s(\mathbf{x}_s) = \beta_s(\mathbf{x}_s)\, m_{ru}(\mathbf{x}_u)\, m_{tv}(\mathbf{x}_v), \qquad b_u(\mathbf{x}_u) = \beta_u(\mathbf{x}_u)\, m_{ru}(\mathbf{x}_u)\, m_{su}(\mathbf{x}_u)\, m_{vw}(\mathbf{x}_w)$$

$$b_w(\mathbf{x}_w) = \beta_w(\mathbf{x}_w)\, m_{uw}(\mathbf{x}_w)\, m_{vw}(\mathbf{x}_w), \quad b_v(\mathbf{x}_v) = \beta_v(\mathbf{x}_v)\, m_{sv}(\mathbf{x}_v)\, m_{tv}(\mathbf{x}_v)\, m_{uw}(\mathbf{x}_w)$$

Note that this algorithm contains a 'loop': m_{su} depends on m_{vw}, which depends on m_{sv}, which depends on m_{uw}, which in turn depends on m_{su}. This means that

the above messages will not converge in finite time, even though, as is apparent from Figure 11.2(b), a junction tree does exist.

Compare the above algorithm with the following:

GBP on S_R:

Messages:

$$m_{ru}(\mathbf{x}_u) = \frac{\sum_{\mathbf{x}_{r\setminus u}} \beta_r(\mathbf{x}_r)}{\beta_u(\mathbf{x}_u)}, \qquad m_{tv}(\mathbf{x}_v) = \frac{\sum_{\mathbf{x}_{t\setminus v}} \beta_t(\mathbf{x}_t)}{\beta_v(\mathbf{x}_v)}$$

$$m_{su}(\mathbf{x}_u) = \frac{\sum_{\mathbf{x}_{s\setminus u}} \beta_s(\mathbf{x}_s)\, m_{tv}(\mathbf{x}_v)}{\beta_u(\mathbf{x}_u)}, \quad m_{sv}(\mathbf{x}_v) = \frac{\sum_{\mathbf{x}_{s\setminus v}} \beta_s(\mathbf{x}_s)\, m_{ru}(\mathbf{x}_u)}{\beta_v(\mathbf{x}_v)\, m_{uw}(\mathbf{x}_w)}$$

$$m_{uw}(\mathbf{x}_w) = \frac{\sum_{\mathbf{x}_{u\setminus w}} \beta_u\, m_{ru}\, m_{su}}{\beta_w(\mathbf{x}_w)}$$

Beliefs:

$$b_r(\mathbf{x}_r) = \beta_r(\mathbf{x}_r)\, m_{su}(\mathbf{x}_u), \qquad b_t(\mathbf{x}_t) = \beta_t(\mathbf{x}_t)\, m_{sv}(\mathbf{x}_v)\, m_{uw}(\mathbf{x}_w)$$

$$b_s(\mathbf{x}_s) = \beta_s(\mathbf{x}_s)\, m_{ru}(\mathbf{x}_u)\, m_{tv}(\mathbf{x}_v), \quad b_u(\mathbf{x}_u) = \beta_u(\mathbf{x}_u)\, m_{ru}(\mathbf{x}_u)\, m_{su}(\mathbf{x}_u)$$

$$b_w(\mathbf{x}_w) = \beta_w(\mathbf{x}_w)\, m_{uw}(\mathbf{x}_w), \qquad b_v(\mathbf{x}_v) = \beta_v(\mathbf{x}_v)\, m_{sv}(\mathbf{x}_v)\, m_{tv}(\mathbf{x}_v)\, m_{uw}(\mathbf{x}_w)$$

Notice that the above-mentioned loop is now broken, since m_{vw} does not exist anymore. This means that the messages in the above algorithm will converge after just one round of updates (performed in the correct order). This of course is not surprising; based on the discussion above, this algorithm is no more than the belief propagation algorithm on the junction tree of Figure 11.2(b). □

Chapter 12

Experimental Results

In the previous chapter we proved that the fixed points of GBP algorithms on any graphical representation for a poset R coincide with the solutions to the Kikuchi approximation problem of Section 9.2. We further argued that the algorithm on the minimal graph S_R has the smallest complexity per each iteration. Two important questions are not addressed in this document so far: 1) *how close are the Kikuchi approximations to the true marginals?* And 2) *how does the convergence behavior of the GBP algorithm on the minimal graph S_R compare to that on the full Hasse graph G_R?* In this section we address these questions with some simulation results.

We considered three simple loopy posets below. In each case, all the variables were binary. For each run of the experiment for a given poset, first we generated a random collection of potential functions $\{\alpha_r(\mathbf{x}_r)\}$, where each value $\alpha_r(\mathbf{x}_r)$ was chosen independently and uniformly in the interval $[0, 1]$. Next we calculated the product distribution $B(\mathbf{x}) = \prod_{r \in R} \alpha_r(\mathbf{x}_r)$ together with its true marginals $B_r(\mathbf{x}_r)$. The GBP algorithm of Section 11 then was run on each of the two graphs G_R and S_R for that poset. Further, two different schedules were incorporated to update the messages for each algorithm: parallel and serial. With the parallel schedule, all messages were updated together at each iteration. For the serial schedule, we update the messages one after another, in an order chosen so as to minimize the number of edges which are updated before their requisite set of edges have been updated. Each message is updated exactly once during each iteration. To ensure

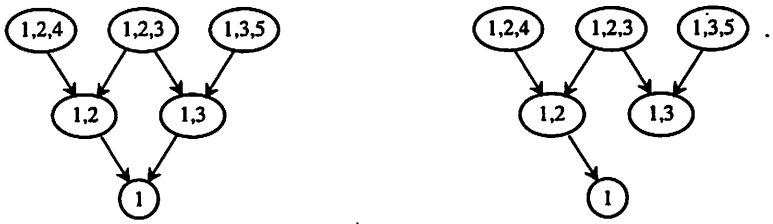
convergence of some algorithms we used damping in the update rule for the messages. The quantity w reported for each algorithm is the damping factor. In particular, we used $m_{pr}^{n+1}(\mathbf{x}_r) = w\, F(\{m^n\}) + (1-w)\, m_{pr}^n(\mathbf{x}_r)$, where m_{pr}^n is the message at iteration n, and $F(\{m^n\})$ is the 'pure' update rule of equation (11.7). The value of w is always between 0 and 1, with $w = 1$ corresponding to (11.7). For each case, we decreased w gradually to ensure that the algorithm converged.

For each poset, we report the savings in complexity per each iteration of GBP on the minimal graph compared to that on the Hasse graph. To compute these savings, we calculated the total arithmetic complexity, i.e. the number of additions, multiplications and divisions involved in update rules of (11.7), for both algorithms. Note that this is *not* simply the fraction of edges that are removed in forming the minimal graph, since the update rules for the messages that remain in the minimal graph are less complex than the ones on the Hasse graph.

To summarize the performance of each algorithm, at each iteration we calculated a special measure of distance between the beliefs $\{b_r\}$ and the true marginals $\{B_r\}$. We define a distance function $D(b_r, B_r) := \frac{\max_{\mathbf{x}_r} |b_r(\mathbf{x}_r) - B_r(\mathbf{x}_r)|}{\max_{\mathbf{x}_r} B_r(\mathbf{x}_r)}$ as the measure of distance from the belief b_r to the marginal B_r; this is a normalized maximum point-wise difference between the two distributions. The closer D is to 0, the closer the belief $b_r(\mathbf{x}_r)$ is to the true marginal $B_r(\mathbf{x}_r)$ at *all* configurations of $\mathbf{x}_r$. At each iteration we then calculate the *maximum distance* $\max_{r \in R} D(b_r, B_r)$, and the *mean distance* $\frac{1}{|R|} \sum_{r \in R} D(b_r, B_r)$. For each poset, the averages of these quantities over 200 runs are reported for each algorithm. The results are reported below:

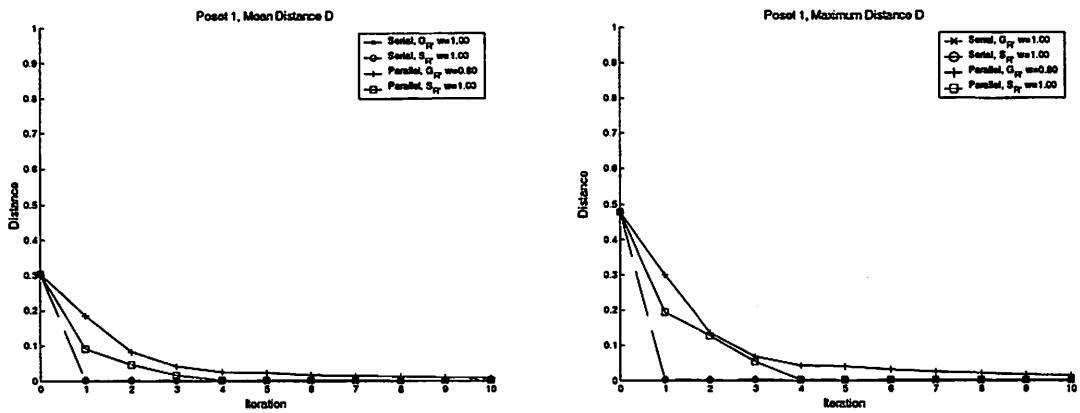
Poset 1: The Hasse diagram of this poset has one loop, but the minimal graph is loop-free. There is a saving of 35.7% per each iteration of GBP on the minimal graph compared to that on the Hasse graph. As expected, the Kikuchi approximations coincide with the true marginals in this loop-free case. The serial algorithms converge to the fixed-points after one iteration, because we use an optimal schedule for activating the messages. The parallel algorithm on the minimal graph takes four iteration

(equal to the girth of the graph). The parallel algorithm on the Hasse graph requires damping, and converges much more slowly. Note that in this case, the algorithm on the minimal graph both gives better performance iteration by iteration and has less complexity per iteration.



(a) (b)

Figure 12.1: Graphical representations of Posets 1
(a) The Hasse graph G_R, and (b) the minimal graph S_R.

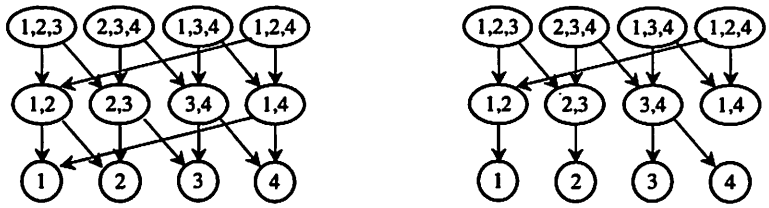


(a) (b)

Figure 12.2: Simulation results on Poset 1

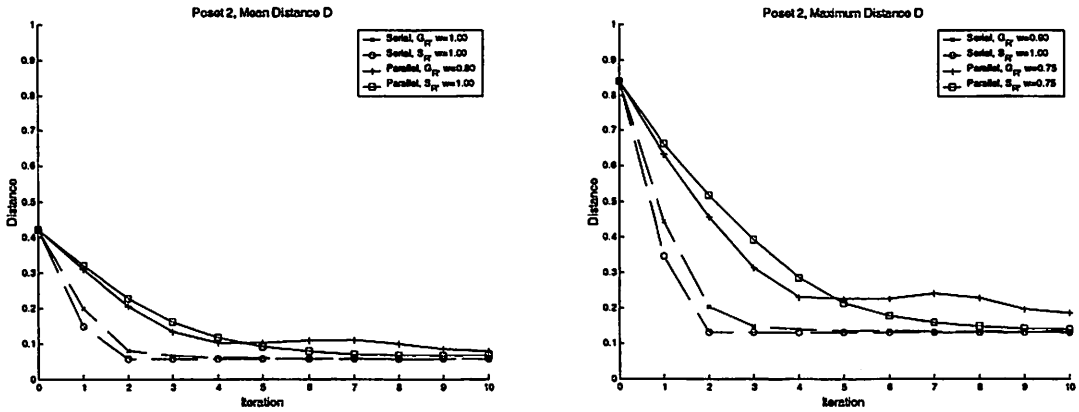
Poset 2: The Hasse diagram of this poset has five loops. All but one of these loops are broken in the minimal graph. There is a saving of 46.2% per each iteration of GBP on the minimal graph compared to that on the Hasse graph. The Kikuchi approximations are at an average distance of about 0.05 from the true marginals, while the worst estimates have distance of about 0.13. Again the serial algorithms converge very quickly, although the one on the Hasse graph requires a slight damping.

Comparing the parallel algorithms, the one on the minimal graph clearly outperforms the one on the full Hasse graph, even with equal damping factors.



(a) (b)

Figure 12.3: Graphical representations of Posets 2
(a) The Hasse graph G_R, and (b) the minimal graph S_R.



(a) (b)

Figure 12.4: Simulation results on Poset 2

Poset 3: The Hasse diagram of this poset has five loops, whereas the minimal graph has only two loops. There is a saving of 28.5% per each iteration of GBP on the minimal graph compared to that on the Hasse graph. The Kikuchi approximations are at an average distance of about 0.05 from the true marginals, while the worst estimates have distance of about 0.14. Once again the serial algorithms converge very quickly, without the need for damping. The parallel algorithm on the minimal graph again outperforms that on the full Hasse graph, the latter requiring a damping factor $w = 0.70$ to avoid oscillations.

At least for the simple posets considered here, the less complex GBP algorithm

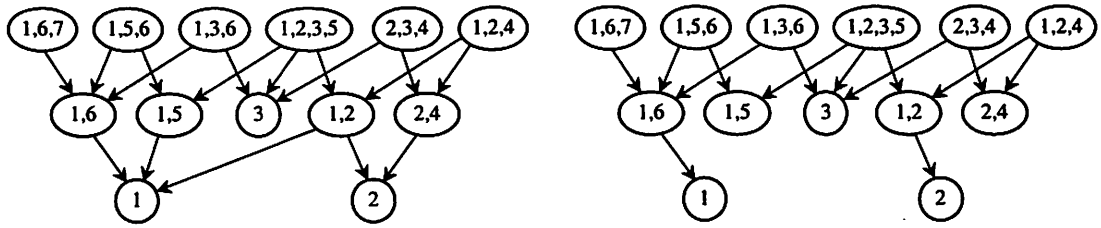


Figure 12.5: Graphical representations of Posets 3
(a) The Hasse graph G_R, and (b) the minimal graph S_R.

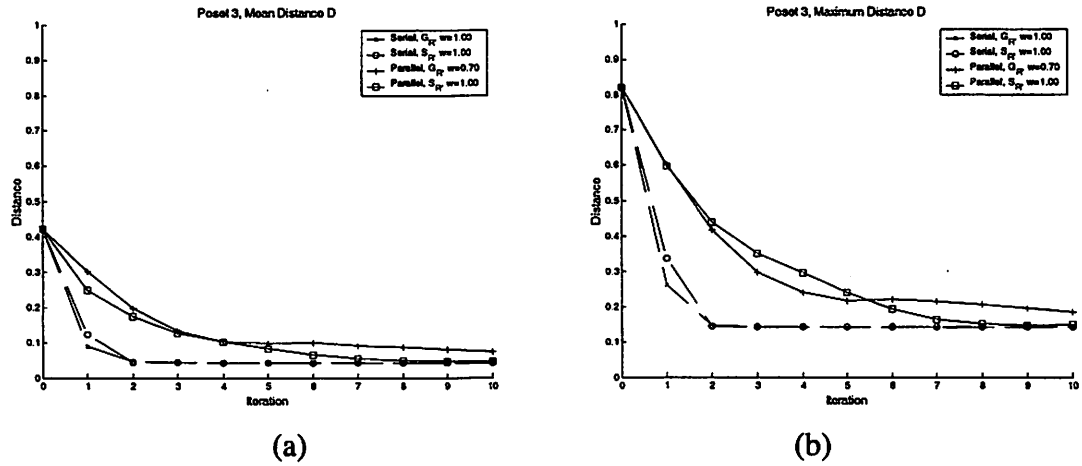


Figure 12.6: Simulation results on Poset 3

on the minimal graph, developed in this document, seems to perform better than the full GBP on the Hasse graph, especially with the parallel versions of the algorithm. Considering that each iteration of the algorithm on the minimal graph is less complex than that on the full Hasse graph, this suggests that there is considerable saving in the complexity to be gained by using the algorithm on the minimal graph.

12.1 Joint Decoding of LDPC Codes and Partial Response Channels

An interesting and promising configuration with application in magnetic recording is a partial response (PR) channel precoded by a LDPC code, as depicted in Figure 12.7, see e.g. [18]. In this section, we apply the Kikuchi approximation method to the problem of joint decoding of such system. (See also [52; 53] for practical considerations

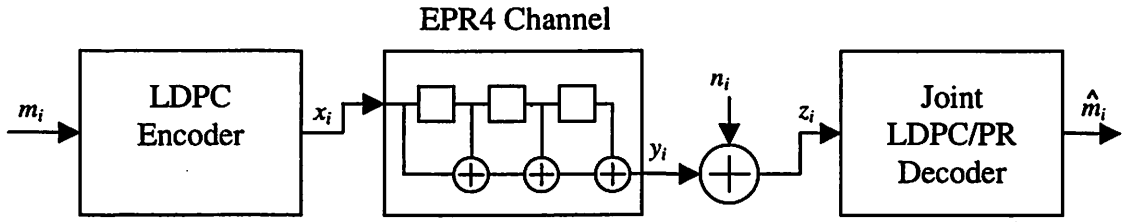


Figure 12.7: Block diagram for an LDPC/PR system

of the application of iterative decoding techniques in magnetic recording channels.)

As in Example 6.6 we identify the LDPC code by its parity check matrix $\mathbf{H} \in \mathrm{GF}(2)^{m \times n}$ where n is the block-length of the code, and m is the number of parity checks. Therefore a codeword $\mathbf{x} := (x_1, \cdots, x_n)$ satisfies $\mathbf{H} \cdot \mathbf{x} = \mathbf{0}$. The partial response channel is identified by a transfer polynomial $h(D) := \sum_{i=0}^{\nu} h_i D^i$, where ν is the degree of the channel. For example, the EPR4 channel depicted is identified by $h(D) = 1 + D - D^2 - D^3$. Therefore the output of the channel is related to its input by $y(D) = h(D)x(D)$, in the Z-transfer domain. We will assume an additive white Gaussian noise (AWGN) with variance σ_{n}^2. The objective is to find the maximum likelihood estimates of the transmitted code symbols x_i's given the noisy observations $\mathbf{z} = (z_1, \cdots, z_n)$.

It is clear that this problem can be described as MPD problem of Section 1.1. Let $P(\mathbf{x})$ denote the joint distribution of the codeword $\mathbf{x}$ given the observations. Then

$$P(\mathbf{x}) = \frac{1}{Z} \prod_{j=1}^{m} 1(H_j \cdot \mathbf{x} = 0) \prod_{i=1}^{n} p(z_i|\mathbf{x}) \tag{12.1}$$

$$= \frac{1}{Z} \prod_{j=1}^{m} 1(H_j \cdot \mathbf{x} = 0) \prod_{i=1}^{n} p_{\mathrm{n}}(z_i - \sum_{j=0}^{\nu} h_j x_{i-j}) \tag{12.2}$$

where H_j denotes the jth row of the parity check matrix $\mathbf{H}$, and p_{n} is the probability density of the noise. In particular, for the AWGN of variance σ_{n}^2, we have $p_{\mathrm{n}}(n) = ke^{-\frac{n^2}{2\sigma_{\mathrm{n}}^2}}$.

The best performing method discussed in [18] involves iteration between BCJR decoding of the PR channel, and the Gallager-Tanner decoding of the LDPC code. This corresponds to the standard BP algorithm performed on the graph of Figure 12.8,

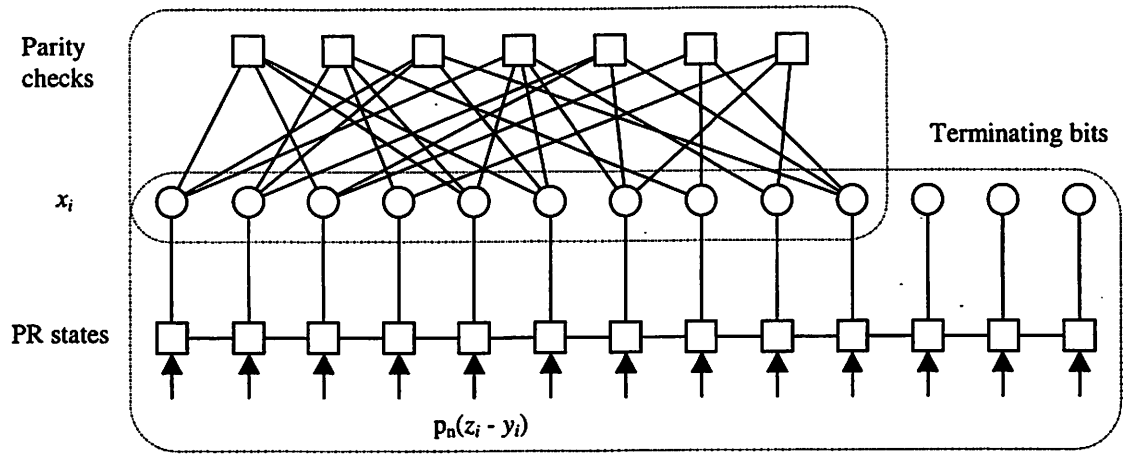


Figure 12.8: Graphical model for joint BP decoding of LDPC/PR problem

with three classes of nodes: the 'bit-nodes' corresponding to the bits x_i of the LDPC code; the 'check-nodes' corresponding to the parity checks of the LDPC code; and 'PR-state-nodes' corresponding to the states of the PR channel. The corresponding regions are, respectively, $\{i\}$; $\{j : \mathbf{H}_{i,j} = 1\}$; and $\{i - j : h_j \neq 0\}$, for all possible values of i.

To apply the Kikuchi approximation method for this problem, we used the poset obtained by the cluster variation method. Specifically, we appended all the intersection of the above regions to form the collection R of regions. Notice that good LDPC codes do not have small loops of size 4, so no two check-nodes can intersect in more than one index. However, the check-nodes can have nontrivial intersections with the PR-state-nodes. We considered a specific example from [18], with a rate 7/8 LDPC code with block-length $n = 495$, and with an EPR4 channel. The resulting poset had 495 bit-node regions (singletons), 62 check-node regions (each of size 24, since the LDPC code is regular with 24 bits per check), 495 PR-state-node regions (each of size 4, since the channel is EPR4), and a total of 659 nontrivial intersection regions, with nonzero overcounting factors. Of these 659 regions, 494 correspond to the intersections of neighboring PR-state-node regions (each of size 3). The remaining 165 regions (with sizes 2 or 3) are the regions that make the difference from the Bethe case.

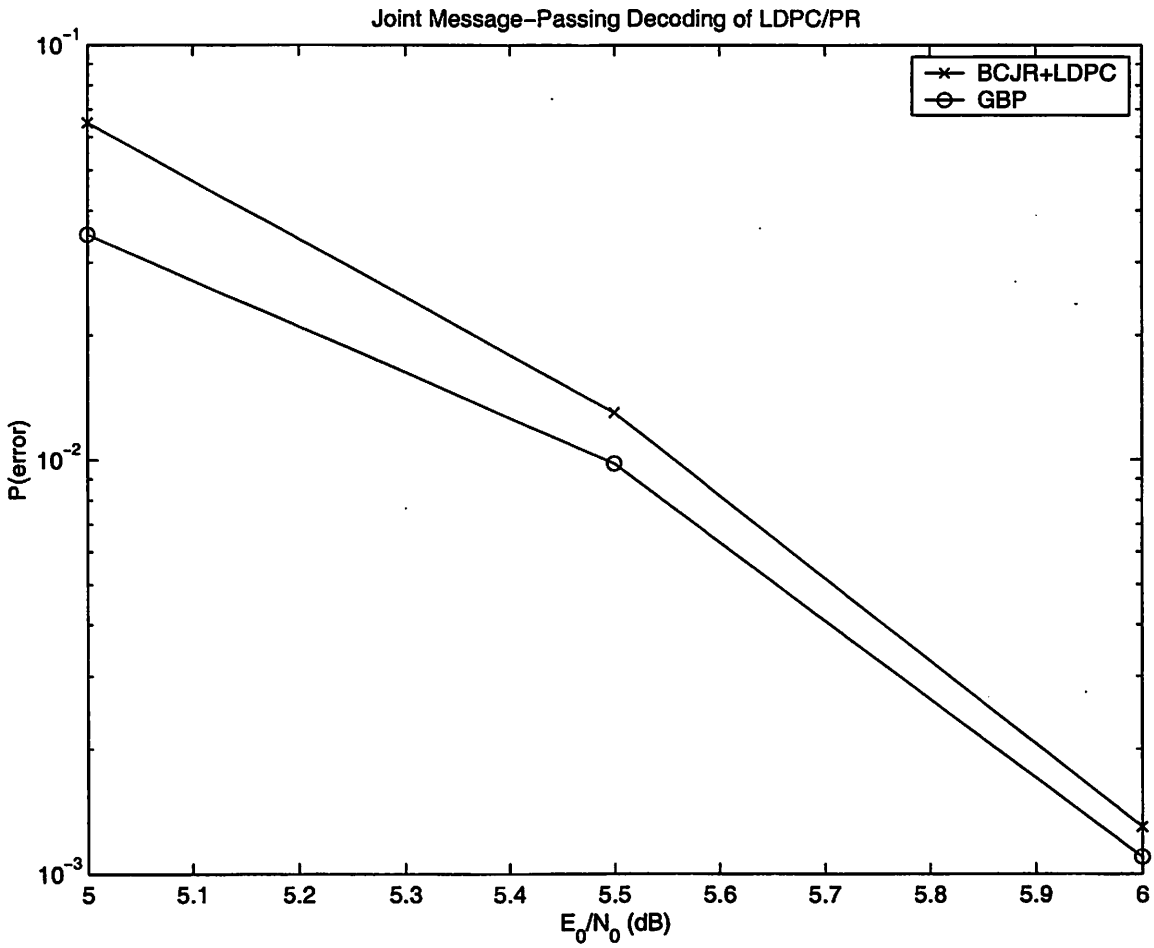


Figure 12.9: Simulation results for joint decoding of LDPC/PR
Each data point is the average over 20 simulation runs with 8 iterations of the corresponding algorithm.

The full Hasse diagram on this collection of regions has 1711 vertices, and 3973 edges. The minimal graph for this collection has 2951 edges. For comparison, the corresponding Hasse graph for the original problem, before addition of the intersection regions, which corresponds to the loopy BP algorithm, has 2476 edges.

Simulation results are reported in Figure 12.9 below. For each SNR and for each of the 20 simulation runs, we ran both the BCJR+LDPC algorithm of [18], and our GBP algorithm on the minimal graph for 8 full iterations.

These results suggest that, as expected, the GBP algorithm considered performs better than the BCJR+LDPC method. Our new technique appears to be particularly

well suited to the low SNR regime, which is the one that is most important for current magnetic recording applications.

Chapter 13

Summary and Discussion

Building on the original ideas introduced by Yedidia, Freeman and Weiss in [49], we have developed a general version of the Kikuchi approximation method, defined on a poset of regions.[1] It was shown in [49] that the fixed points of the very-successful loopy belief propagation algorithm can be viewed as the constrained minima of the Bethe free energy, which itself is an approximation to the variational free energy. Minimizing the exact form of this variational free energy results in the true marginals of the underlying product distribution. Then one expects that minimizing better approximations to the variational free energy would yield better approximations to the desired marginals. Kikuchi free energy approximations are such generalizations which are expected to improve on the Bethe free energy. We showed in Proposition 9.1 that the choice of the overcounting factors used in Kikuchi free energy is the only sensible one for the given class of the approximations.

Although we defined the Kikuchi problem on an arbitrary poset, we have discussed desirable connectivity and balance conditions on the poset, that are designed to improve the approximations. We have also introduced a set of conditions for the convexity of the Kikuchi problem. A well-known result, see e.g. [46], states that the loopy belief propagation converges if the underlying graph has a single loop. Our convexity conditions, when restricted to the Bethe case immediately imply this result,

[1]Other independent attempts at generalization of these ideas have also been made with similar results, most notably in [50] and [23], although we believe none are as comprehensive as ours. In particular, our discussion of graphical representations of Kikuchi regions is unique.

and more generally amount to a generalization of the single-loop condition.

We have introduced the concepts of graphical representation and the minimal graphical representations for a Kikuchi problem. These graphs serve as a basis for the iterative message-passing algorithms such as GBP to solve the Kikuchi approximation problem. The fixed points of such algorithms on any graphical representation of a Kikuchi problem are identical. Thus the minimal graphical representations, having the fewest number of edges (and hence messages), yield algorithms which are the least complex per each iteration. Our experimental results suggest that these minimal algorithms perform as well as the algorithms on the bigger graphical representations, even when compared on an iteration-to-iteration basis.

We have shown that, under some natural connectivity conditions, the minimal graph of a Kikuchi problem is a tree if and only if there is a junction tree on the collection of regions. We have further shown that except on a set of measure zero of the choices of the kernels, the exactness of the Kikuchi approximation corresponds precisely to the minimal graph being loop-free. Together these imply that the Kikuchi method can solve the MPD problem exactly precisely when the conventional junction tree method can also solve that problem exactly. Notice however that the main advantage of the Kikuchi method is expected in the approximating, 'loopy' domain of problems.

We have successfully applied the methods discussed in the document to the very real and practical problem of joint decoding of an LDPC code and a partial response channel. As expected, the results show an improvement over the best known conventional iterative algorithm, based on loopy belief propagation, although further investigation of the practicality of use of the Kikuchi method for this problem is required.

Appendices

Appendix A

Proofs from Part I

A.1 Proof of Theorem 2.1

Proof. Let f, g, x, and y be arbitrary elements of $\mathcal{F}, \mathcal{G}, \mathcal{X}$ and $\mathcal{Y}$ respectively. The proofs below consider all possible cases for the value of the μ on these atoms.

(2.3a): Symmetry is obvious, since $f \cap g = g \cap f$.

(2.3b): If $\mu(y) = 0$, then $\mu(f, g, x, y) = 0$ and if $\mu(y) \neq 0$, then $\mu(f, g, x, y) = \frac{\mu(f,y)\mu(g,x,y)}{\mu(y)}$ for all choices of f, g and x in $\mathcal{F}, \mathcal{G}$ and $\mathcal{X}$ respectively. In particular, ranging x over all atoms of $\mathcal{X}$ and summing the above equation yields the first part of result, since $\sum_{x \in \mathcal{A}(\mathcal{X})} \mu(f, g, x, y) = \mu(f, g, y)$ and $\sum_{x \in \mathcal{A}(\mathcal{X})} \mu(g, x, y) = \mu(g, y)$. Similarly, ranging g over all atoms of $\mathcal{G}$ and summing yields the second part of result.

(2.3c):

- $\mu(x) = 0$. Then from $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \mid \mathcal{X}$, we get $\mu(g, x) = 0$ for all g. Then from $\mathcal{F} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G} \vee \mathcal{X}$ we get $\mu(f, g, x, y) = 0$ for all f and y, and so we are done.

- $\mu(x) \neq 0$ and $\mu(g, x) = 0$. Then from $\mathcal{F} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G} \vee \mathcal{X}$ we get $\mu(f, g, x, y) = 0$ for all f and y, and so $\mu(f, g, x, y)\mu(x) = \mu(f, x)\mu(g, x, y) = 0$ and we are done.

- $\mu(x) \neq 0$ and $\mu(g,x) \neq 0$.Then from $\mathcal{F} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G} \vee \mathcal{X}$ we get

$$\mu(f,g,x,y) = \mu(f,g,x)\mu(g,x,y)/\mu(g,x) \tag{A.1}$$

  Also from $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \mid \mathcal{X}$ we have $\mu(f,g,x)/\mu(g,x) = \mu(f,x)/\mu(x)$. Replacing this into (A.1) we obtain $\mu(f,g,x,y) = \mu(g,x,y)\mu(f,x)/\mu(x)$ and we are done.

(2.3d):

- $\mu(x) = 0$. Then from $\mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X}$, we get $\mu(f,g,x,y) = 0$ for all f, g and y, and so we are done.

- $\mu(x) \neq 0$ and $\mu(x,y) = 0$. Then from $\mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X}$ we get $\mu(f,g,x,y) = \mu(f,g,x)\mu(x,y)/\mu(x) = 0$ for all f, g and y, and in particular $\mu(g,x,y) = 0$ and so we have the desired equality $\mu(f,g,x,y)/\mu(x) = \mu(f,x)\mu(g,x,y) = 0$.

- $\mu(x) \neq 0$ and $\mu(x,y) \neq 0$. We have

$$\mu(f,g,x) = \mu(f,x)\mu(g,x)/\mu(x) \qquad \text{since } \mathcal{F} \perp\!\!\!\perp \mathcal{G} \mid \mathcal{X} \tag{A.2}$$

$$\mu(f,g,x,y) = \mu(f,g,x)\mu(x,y)/\mu(x) \qquad \text{since } \mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X} \tag{A.3}$$

$$\mu(g,x)/\mu(x) = \mu(g,x,y)/\mu(x,y) \qquad \text{since, by (2.3b), } \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X} \tag{A.4}$$

  Replacing (A.4) into (A.2) and then into (A.3) we obtain

$$\mu(f,g,x,y) = \mu(f,x)\mu(g,x,y)/\mu(x).$$

(2.3e):

- $\mu(x) = 0$ and $\mu(g) = 0$. Then from $\mathcal{F} \vee \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G}$, we get $\mu(f,g,x,y) = 0$ and we are done.

- $\mu(x) = 0$ and $\mu(g) \neq 0$. From $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \mid \mathcal{X}$ we have $\mu(f,g,x) = 0$. Then from $\mathcal{F} \vee \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G}$, $\mu(f,g,x,y) = \mu(f,g,x)\mu(y,g)/\mu(g) = 0$ and we are done.

- $\mu(x) \neq 0$ and $\mu(g) = 0$. Then from $\mathcal{F} \vee \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G}$, we get both $\mu(f,g,x,y) = 0$ and $\mu(g,x,y) = 0$, so the desired equality hold:

$$\mu(f,g,x,y) = \mu(f,x)\mu(g,x,y)/\mu(x) = 0.$$

- $\mu(x) \neq 0$ and $\mu(g) \neq 0$. Then from $\mathcal{F} \vee \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G}$, we get $\mu(f,g,x,y) = \mu(f,g,x)\mu(g,y)/\mu(g)$. Also from $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \mid \mathcal{X}$ we have

$$\mu(f,g,x) = \mu(f,x)\mu(g,x)/\mu(x).$$

So we obtain the equality $\mu(f,g,x,y) = \mu(f,x)\mu(g,x)\mu(g,y)/(\mu(g)\mu(x))$. Finally, *decomposition* applied to $\mathcal{F} \vee \mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{G}$ yields $\mu(g,x)\mu(g,y)/\mu(g) = \mu(g,x,y)$. So we have proved $\mu(f,g,x,y) = \mu(f,x)\mu(g,x,y)/\mu(x)$ and this completes the proof.

(2.3f):

- $\mu(x) = 0$. Then from $\mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X}$, we have $\mu(f,g,x,y) = 0$ and we are done.

- $\mu(x) \neq 0$ and $\mu(x,y) = 0$. Then from $\mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X}$ we have $\mu(f,g,x,y) = \mu(f,g,x)\mu(x,y)/\mu(x)$ and so $\mu(f,g,x,y) = 0$. Also after applying (2.3b) to the above, we have $\mu(f,x,y) = \mu(f,x)\mu(x,y)/\mu(x) = 0$ and $\mu(g,x,y) = \mu(g,x)\mu(x,y)/\mu(x) = 0$. So we have the equality $\mu(f,g,x,y)\mu(x) = \mu(f,x,y)\mu(g,x) = \mu(f,x)\mu(g,x,y) = 0$ and we are done.

- $\mu(x) \neq 0$ and $\mu(x,y) \neq 0$. Then also $\mu(y) \neq 0$ or else from $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{X} \mid \mathcal{Y}$ we would have $\mu(x,y) = 0$. Then from $\mathcal{F} \perp\!\!\!\perp \mathcal{G} \vee \mathcal{X} \mid \mathcal{Y}$ we get $\mu(f,g,x,y) = \mu(f,y)\mu(g,x,y)/\mu(y)$, and also after (2.3b) to the above, we get $\mu(f,x,y)/\mu(x,y) = \mu(f,y)/\mu(y)$. Replacing the latter equation into the former we obtain

$$\mu(f,g,x,y) = \mu(g,x,y)\mu(f,x,y)/\mu(x,y) \qquad \text{(A.5)}$$

But from $\mathcal{F} \vee \mathcal{G} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{X}$ and by (2.3b) we have both $\mu(f,x,y)/\mu(x,y) = \mu(f,x)/\mu(x)$ and $\mu(g,x,y)/\mu(x,y) = \mu(g,x)/\mu(x)$. Replacing each of these into (A.5) we obtain

$$\mu(f,g,x,y) = \mu(g,x,y)\mu(f,x)/\mu(x)$$

and

$$\mu(f,g,x,y) = \mu(f,x,y)\mu(g,x)/\mu(x)$$

and we are done.

□

A.2 Proof of Correctness of Algorithm 3.1

In this appendix we give a proof for the correctness of the probabilistic junction tree algorithm 3.1. We will use a proof that parallels that given in [2]. We will need the following lemmas:

Lemma A.1. *Suppose there exists a junction tree with nodes corresponding to σ-fields $\{\mathcal{F}_1, \cdots, \mathcal{F}_M\}$. Then if f is a zero-measure atom of any of the $\mathcal{F}_i$'s, and $g \subset f$ is measurable in $\bigvee_{i=1}^{M} \mathcal{F}_i$, then $\mu(g) = 0$.*

Proof. Node i vacuously separates the empty subset of $\{1, \cdots, M\}$ from $\{1, \cdots, M\} \backslash \{i\}$. Thus $\{\emptyset, \Omega\} \perp\!\!\!\perp \bigvee_{\substack{j=1 \\ j \neq i}}^{M} \mathcal{F}_j \mid \mathcal{F}_i$. Thus by the definition of conditional independence, whenever $f \in \mathcal{A}(\mathcal{F}_i)$ has zero measure, all its subsets measurable in $\bigvee_{i=1}^{M} \mathcal{F}_i$ also have measure zero. □

Lemma A.2. (cf. Lemma A.1 in [2]) *Let $\mathcal{F}_1, \mathcal{F}_2$ and $\mathcal{F}_3$ be σ-fields such that $\mathcal{F}_1 \perp\!\!\!\perp \mathcal{F}_3 \mid \mathcal{F}_2$. Then for any partially-defined random variable $X \in \mathcal{F}_1$, the following equality holds:*

$$\mathbf{E}\Big[\mathbf{E}\big[X \big| \mathcal{F}_2\big] \Big| \mathcal{F}_3\Big] = \mathbf{E}\big[X \big| \mathcal{F}_3\big]$$

Proof. Let $Y = \mathbf{E}\big[X\big|\mathcal{F}_2\big]$. Then $\mathcal{A}_Y(\mathcal{F}_2) = \mathcal{A}'(\mathcal{F}_2) = \{b \in \mathcal{A}(\mathcal{F}_2) \ : \ \mu(b) \neq 0\}$ and for $b \in A_Y(\mathcal{F}_2)$, $Y(b) = \frac{1}{\mu(b)} \sum_{a \in A_X(\mathcal{F}_1)} X(a)\mu(a,b)$. Then, for any nonzero-measure atom $c \in \mathcal{A}(\mathcal{F}_3)$,

$$\begin{aligned}
\mathbf{E}\Big[\mathbf{E}\big[X\big|\mathcal{F}_2\big]\Big|\mathcal{F}_3\Big](c) &= \mathbf{E}\big[Y\big|\mathcal{F}_3\big](c) \\
&= \frac{1}{\mu(c)} \sum_{b \in A_Y(\mathcal{F}_2)} Y(b)\mu(b,c) \\
&= \frac{1}{\mu(c)} \sum_{b \in A_Y(\mathcal{F}_2)} \frac{1}{\mu(b)} \sum_{a \in A_X(\mathcal{F}_1)} X(a)\mu(a,b)\mu(b,c) \\
&= \frac{1}{\mu(c)} \sum_{b \in A_Y(\mathcal{F}_2)} \sum_{a \in A_X(\mathcal{F}_1)} X(a)\mu(a,b,c) \qquad \text{since } \mathcal{F}_1 \perp\!\!\!\perp \mathcal{F}_3 \,\big|\, \mathcal{F}_2 \\
&= \frac{1}{\mu(c)} \sum_{a \in A_X(\mathcal{F}_1)} X(a) \sum_{b \in A_Y(\mathcal{F}_2)} \mu(a,b,c) \\
&= \frac{1}{\mu(c)} \sum_{a \in A_X(\mathcal{F}_1)} X(a)\mu(a,c) \\
&= \mathbf{E}\big[X\big|\mathcal{F}_3\big](c)
\end{aligned}$$

where we have used the fact that $\mathcal{F}_1 \perp\!\!\!\perp \mathcal{F}_3 \,\big|\, \mathcal{F}_2$, so that $\sum_{b \in A_{\mathcal{F}_2}(Y)} \mu(a,b,c) = \sum_{b \in \mathcal{A}(\mathcal{F}_2)} \mu(a,b,c) = \mu(a,c)$. □

Lemma A.3. (cf. Lemma A.2 in [2]) *Let $\{\mathcal{F}_1, \cdots, \mathcal{F}_l\}$ and $\mathcal{F}$ be σ-fields such that $\{\mathcal{F}_1, \cdots, \mathcal{F}_l\}$ are mutually conditionally independent given $\mathcal{F}$. For each $i = 1, \cdots, l$, let X_i be a partially-defined random variable in $\mathcal{F}_i$. Then:*

$$\mathbf{E}\Big[\prod_{i=1}^{l} X_i \Big| \mathcal{F}\Big] = \prod_{i=1}^{l} \mathbf{E}\big[X_i\big|\mathcal{F}\big]$$

Proof. We shall proceed by induction. The statement is vacuous for $l = 1$. For $l = 2$, let $Y = \mathbf{E}\big[X_1X_2\big|\mathcal{F}\big]$. Then $\mathcal{A}_Y(\mathcal{F}) = \{f \in \mathcal{A}(\mathcal{F}) \ : \ \mu(f) \neq 0\}$. Also note that X_1X_2 is a partially-defined random variable in $\mathcal{F}_1 \vee \mathcal{F}_2$ with $\mathcal{A}_{X_1X_2}(\mathcal{F}_1 \vee \mathcal{F}_2) = \mathcal{A}_{X_1}(\mathcal{F}_1 \vee \mathcal{F}_2) \cap \mathcal{A}_{X_2}(\mathcal{F}_1 \vee \mathcal{F}_2)$, and that any atom of $\mathcal{F}_1 \vee \mathcal{F}_2$ can be written as $a \cap b$ for $a \in \mathcal{A}(\mathcal{F}_1)$ and $b \in \mathcal{A}(\mathcal{F}_2)$. Then for any $f \in \mathcal{A}_Y(\mathcal{F})$ we have:

$$
\begin{aligned}
Y(f) &= \frac{1}{\mu(f)} \sum_{\substack{(a\cap b)\in \mathcal{A}_{X_1X_2}(\mathcal{F}_1\vee\mathcal{F}_2) \\ a\in\mathcal{F}_1, b\in\mathcal{F}_2}} \cdot\ X_1(a)X_2(b)\mu(a,b,f) \\
&= \frac{1}{\mu(f)} \sum_{a\in\mathcal{A}_{X_1}(\mathcal{F}_1)} \sum_{b\in\mathcal{A}_{X_2}(\mathcal{F}_2)} X_1(a)X_2(b)\frac{\mu(a,f)\mu(b,f)}{\mu(f)} \\
&= \frac{1}{\mu(f)} \sum_{a\in\mathcal{A}_{X_1}(\mathcal{F}_1)} X_1(a)\mu(a,f)\frac{1}{\mu(f)} \sum_{b\in\mathcal{A}_{X_2}(\mathcal{F}_2)} X_2(b)\mu(b,f) \\
&= \mathbf{E}\big[X_1\big|\mathcal{F}\big](f)\mathbf{E}\big[X_2\big|\mathcal{F}\big](f)
\end{aligned}
$$

where we have used the fact that $\mathcal{F}_1 \perp\!\!\!\perp \mathcal{F}_2 \mid \mathcal{F}$, so $\frac{\mu(a,f)\mu(b,f)}{\mu(f)} = \mu(a,b,f)$.

For $l > 2$ assume inductively that the equality holds for $l-1$. Then:

$$
\begin{aligned}
\mathbf{E}\Big[\prod_{i=1}^{l} X_i\Big|\mathcal{F}\Big] &= \mathbf{E}\Big[X_1\prod_{i=2}^{l} X_i\Big|\mathcal{F}\Big] \\
&= \mathbf{E}\big[X_1\big|\mathcal{F}\big]\mathbf{E}\Big[\prod_{i=2}^{l} X_i\Big|\mathcal{F}\Big] && \text{since } \mathcal{F}_1 \perp\!\!\!\perp \bigvee_{i=2}^{l}\mathcal{F}_i \mid \mathcal{F} \\
&= \prod_{i=1}^{l}\mathbf{E}\big[X_i\big|\mathcal{F}\big] && \text{by induction hypothesis}
\end{aligned}
$$

□

Using Lemmas A.2 and A.3 above, we are in position to prove a version of the scheduling theorem 3.1 of [2] for our measure-theoretic framework.

Proof of correctness of Algorithm 3.1. We will show that if E_t is the schedule for activation of the nodes, (i.e. a directed edge $(i,j) \in E_t$ iff node i updates its message to its neighbor, j at time t) then the message from a node i to a neighboring node j is:

$$
Y_{i,j}(t) = \mathbf{E}\Big[\prod_{k\in K_{i,j}(t)} X_k\Big|\mathcal{F}_j\Big], \tag{A.6}
$$

where $K_{i,j}(t)$ is a subset of the nodes defined recursively by:

$$
K_{i,j}(t) = \begin{cases} \emptyset & \text{if } t = 0, \\ K_{i,j}(t-1) & \text{if } (i,j)\notin E_t, \\ \{i\}\bigcup_{l\in N_{i,j}} K_{l,i}(t-1) & \text{if } (i,j)\in E_t \end{cases}
$$

We will prove this by induction on t. Case $t = 0$ is clear from the initialization. Now let $t > 0$ and assume that (A.6) above holds for $t-1$. We can also assume that the $(i,j) \in E_t$ so the message $Y_{i,j}$ is being updated at time t. Then:

$$
\begin{aligned}
Y_{i,j}(t) &= \mathbf{E}\Big[X_i \prod_{l \in N_{i,j}} Y_{l,i} \Big| \mathcal{F}_j\Big] \\
&= \mathbf{E}\Big[X_i \prod_{l \in N_{i,j}} \mathbf{E}[\prod_{k \in K_{l,i}(t-1)} X_k | \mathcal{F}_i] \Big| \mathcal{F}_j\Big] && \text{by induction} \\
&= \mathbf{E}\Big[\mathbf{E}[X_i \prod_{l \in N_{i,j}} \prod_{k \in K_{l,i}(t-1)} X_k | \mathcal{F}_i] \Big| \mathcal{F}_j\Big] && \text{by the j.t. property and Lemma A.3} \\
&= \mathbf{E}\Big[X_i \prod_{l \in N_{i,j}} \prod_{k \in K_{l,i}(t-1)} X_k \Big| \mathcal{F}_j\Big] && \text{by the j.t. property and Lemma A.2} \\
&= \mathbf{E}\Big[\prod_{k \in K_{i,j}(t)} X_k \Big| \mathcal{F}_j\Big] && \text{by definition of } K_{i,j}(t)
\end{aligned}
$$

Indeed $K_{i,j}(t)$ above is the set of all the nodes whose 'information' has reached the edge (i,j) by time t. Similarly, with $J_i(t) := \{i\} \bigcup_{j \in N_i} K_{j,i}(t)$, $J_i(t)$ is the collection of all the nodes whose 'information' has reached a node i by time t. As in [2], we define a *message trellis* up to time t, which is an $M \times t$ directed graph, where for any $i, j \in \{1, \cdots, M\}$ and $n < t$, $i(n)$ is always connected to $i(n+1)$, and $i(n)$ is connected to $j(n+1)$ iff $(i,j) \in E_n$. It follows that we will have $J_i(t) = \{1, \cdots, M\}$ when there is a path from every initial node (i.e. at $t = 0$) in the trellis to the node $i(t)$. Then, since the tree has finite diameter, *any* infinite schedule that activates all the edges infinitely many times has a finite sub-schedule, say of length t_0 such that

$J_i(t_0) = \{1, \cdots, M\}$ for all i. At that time we have:

$$\begin{aligned}
\mathbf{E}\Big[X_i \prod_{j\in N_i} Y_{j,i}(t_0)\Big|\mathcal{F}_i\Big] &= \mathbf{E}\Big[X_i \prod_{j\in N_i} \mathbf{E}\Big[\prod_{k\in K_{j,i}(t_0)} X_k\Big|\mathcal{F}_i\Big]\Big|\mathcal{F}_i\Big] && \text{by (A.6)} \\
&= \mathbf{E}\Big[\mathbf{E}\Big[X_i \prod_{j\in N_i}\prod_{k\in K_{j,i}(t_0)} X_k\Big|\mathcal{F}_i\Big]\Big|\mathcal{F}_i\Big] && \text{by the j.t. property and Lemma A.3} \\
&= \mathbf{E}\Big[X_i \prod_{j\in N_i}\prod_{k\in K_{j,i}(t_0)} X_k\Big|\mathcal{F}_i\Big] \\
&= \mathbf{E}\Big[\prod_{k\in J_i(t_0)} X_k\Big|\mathcal{F}_i\Big] && \text{by defn. of } J_i(t) \\
&= \mathbf{E}\Big[\prod_{k=1}^{M} X_k\Big|\mathcal{F}_i\Big]
\end{aligned}$$

This completes the proof of correctness of Algorithm 3.1. □

Appendix B

Proofs from Part II

B.1 Proof of Theorem 10.9

Proof. Suppose (2) does not hold, so for some $s \subseteq [N]$ such that $\cup_{i\in s}\mathcal{F}(i)$ is connected, $\sum_{r\in\cup_{i\in s}\mathcal{F}(i)} k_r \neq 1$. We will choose kernels $\{\alpha_r, r \in R\}$ so that (1) will be violated. Specifically, for each r we choose $\alpha_r(\mathbf{x}_r) = \prod_{j\in r\setminus s} 1(x_j = 0)\big(\prod_{i\in r\cap s} 1(x_i = 0) + \prod_{i\in r\cap s} 1(x_i = 1)\big)$. Under the product distribution $B(\mathbf{x}) = \frac{1}{Z}\prod_{r\in R}\alpha_r(\mathbf{x}_r)$, for each $j \in [N]\setminus s$, x_j will have zero probability of taking a value other than 0, i.e. random variable x_j is deterministic and will have zero entropy. On the other hand, since $\cup_{i\in s}\mathcal{F}(i)$ is connected, for each pair $i, j \in s$ the probability that $x_i \neq x_j$ is zero; in fact there is exactly a probability of 0.5 that $\mathbf{x}_s = (0, 0, \cdots, 0)$ and a probability of 0.5 that $\mathbf{x}_s = (1, 1, \cdots, 1)$. Therefore random variables x_i for $i \in s$ are redundant, and for all $t \subseteq [N]$ s.t. $s \cap t \neq \emptyset$, $H_t(B_t) = 1$(bit). Now by independence of $\mathbf{x}_s$ and $\mathbf{x}_{[N]\setminus s}$ we have

$$H(B) = H_s(B_s) + H_{[N]\setminus s}(B_{[N]\setminus s}) = H_s(B_s) = 1$$

On the other hand,

$$\sum_{r\in R} k_r\, H_r(B_r) = \sum_{r\in\cup_{i\in s}\mathcal{F}(i)} k_r\, H_r(B_r) + \sum_{r\in R\setminus\cup_{i\in s}\mathcal{F}(i)} k_r\, H_r(B_r) = \sum_{r\in\cup_{i\in s}\mathcal{F}(i)} k_r \cdot 1 + 0 \neq 1$$

so that $H(B) \neq \sum_{r\in R} k_r\, H_r(B_r)$. Therefore we have shown that (1) implies (2).

Now suppose that (2) holds. We will show that (3) must hold, using induction

on the *lexicographical order on the strings of the decreasingly-sorted cardinalities of elements of* S defined on all $S \subseteq 2^{[N]}$; we clarify this ordering using an example:

Suppose $N = 12$, and $S_1 = \{\{10,11,0\}, \{1,\cdots,10\}\}$, $S_2 = \{\{1,2,3,4,5\},\{3,4,5,6\},\{6,7,8,9,10\}\}$, $S_3 = \{\{1\},\cdots,\{7\}\}$ and $S_4 = \{\{6\},\{7\},\{8\}\}$. Then the 'sorted strings of the cardinalities' are $str(S_1) = [10.3]$, $str(S_2) = [5.5.4]$, $str(S_3) = [1.1.1.1.1.1.1]$ and $str(S_4) = [1.1.1]$, so that $str(S_1) >_{lex} str(S_2) >_{lex} str(S_3) >_{lex} str(S_4)$.

It is clear that if all $s \in S$ were singletons, so that $str(S) = [1.\cdots.1]$, then (3) is equivalent to (2). Now suppose $S = \{s_1,\cdots,s_n\}$, and $|s_n| \geq 2$. We split s_n as the disjoint union of t_1 and t_2, i.e. $s_n = t_1 \cup t_2$ and $t_1 \cap t_2 = \emptyset$, so that $0 < |t_1|, |t_2| < |s_n|$. Define $T_1 := \{s_1,\cdots,s_{n-1},t_1\}, T_2 := \{s_1,\cdots,s_{n-1},t_2\}$ and $T_{12} := \{s_1,\cdots,s_{n-1},t_1,t_2\}$. Clearly now, with the above lexicographical order, $str(T_1), str(T_2)$ and $str(T_{12})$ each are smaller than $str(S)$. Furthermore, $\cup_{s\in T_1}\mathcal{F}(s), \cup_{s\in T_2}\mathcal{F}(s)$ and $\cup_{s\in T_{12}}\mathcal{F}(s)$ are each connected, since they all contain $\cup_{s\in S}\mathcal{F}(s)$ as an up-set. But

$$\begin{aligned}
\sum_{r\in\cup_{s\in T_{12}}\mathcal{F}(s)} k_r &= \sum_{r\in\cup_{s\in S}\mathcal{F}(s)} k_r + \sum_{r\in\mathcal{F}(t_1)\setminus\cup_{s\in S}\mathcal{F}(s)} k_r + \sum_{r\in\mathcal{F}(t_2)\setminus\cup_{s\in S}\mathcal{F}(s)} k_r &&= 1\\
\sum_{r\in\cup_{s\in T_1}\mathcal{F}(s)} k_r &= \sum_{r\in\cup_{s\in S}\mathcal{F}(s)} k_r + \sum_{r\in\mathcal{F}(t_1)\setminus\cup_{s\in S}\mathcal{F}(s)} k_r &&= 1\\
\sum_{r\in\cup_{s\in T_2}\mathcal{F}(s)} k_r &= \sum_{r\in\cup_{s\in S}\mathcal{F}(s)} k_r + \sum_{r\in\mathcal{F}(t_2)\setminus\cup_{s\in S}\mathcal{F}(s)} k_r &&= 1
\end{aligned}$$

where we have used induction hypothesis to conclude that each sum must be equal to 1. Using the above three equations we get $\sum_{r\in\cup_{s\in S}\mathcal{F}(s)} k_r = 1$. This completes the inductive proof.

Next suppose that (3) holds for a choice of factors $\{k_r, r \in R\}$. First note that choosing $S = \{r\}$ for each $r \in R$ we get equations (9.9), implying that $\{k_r, r \in R\}$ must in fact be the same as the (Möbius) overcounting factors, $\{c_r, r \in R\}$. Suppose now that S_R, the minimal graph of R has a loop. Let $L \subseteq R$ be a loop of S_R, and let $L_0 \subset R$ be the collection of minimal regions of L, i.e. every $r \in L$ contains some

$r_0 \in L_0$, and that no region in L_0 properly contains another region in L_0. Therefore $\mathcal{F}(L_0)$ contains loop L of S_R. We now claim that one can find a loop L with minimal regions L_0 such that for any proper subset $L_0' \subset L_0$, $\mathcal{F}(L_0')$ is loop-free. This is because if $\mathcal{F}(L_0')$ contains a loop L' for a proper subset L_0' of L_0, then we can choose L_0' in place of L_0, and $\mathcal{F}(L_0')$ still has a loop L'. But $|L_0|$ is finite and $|L_0'| < |L_0|$, so this process must end, yielding a loop L with collection L_0 of minimal regions, with the desired property. Further note that L_0 cannot have cardinality 1, since if $L_0 = \{r_0\}$ for some $r_0 \in R$, then all the edges of the Hasse diagram that terminate in r_0 and participate in the loop L would be EER; all but one of these edges would be removed in S_R, therefore r_0 cannot be part of a loop.

Therefore for each region $r \in \mathcal{F}(L_0)$, $S_{\mathcal{F}(r)}$ is loop-free. Noting that the overcounting factor c_r only depends on $\mathcal{F}(r)$, and using Lemma 10.6, $c_r = 1 - |\mathcal{P}_{S_R}(r)|$. Then, as before, the sum $\sum_{r\in\mathcal{F}(L_0)} c_r$ can be rewritten as the difference between the number of vertices and the number of edges of $S_{\mathcal{F}(L_0)}$. But $S_{\mathcal{F}(L_0)}$ has at least one loop, therefore it has at least as many edges as vertices. Therefore $\sum_{r\in\mathcal{F}(L_0)} c_r \leq 0$ and cannot be equal to 1. This would contradict (3), and hence S_R must be loop-free.

Now suppose that S_R is loop-free. Then by Proposition 10.4, S_R is a junction tree. Choose then $\{k_r\}$ to be equal to the overcounting factors $\{c_r\}$, so that $k_r = 1 - |\mathcal{P}_{S_R}(r)|$. Then by standard results on the junction trees, any distribution $B(\mathbf{x})$ that decomposes on the junction tree S_R, factors as $\prod_{r\in R} B_r(\mathbf{x}_r)^{k_r}$ (see [10]). From this (1) follows immediately. This completes the proof of the theorem. □

B.2 Proof of Theorem 10.11

Proof. We define the set T and error function $f(\cdot)$ similar to the proof of Theorem 10.10, with the convention that $0\log(0) = 0$. Here, however, f is a function on $\mathbb{R}_+^l$ with $l = \sum_{r\in R_0}(q_r - |Z_r|)$. Once again, $f(\cdot)$ is seen to be analytic on $\mathbb{R}_+^l$.

The argument to show that f is not identically zero proceeds exactly as before. It

only remains to show that Theorem 10.9 still holds for the restricted case imposed by $\{Z_r\}$. The proof of Theorem 10.9 remains unchanged with the following exception: in the first part, to prove that "not (2) implies not (1)", for each r we choose $\alpha_r(\mathbf{x}_r) = \prod_{j\in[N]\setminus(r\cap s)} 1(x_j = \tilde{x}_j)\big(\prod_{i\in r\cap s} 1(x_i = \tilde{x}_i^1) + \prod_{i\in r\cap s} 1(x_i = \tilde{x}_i^2)\big)$, where $\tilde{\mathbf{x}}_s^1, \tilde{\mathbf{x}}_s^2$ and $\tilde{\mathbf{x}}_{[N]\setminus s}$ are chosen according to (10.4). The fact that these α's are consistent with $\{Z_r\}$ is then guaranteed by (10.4).

Therefore f is not identically zero on its domain $\mathbb{R}_+^l$ of kernels consistent with $\{Z_r\}$, and hence the Lebesgue measure of the set T is zero. □

B.3 Proof of Proposition 11.1

Proof. Given $t \in R, s \in \mathcal{P}_G(t)$, by definition of the overcounting factors

$$\sum_{u\in\mathcal{F}(t)} c_u = 1 \quad \text{and} \quad \sum_{u\in\mathcal{F}(s)} c_u = 1$$

$$\text{Therefore} \sum_{u\in\mathcal{F}(t)\setminus\mathcal{F}(s)} c_u = 0 \tag{B.1}$$

Now if $\{b_r,\, r \in R\} \in \Delta_R^K$, then $\forall u \in \mathcal{F}(t)$, $\sum_{\mathbf{x}_{u\setminus t}} b_u(\mathbf{x}_u) = b_t(\mathbf{x}_t)$. Therefore

$$\begin{aligned}
\mathrm{EC}'_{(s\to t)}(\{b_r\,,\, r \in R'\}) &= \sum_{u\in\mathcal{F}(t)\setminus\mathcal{F}(s)} c_u \sum_{\mathbf{x}_{u\setminus t}} b_u(\mathbf{x}_u) \\
&= \sum_{u\in\mathcal{F}(t)\setminus\mathcal{F}(s)} c_u\, b_t(\mathbf{x}_t) \\
&= b_t(\mathbf{x}_t) \sum_{u\in\mathcal{F}(t)\setminus\mathcal{F}(s)} c_u \\
&= 0
\end{aligned}$$

Hence $\{b_r,\, r \in R'\} \in \Delta'_R$, and we have proven that $\Delta_R^K\big|_{R'} \subseteq \Delta'_R$.

Now conversely suppose that $\{b_r,\, r \in R'\} \in \Delta'_R$. We will show by induction on depth function $d(t)$ of region $t \in R$ (w.r.t. poset R, and not graph G) that for all $s \in \mathcal{A}(t)$, $\sum_{\mathbf{x}_{s\setminus t}} b_s(\mathbf{x}_s) = b_t(\mathbf{x}_t)$. The statement holds vacuously for the maximal

regions, since these regions cannot have parents. Now let t be a region with depth $d(t) = l > 0$ and let $\mathcal{P}_G(t) = \{s_1, \cdots, s_m\}$. For each pair s_i and s_j of parents of t in G, consider the following cases on $\mathcal{A}(s_i) \cap \mathcal{A}(s_j)$:

- Suppose $\mathcal{A}(s_i) \cap \mathcal{A}(s_j) = \emptyset$. Then, because $\{b_r, r \in R'\} \in \Delta'_R$, we have

$$\sum_{u \in \mathcal{F}(t) \setminus \mathcal{F}(s_i)} c_u \sum_{\mathbf{x}_{u \setminus t}} b_u(\mathbf{x}_u) = 0$$
$$\sum_{u \in \mathcal{F}(t) \setminus \mathcal{F}(s_j)} c_u \sum_{\mathbf{x}_{u \setminus t}} b_u(\mathbf{x}_u) = 0$$

  Subtracting one from another we obtain the following equality:

$$\sum_{u \in \mathcal{F}(s_i)} c_u \sum_{\mathbf{x}_{u \setminus t}} b_u(\mathbf{x}_u) = \sum_{u \in \mathcal{F}(s_j)} c_u \sum_{\mathbf{x}_{u \setminus t}} b_u(\mathbf{x}_u) \tag{B.2}$$

  Since $d(s_i)$ and $d(s_j)$ are each no larger than $l - 1$, by induction hypothesis we have

$$\forall u \in \mathcal{F}(s_i), \quad \sum_{\mathbf{x}_{u \setminus s_i}} b_u(\mathbf{x}_u) = b_{s_i}(\mathbf{x}_{s_i})$$
$$\forall u \in \mathcal{F}(s_j), \quad \sum_{\mathbf{x}_{u \setminus s_j}} b_u(\mathbf{x}_u) = b_{s_j}(\mathbf{x}_{s_j})$$

  Replacing these in (B.2) we obtain

$$\sum_{\mathbf{x}_{s_i \setminus t}} b_{s_i}(\mathbf{x}_{s_i}) \sum_{u \in \mathcal{F}(s_i)} c_u = \sum_{\mathbf{x}_{s_j \setminus t}} b_{s_j}(\mathbf{x}_{s_j}) \sum_{u \in \mathcal{F}(s_j)} c_u$$

  But by definition of the overcounting factors, $\sum_{u \in \mathcal{F}(s_i)} c_u = \sum_{u \in \mathcal{F}(s_j)} c_u = 1$, so that $\sum_{\mathbf{x}_{s_i \setminus t}} b_{s_i}(\mathbf{x}_{s_i}) = \sum_{\mathbf{x}_{s_j \setminus t}} b_{s_j}(\mathbf{x}_{s_j})$.

- Suppose $u \in \mathcal{A}(s_i) \cap \mathcal{A}(s_j)$. Then again by induction hypothesis, $\sum_{\mathbf{x}_{u \setminus s_i}} b_u(\mathbf{x}_u) = b_{s_i}(\mathbf{x}_{s_i})$ and $\sum_{\mathbf{x}_{u \setminus s_j}} b_u(\mathbf{x}_u) = b_{s_j}(\mathbf{x}_{s_j})$. Therefore $\sum_{\mathbf{x}_{s_i \setminus t}} b_{s_i}(\mathbf{x}_{s_i}) = \sum_{\mathbf{x}_{u \setminus t}} b_u(\mathbf{x}_u) = \sum_{\mathbf{x}_{s_j \setminus t}} b_{s_j}(\mathbf{x}_{s_j})$

We can therefore show that for all pairs s_i and s_j of parents of t in G, $\sum_{\mathbf{x}_{s_i \setminus t}} b_{s_i}(\mathbf{x}_{s_i}) = \sum_{\mathbf{x}_{s_j \setminus t}} b_{s_j}(\mathbf{x}_{s_j}) = b'_t(\mathbf{x}_t)$ for a unique function $b'_t(\mathbf{x}_t)$. Now if $t \notin R'$, we define

$b_t(\mathbf{x}_t) := b'_t(\mathbf{x}_t)$. If $t \in R'$, using the fact that $\{b_r,\, r \in R'\} \in \Delta'_R$, we have

$$\begin{aligned} & \sum_{u \in \mathcal{F}(t)\backslash\mathcal{F}(s_i)} c_u \sum_{\mathbf{x}_{u\backslash t}} b_u(\mathbf{x}_u) = 0 \\ \Longrightarrow \quad & c_t\, b_t(\mathbf{x}_t) + \sum_{u \in \mathcal{A}(t)\backslash\mathcal{F}(s_i)} c_u\, b'_t(\mathbf{x}_t) = 0 \\ \Longrightarrow \quad & b_t(\mathbf{x}_t) = b'_t(\mathbf{x}_t) \end{aligned}$$

since by (B.1), $c_t + \sum_{u \in \mathcal{A}(t)\backslash\mathcal{F}(s_i)} c_u = 0$, and $c_t \neq 0$.

So we have shown that $\sum_{\mathbf{x}_{s_i\backslash t}} b_{s_i}(\mathbf{x}_{s_i}) = b_t(\mathbf{x}_t)$ for all $s_i \in \mathcal{P}_G(t)$. But G is a graphical representation of Δ^K_R, therefore by argument similar to those of Proposition 10.3 for each $(s \rightarrow t) \in \mathcal{E}(G_R)\backslash\mathcal{E}(G)$, the edge-constraint $\sum_{\mathbf{x}_{s\backslash t}} b_s(\mathbf{x}_s) = b_t(\mathbf{x}_t)$ is implied by the edge-constraints of those edges of G at the same, or at a lower, depth. Specifically, there must be a path in G between u and t for each $u \in \mathcal{A}(t)$, consisting only of vertices that contain t, or else consistency between b_u and b_t could not be implied by the edge-constraints of G. But any vertex that contains t must have a depth less than t (remember that we are using the depth function on R, and not on G: a region containing t *could* have a G-depth higher than that of t.) Therefore all the G-edges in this path have depths no more than $l = d(t)$ and can be used in our inductive argument. Together, they imply the consistency between u and t, i.e. $\sum_{\mathbf{x}_{u\backslash t}} b_u(\mathbf{x}_u) = b_t(\mathbf{x}_t)$.

Therefore we have found the desired extension $\{b_r,\, r \in R\} \in \Delta_R$, and so $\Delta'_R \subseteq \Delta^K_R\big|_{R'}$. This proves that $\Delta'_R = \Delta^K_R\big|_{R'}$ as claimed. □

Appendix C

Pairwise Partitions vs. Valid Partitions

In Section 3.2 we defined valid partitions of $\{1, \cdots, M\}\backslash\{i\}$ with respect to a node i, and showed that a finest valid partition, P_i exists. In this appendix we derive similar results with *pairwise partitions*. The significance of this discussion is that, in the case with an unsigned measure μ, as we will describe, the finest valid partitions coincide with the finest pairwise partitions, which are much simpler to compute.

Definition C.1. A *pairwise partition* of $\{1, \cdots, M\}\backslash\{i\}$ with respect to a node i is a partition $\{p_1, \cdots, p_l\}$ of $\{1, \cdots, M\}\backslash\{i\}$ (i.e. $\bigcup_{j=1}^{l} p_j = \{1, \cdots, M\}\backslash\{i\}$ and $p_j \cap p_k = \emptyset$ for $j \neq k$) such that $\mathcal{F}_{p_j}$'s are pairwise conditionally independent, given $\mathcal{F}_i$.

□

Proposition C.1. *$\forall i \in \{1, \cdots, M\}$, there is a* finest pairwise partition *w.r.t. i, which we shall denote by Q_i, such that every other pairwise partition w.r.t. i is a coarsening of Q_i.*

Proof. Similar to the proof of Proposition 3.2. □

Proposition C.2. *If the measure μ is unsigned, $P_i = Q_i$, i.e. the finest valid partition w.r.t. a node i coincides with the finest pairwise partition w.r.t. i.*

Proof. Let $P_i := \{c_1, \cdots, c_{l'}\}$ and $Q_i := \{d_1, \cdots, d_l\}$ and suppose that $P_i \neq Q_i$. Then for some $r \in \{1, \cdots, M\}$, there must exist distinct $j_1, \cdots, j_s \in \{1, \cdots, l\}$ with $s \geq 2$ such that $c_r = \bigcup_{k=1}^{s} d_{j_k}$. Now let $\mathbf{G}$ be a junction tree compatible with P_i at i. Let $c' = \{1, \cdots, M\} \backslash \{i\} \backslash c_r$.

Assume WLOG that $\mathcal{F}_{d_{j_1}}$ is a leaf node of $\mathbf{G}$ lying in c_r and that $\mathcal{F}_{d_{j_2}}$ is its neighbor. Then $\mathcal{F}_{d_{j_1}} \perp\!\!\!\perp \bigvee_{k=3}^{s} \mathcal{F}_{d_{j_k}} \vee \mathcal{F}_i \vee \mathcal{F}_{c'} \mid \mathcal{F}_{d_{j_2}}$. This implies by weak union that $\mathcal{F}_{d_{j_1}} \perp\!\!\!\perp \bigvee_{k=3}^{s} \mathcal{F}_{d_{j_k}} \vee \mathcal{F}_{c'} \mid \mathcal{F}_{d_{j_2}} \vee \mathcal{F}_i$. But also $\mathcal{F}_{d_{j_1}} \perp\!\!\!\perp \mathcal{F}_{d_{j_2}} \mid \mathcal{F}_i$ because $d_{j_1}, d_{j_2} \in Q_i$. Then by contraction, we obtain $\mathcal{F}_{d_{j_1}} \perp\!\!\!\perp \bigvee_{k=2}^{s} \mathcal{F}_{d_{j_k}} \vee \mathcal{F}_{c'} \mid \mathcal{F}_i$. Then d_{j_1} should be a set in a finer valid partition than P_i, which is a contradiction. □

Note that while the computation of the finest valid partitions can be exponentially complex in the number M of nodes, the finest pairwise partitions can be found in polynomial time; one only needs to look at the pairs of σ-fields at each time to check conditional independence. Therefore, starting with a conventional MPF problem, the question of existence of a probabilistic junction tree can be answered in polynomial time in M, as long as an unsigned measure is chosen.

Appendix D

On the Positive Rank Decomposition of Matrices

Consider a non-negative matrix $A \in \mathbb{R}_+^{m \times n}$ with rank r. We would like to additively decompose A as the sum of q non-negative rank-1 matrices, $B_1, \cdots, B_q \in \mathbb{R}_+^{m \times n}$. The smallest integer q for which such decomposition is possible is called the *non-negative rank* of A, and the decomposition is called the *non-negative rank decomposition* of A. It is easy to see that this decomposition is equivalent to factorization of A as the product of two non-negative matrices, $V \in \mathbb{R}_+^{m \times q}$ and $U \in \mathbb{R}_+^{q \times n}$ (see [7]).

Geometrically we can view A and V as collection of vectors in $\mathbb{R}_+^m$, with corresponding *polyhedral cones*[1] $\mathbf{C}(A)$ and $\mathbf{C}(V)$. Any non-negative factorization $A = VU$ is equivalent to the following (see [43]):

$$\mathbf{C}(A) \subseteq \mathbf{C}(V) \subseteq \mathbb{R}_+^m$$

Then the problem of minimizing q is equivalent to that of finding a polyhedral cone with minimal number of spanning vectors, which contains the cone of A, and is itself contained in the positive orthant.

If A has rank r, then it has r linearly independent column vectors in $\mathbb{R}_+^m$, so we can write $A = B\Gamma$ where $B = (\beta_{i,j}) \in \mathbb{R}_+^{m \times r}$ and $\Gamma = (\gamma_{j,k}) \in \mathbb{R}^{r \times n}$. Note that without

[1]For a matrix $X \in \mathbb{R}^{m \times n}$, we define $\mathbf{C}(X)$, the *polyhedral cone* of X, as the subset of $\mathbb{R}^m$ consisting of all non-negative linear combinations of the column vectors of X. The *span* of X, $\mathbf{S}(X)$ is defined as the subspace consisting of *all* linear combinations of columns of X.

loss of generality we can assume that A has no zero columns; if it does, then let A' be the matrix obtained by eliminating the zero columns of A, and let $A' = V'U'$ be a non-negative rank factorization of A'. Then $A = VU$ is a non-negative rank factorization of A, where $V = V'$ and U is obtained by inserting zero columns into U'.

Now note that for any cone $\mathbf{C}^*$, satisfying $\mathbf{C}(A) \subseteq \mathbf{C}^* \subseteq \mathbb{R}^m_+$, its projection, $\tilde{\mathbf{C}}^*$ onto $\mathbf{S}(A)$ satisfies $\tilde{\mathbf{C}}(A) \subseteq \tilde{\mathbf{C}}^* \subseteq \tilde{\mathbb{R}}^m_+$, where $\tilde{\mathbf{C}}(A)$ and $\tilde{\mathbb{R}}^m_+$ denote the projections of $\mathbf{C}(A)$ and the positive orthant onto $\mathbf{S}(A)$ respectively.

Viewing $\mathbf{S}(A)(= \mathbf{S}(B))$ as a copy of $\mathbb{R}^r$ (with basis vectors $b^1, \cdots, b^r$, the column vectors of B), $\tilde{\mathbf{C}}(A)$ is the cone generated by Γ, and $\tilde{\mathbb{R}}^m_+$ is defined by inequalities $\beta_{i,1}x_1 + \cdots + \beta_{i,r}x_r \geq 0$ for $i = 1, \cdots, m$. Now let $\mathbf{P}_i$ denote the $(r-1)$-dimensional subspace in $\mathbb{R}^r$ that is perpendicular to b_i, the ith row vector of B. Then $\tilde{\mathbb{R}}^m_+$ is the cone bounded by $\mathbf{P}_1, \cdots, \mathbf{P}_m$ (and in the directions of b_i's). Let b_0 be a vector in the interior of the cone of $\{b_1, \cdots, b_m\}$, say $b_0 = b_1 + \cdots + b_m$. Let $\mathbf{P}$ be the $(r-1)$-dimensional hyperplane perpendicular to, and passing through b_0. Then for any cone $\tilde{\mathbf{C}}^*$, satisfying $\tilde{\mathbf{C}}(A) \subseteq \tilde{\mathbf{C}}^* \subseteq \tilde{\mathbb{R}}^m_+$, the intersections $\big(\tilde{\mathbf{C}}(A) \cap \mathbf{P}\big)$, $\big(\tilde{\mathbf{C}}^* \cap \mathbf{P}\big)$ and $\big(\tilde{\mathbb{R}}^m_+ \cap \mathbf{P}\big)$ are $(r-1)$-dimensional convex subsets of space satisfying $\big(\tilde{\mathbf{C}}(A) \cap \mathbf{P}\big) \subseteq \big(\tilde{\mathbf{C}}^* \cap \mathbf{P}\big) \subseteq \big(\tilde{\mathbb{R}}^m_+ \cap \mathbf{P}\big)$. Further, $\big(\tilde{\mathbf{C}}(A) \cap \mathbf{P}\big)$ is a bounded polytope; to see this, note that $\tilde{\mathbf{C}}(A) = \mathbf{C}(\Gamma)$ is generated by $g^1, \cdots, g^n$, the column vectors of Γ. Now for each $k = 1, \cdots, n$, the dot-product $b_0 \cdot g^k = \sum_{i=1}^m b_i \cdot g^k$ is strictly positive since it is exactly the sum of the kth column of A. Then g^k intersects $\mathbf{P}$ at the point $\frac{|b_0|^2}{b_0 \cdot g^k} g^k$, which is finite and in the positive direction of g^k. Thus $\tilde{\mathbf{C}}(A) \cap \mathbf{P}$ is the convex hull of points $\frac{|b_0|^2}{b_0 \cdot g^k} g^k$ on $\mathbf{P}$, for $k = 1, \cdots, n$.

We have therefore shown that the problem can further be reduced to embedding of $(r-1)$-dimensional polytopes. It only remains to describe how to project $\tilde{\mathbf{C}}(A)$ and $\tilde{\mathbb{R}}^m_+$ onto $\mathbf{P}$. Remember that $\mathbf{P}$ is described by the equation $b_0 \cdot \mathrm{x} = \sum_{i=1}^m \beta_{i,1}x_1 + \cdots + \sum_{i=1}^m \beta_{i,r}x_r = |b_0|^2$. Noting that each column sum of B is strictly positive, we can write this as $x_r = \big(\sum_{i=1}^m \beta_{i,r}\big)^{-1}\big(|b_0|^2 - \big(\sum_{i=1}^m \beta_{i,1}x_1 + \cdots + \sum_{i=1}^m \beta_{i,r-1}x_{r-1}\big)\big)$. We can

then view the first $(r-1)$ coordinates of the points of $\mathbf{P}$ as the coordinates of a copy of $\mathbb{R}^{r-1}$. Then the inner polytope corresponding to the projection of $\mathbf{C}(\Gamma)$ onto $\mathbf{P}$ can be represented as the convex hull of points $\tilde{g}^k \in \mathbb{R}^{r-1}$, whose coordinates respectively are the first $(r-1)$ coordinate of $\frac{|b_0|^2}{b_0 \cdot g^k} g^k$, for $k = 1, \cdots, n$. The outer polytope corresponding to the projection of $\tilde{\mathbb{R}}^m_+$ onto $\mathbf{P}$ is represented by the inequalities

$$\text{for } i = 1, \cdots, m$$

$$\sum_{j=1}^{r-1} \beta_{i,j} x_j + \frac{\beta_{i,r}}{\sum_{k=1}^m \beta_{k,r}} \left(|b_0|^2 - \sum_{j=1}^{r-1} \sum_{k=1}^m \beta_{k,j} x_j \right) \geq 0$$

$$\text{or} \quad \sum_{j=1}^{r-1} (\beta_{i,j} - \frac{\beta_{i,r}}{\sum_{k=1}^m \beta_{k,r}} \sum_{k=1}^m \beta_{k,j}) x_j + \frac{\beta_{i,r}}{\sum_{k=1}^m \beta_{k,r}} |b_0|^2 \geq 0 \quad \text{(D.1)}$$

Let the columns of $\tilde{Q} \in \mathbb{R}^{(r-1) \times q}$ denote the vertices of a polytope with minimal number of vertices which contains all the vertices $\tilde{g}^k$ of the inner polytope, and also lie inside the outer polytope. Then there is a matrix $U' \in \mathbb{R}^{q \times n}_+$, such that the kth column of $\tilde{Q}U'$ is the vertex $\tilde{g}^k$ of the inner polytope. Now let $Q \in \mathbb{R}^{r \times q}$ be obtained from $\tilde{Q}$ by adding an rth row, where $q_{r,l} = (\sum_{i=1}^m \beta_{i,r})^{-1} \left(|b_0|^2 - \sum_{j=1}^{r-1} \sum_{i=1}^m \beta_{i,j} q'_{j,l} \right)$ for each $l = 1, \cdots, q$. Also construct the matrix $U \in \mathbb{R}^{q \times n}_+$ from U by multiplying the kth column by the positive quantity $b_0 \cdot g^k / |b_0|^2$. Then the kth column of QU is precisely the vector g^k in $\mathbb{R}^r$, i.e. $QU = \Gamma$. Thus we have $A = BQU$. But note that by lying inside the outer polytope, the column vectors of Q satisfy the inequalities (D.1). Thus BQ is an $m \times q$ matrix with non-negative entries. Let $V = BQ$, so we have our non-negative rank factorization, $A = VU$.

In the particular case when a non-negative matrix A has rank 3 the problem of non-negative rank decomposition of A reduces to the minimal nesting of convex polygons. In [1] Aggarwal et.al. present an $O(n \log(q))$ algorithm to solve this problem, where n is the total number of the vertices of the two polygons, and q is the number of vertices of the nested polygon, i.e. the non-negative rank of A. Note that q is upper-bounded by the maximum of length and width of A.

Appendix E

Overcounting Factors and Möbius Inversion Formula

With the setup of Section 9.1, let $\hat{R}$ be the collection R of subsets of $[N] := \{1, \ldots, N\}$ together with the set $[N]$ itself. Then $\hat{R}$ itself can be viewed as a poset with the partial ordering of inclusion. Let $\bar{H}_r$ denote the Möbius dual of the the regional entropy $H_r(b_r)$, see e.g. [40]. Then Möbius inversion formula states that

$$H_t = \sum_{\substack{r \in \hat{R} \\ r \subseteq t}} \bar{H}_r \qquad \text{for each } t \in \hat{R}, \tag{E.1}$$

$$\bar{H}_r = \sum_{\substack{u \in \hat{R} \\ u \subseteq r}} H_u \mu(u, r) \qquad \text{for each } r \in \hat{R} \tag{E.2}$$

Here the Möbius function $\mu(u, r)$ is defined for $u, r \in R, u \subseteq r$ by equations

$$\sum_{\substack{u \in R' \\ t \subseteq u \subseteq r}} \mu(u, r) = \delta_{tr} \tag{E.3}$$

Setting $r = [N]$ in (E.2) yields

$$H_{[N]} = H(b) = -\sum_{r \in R} H_r \mu(r, [N]) + \bar{H}_{[N]} \tag{E.4}$$

If the term $\bar{H}_{[N]}$ can be ignored, we get the approximation

$$H \simeq -\sum_{r \in R} H_r \, \mu(r, [N]) \tag{E.5}$$

This is precisely the Kikuchi approximation of the entropy term, where $c_r = -\mu(r, [N])$ are the overcounting factors, since for $r = [N]$, (E.3) reduces to (9.9).

Appendix F

Some Bounds on the Error of Kikuchi Approximate Free Energy

In this appendix we will show that the error of approximating the variational free energy (9.2) with the Kikuchi free energy (9.11) is bounded.

As before, let $\mathbf{x} = (x_1, \cdots, x_N)$ be the state vector, and let R is a collection of subsets of $\{1, \cdots, N\}$. We define the error term $D(b) := F(b) - F_R^K(\{b_r, r \in R\})$ for any probability distribution $b(\mathbf{x})$ with R-marginals $\{b_r(\mathbf{x}_r), r \in R\}$.

Let $\{c_r\,,\, r \in R\}$ be the collection of overcounting factors, defined in Section 9.1, and define $R^+ := \{r \in R : c_r \geq 0\}$ and $R^- := \{r \in R : c_r < 0\}$.

Notice that the first terms in $F(b)$ and $F_R^K(b)$, viz. the average energy term, are identical since the energy term $E(\mathbf{x})$ is assumed to be R-decomposable and hence

$$\sum_{\mathbf{x}} b(\mathbf{x})E(\mathbf{x}) = \sum_{\mathbf{x}} b(\mathbf{x}) \sum_{r \in R} E_r(\mathbf{x}_r) = \sum_{r \in R} \sum_{\mathbf{x}} b(\mathbf{x})E_r(\mathbf{x}_r) = \sum_{r \in R} \sum_{\mathbf{x}_r} b_r(\mathbf{x}_r)E_r(\mathbf{x}_r)$$

Thus

$$\begin{aligned} D(b) &= \sum_{\mathbf{x}} b(\mathbf{x}) \log(b(\mathbf{x})) - \sum_{r \in R} c_r \sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) \log(b_r(\mathbf{x}_r)) \\ &= \sum_{\mathbf{x}} b(\mathbf{x}) \log(b(\mathbf{x})) - \sum_{r \in R} c_r \sum_{\mathbf{x}} b(\mathbf{x}) \log(b_r(\mathbf{x}_r)) \\ &= \sum_{\mathbf{x}} b(\mathbf{x}) \log(\frac{b(\mathbf{x})}{\prod_r b_r(\mathbf{x}_r)^{c_r}}) \end{aligned}$$

Define $Q(\mathbf{x}) = \prod_{r \in R} b_r(\mathbf{x}_r)^{c_r}$, and notice that $\forall r \in R, \forall \mathbf{x},\ b_r(\mathbf{x}_r) \geq b(\mathbf{x})$. Therefore

$$\begin{aligned} Q(\mathbf{x}) &= \prod_{r \in R^+} b_r(\mathbf{x}_r)^{c_r} \cdot \prod_{r \in R^-} b_r(\mathbf{x}_r)^{c_r} \\ &\geq b(\mathbf{x})^{\sum_{r \in R^+} c_r} \cdot \prod_{r \in R^-} b_r(\mathbf{x}_r)^{c_r} \\ &\geq b(\mathbf{x})^{\sum_{r \in R^+} c_r} \end{aligned}$$

where the last inequality follows from the fact that $0 \leq b_r \leq 1$ and hence $b_r^a \geq 1$ for any negative number a. Then we have $\frac{b(\mathbf{x})}{Q(\mathbf{x})} \leq b(\mathbf{x})^{(1-\sum_{r \in R^+} c_r)}$ and hence

$$\begin{aligned} D(b) &= \sum_{\mathbf{x}} b(\mathbf{x}) \log(\frac{b(\mathbf{x})}{Q(\mathbf{x})}) \\ &\leq (1 - \sum_{r \in R^+} c_r) \sum_{\mathbf{x}} b(\mathbf{x}) \log(b(\mathbf{x})) \\ &= (\sum_{r \in R^+} c_r - 1) H(b) \end{aligned} \tag{F.1}$$

Note that the range of state vector $\mathbf{x}$ is $\prod_{i=1}^{N} [q_i]$, therefore the entropy term $H(b)$ is at most $\log(\prod_{i=1}^{N} q_i = \sum_{i=1}^{N} \log(q_i)$.

Similarly we can write

$$\begin{aligned} Q(\mathbf{x}) &= \prod_{r \in R^+} b_r(\mathbf{x}_r)^{c_r} \cdot \prod_{r \in R^-} b_r(\mathbf{x}_r)^{c_r} \\ &\leq \prod_{r \in R^+} b_r(\mathbf{x}_r)^{c_r} \cdot b(\mathbf{x})^{\sum_{r \in R^-} c_r} \\ &\leq b(\mathbf{x})^{\sum_{r \in R^-} c_r} \end{aligned}$$

Then we have $\frac{b(\mathbf{x})}{Q(\mathbf{x})} \geq b(\mathbf{x})^{(1-\sum_{r \in R^+} c_r)}$ and hence

$$\begin{aligned} D(b) &= \sum_{\mathbf{x}} b(\mathbf{x}) \log(\frac{b(\mathbf{x})}{Q(\mathbf{x})}) \\ &\geq (1 - \sum_{r \in R^-} c_r) \sum_{\mathbf{x}} b(\mathbf{x}) \log(b(\mathbf{x})) \\ &= -(1 - \sum_{r \in R^+} c_r) H(b) \end{aligned} \tag{F.2}$$

Equations (F.1) and (F.2) give upper and lower bounds on the error term $D(b)$.

Appendix G

Legendre Transform, Plefka Expansion and Mean-Field Methods

Consider again the Boltzmann distribution:

$$B(\mathbf{x}) := \frac{e^{-\beta E(\mathbf{x})}}{Z(\beta)} \tag{G.1}$$

Throughout this section we will keep the explicit dependence on the inverse temperature parameter, β. We will further assume that the energy function decomposes as $E(\mathbf{x}) = \sum_i E_i(x_i) + \sum_{(i,j)} E_{i,j}(x_i, x_j)$.

We are interested to find a distribution $P(\mathbf{x})$ belonging to a class $\mathcal{P}$ of (simple) distributions, which is *close* to $B(\mathbf{x})$ in the sense of minimizing the KL-divergence:

$$P_\beta(\mathbf{x}) := \arg \min_{b(\mathbf{x}) \in \mathcal{P}} \mathrm{KL}(b||B) \tag{G.2}$$

But

$$\mathrm{KL}(b||B) = \sum_{\mathbf{x}} b(\mathbf{x}) \ln(\frac{b(\mathbf{x})}{B(\mathbf{x})}) \tag{G.3}$$

$$= \sum_{\mathbf{x}} b(\mathbf{x}) \ln(b(\mathbf{x})) + \sum_{\mathbf{x}} \beta b(\mathbf{x}) E(\mathbf{x}) + \ln(Z(\beta)) \tag{G.4}$$

$$= -S(b) + U(b;\beta) + \ln(Z(\beta)) \tag{G.5}$$

$$= \ln(Z(\beta)) + F(b;\beta) \tag{G.6}$$

where as before, $S(b)$, $U(b;\beta)$ and $F(b;\beta)$ are the entropy, average energy and variational free energy associated with $b(\mathbf{x})$ respectively. Now $Z(\beta)$ does not depend on

$b(\mathbf{x})$. Therefore we have

$$P_\beta(\mathbf{x}) := \arg\min_{b(\mathbf{x})\in\mathcal{P}} F(b;\beta) \tag{G.7}$$

Thus as before we are looking to minimize the variational free energy. Mean Field approximation is obtained when at this stage one chooses the constraint set $\mathcal{P}$ to be the collection of product distributions, i.e. $\mathcal{P} = \{\text{distributions } b(\mathbf{x}) = \prod_{i=1}^{N} b_i(x_i)\}$: for any distribution $b \in \mathcal{P}$,

$$S(b) = -\sum_{\mathbf{x}} b(\mathbf{x}) \sum_{i} \ln(b_i(x_i)) = -\sum_{i}\sum_{x_i} b_i(x_i)\ln(b_i(x_i)) \tag{G.8}$$

$$U(b;\beta) = \sum_{\mathbf{x}} \beta b(\mathbf{x}) E(\mathbf{x}) = \beta \sum_{i}\sum_{x_i} b_i(x_i)E_i(x_i) + \beta \sum_{(i,j)}\sum_{x_i,x_j} b_i(x_i)b_j(x_j)E_{i,j}(x_i,x_j) \tag{G.9}$$

One then minimizes the free energy by differentiating with respect to each $b_i(x_i)$, to get the relations

$$b_i(x_i) \propto e^{-\beta\left(E_i(x_i)+\sum_j \sum_{x_j} b_j(x_j)E_{i,j}(x_i,x_j)\right)} \tag{G.10}$$

In an Ising model, we have binary variables ($x_i \in \{-1,+1\}$), and energy terms in the form $\beta E_{i,j}(x_i,x_j) = x_i x_j J_{i,j}$ and $\beta E_i(x_i) = x_i\theta_i$. Then equation (G.10) translates into the well-known Mean Field Equations,

$$m_i = \tanh(\theta_i + \sum_j J_{i,j} m_j) \tag{G.11}$$

where m_i is the the mean of distribution $b_i(x_i)$.

Mean Field approximation above is the crudest in a family of approximations obtained using *Plefka expansion*, see [34]. To develop these approximations, we start back with the minimization of equation (G.7), and we remove the constraint collection $\mathcal{P}$, so that the minimization is done over the complete collection of distributions. Clearly the minimizing distribution would be the original Boltzmann distribution.

To introduce the approximations, we rewrite the minimization of (G.7) in two steps: first we minimize the free energy over those distributions with fixed means

$\sum_{\mathbf{x}} x_i b(\mathbf{x}) = m_i$ for $i = 1, \cdots, N$. Next we minimize over the vector of the means $(m_1, \cdots, m_N)$.

We define Gibbs free energy $G_\beta(\{m_i\})$ as the constrained minimum in the first step:

$$G_\beta(\{m_i\}) := \min_{b(\mathbf{x}):\sum_{\mathbf{x}} x_i b(\mathbf{x}) = m_i} F(b;\beta) \tag{G.12}$$

Introducing Lagrange multipliers $\{\lambda_i\}$ to enforce the constraints we get

$$G_\beta(\{m_i\}) = \max_{\{\lambda_i\}} \min_{b(\mathbf{x})} \Big(F(b;\beta) + \sum_{i=1}^{N} \lambda_i (m_i - \sum_{\mathbf{x}} x_i b(\mathbf{x})) \Big) \tag{G.13}$$

$$= \min_{b(\mathbf{x})} \Big(F(b;\beta) + \sum_{i=1}^{N} \lambda_i(\beta) (m_i - \sum_{\mathbf{x}} x_i b(\mathbf{x})) \Big) \tag{G.14}$$

where $\{\lambda_i(\beta)\}$ are the set of multipliers chosen to satisfy the constraints. Equation (G.13) is known as the *Legendre transform* between $\{m_i\}$ and $\{\lambda_i\}$.

Differentiating (G.14) with respect to each $b(\mathbf{x})$ and equating to zero, we see that the distribution $b_\beta(\mathbf{x})$ that minimizes (G.14) has the exponential form:

$$b_\beta(\mathbf{x}) = \frac{e^{-\beta E(\mathbf{x}) + \sum_i \lambda_i(\beta) x_i}}{Z(\{\lambda_i(\beta)\};\beta)}, \quad Z(\{\lambda_i(\beta)\};\beta) = \sum_{\mathbf{x}} e^{-\beta E(\mathbf{x}) + \sum_i \lambda_i(\beta) x_i} \tag{G.15}$$

Then we have

$$\begin{aligned} G_\beta(\{m_i\}) &= U(b_\beta;\beta) - S(b_\beta) \\ &= \beta \sum_{\mathbf{x}} b_\beta(\mathbf{x}) E(\mathbf{x}) - \sum_{\mathbf{x}} b_\beta(\mathbf{x}) \big(\beta E(\mathbf{x}) - \sum_i \lambda_i(\beta) x_i + \ln(Z(\{\lambda_i(\beta)\};\beta)) \big) \\ &= \sum_i \lambda_i(\beta) x_i - \ln(Z(\{\lambda_i(\beta)\};\beta)) \end{aligned} \tag{G.16}$$

The advantage of the two-stage approach lies in the ability to Taylor-expand the equation (G.16) for small β; this is called the *Plefka expansion*, see [34]. It can be seen that for the Ising model discussed above, the zeroth and the first order terms of the expansion correspond exactly to the mean field free energy (see [34; 27; 51]). The second order term in the expansion is known as the *Onsager Reaction Term*. For the Ising model above, the approximation to the Gibbs free energy obtained by

keeping upto the second order terms in the Plefka expansion above is known as the TAP free energy ([42]):

$$G_{\text{TAP}}(\{m_i\}) := \sum_i \left(\frac{1+m_i}{2}\ln\left(\frac{1+m_i}{2}\right) + \frac{1-m_i}{2}\ln\left(\frac{1-m_i}{2}\right)\right) - \sum_i \theta_i m_i - \sum_{(i,j)} m_i m_j J_{i,j} - \frac{1}{2}\sum_{(i,j)} J_{i,j}^2 (1-m_i^2)(1-m_j^2) \quad \text{(G.17)}$$

where the last term is the aforementioned Onsager term.

Minimizing $G_{\text{TAP}}(\{m_i\})$ over the vector $\{m_i\}$ of the means gives the TAP approximation to the free energy, $F_0 = -\ln(Z)$ of the original Boltzmann distribution. Correspondingly, the minimizing vector can be used as the approximate means of variables x_i under the Boltzmann distribution.

It is worth mentioning that the TAP approach was devised under the assumptions of the Sherrington-Kirpatrick (SK) Ising model, where the coupling terms $J_{i,j}$'s are independent Gaussian random variables (see [39]). It is therefore important to note that for a general problem, the addition of Onsager's term may even *deteriorate* the approximation over the simple Mean Field method.

Appendix H

CCCP Algorithm to Minimize Kikuchi Free Energy

In this section we present Yuille's concave-convex procedure (CCCP) to minimize the general Kikuchi free energy. This is an iterative algorithm, consisting of an inner and an outer loop, and is guaranteed to converge to a constrained minimum of the Kikuchi free energy. See [54] for more details.

Theorem H.1. (See [54] Theorem 1) *Let* $E(\mathbf{z}) = E_{vex}(\mathbf{z}) + E_{cave}(\mathbf{z})$ *be an energy function to be minimized, where* $E_{vex}(\mathbf{z})$ *and* $E_{cave}(\mathbf{z})$ *are convex and concave functions of* $\mathbf{z}$ *respectively. Then a discrete iterative algorithm* $\mathbf{z}^t \mapsto \mathbf{z}^{t+1}$ *s.t. at each step*

$$\nabla E_{vex}(\mathbf{z}^{t+1}) = -\nabla E_{cave}(\mathbf{z}^t) \tag{H.1}$$

is guaranteed to monotonically decrease the energy function $E(\mathbf{z})$ *and hence to converge to a minimum of* $E(\mathbf{z})$.

Proof. By convexity and concavity of E_{vex} and E_{cave}, for all $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3$ and $\mathbf{z}_4$:

$$E_{vex}(\mathbf{z}_2) \geq E_{vex}(\mathbf{z}_1) + (\mathbf{z}_2 - \mathbf{z}_1) \cdot \nabla E_{vex}(\mathbf{z}_1)$$

$$E_{cave}(\mathbf{z}_4) \leq E_{cave}(\mathbf{z}_3) + (\mathbf{z}_4 - \mathbf{z}_3) \cdot \nabla E_{cave}(\mathbf{z}_3).$$

Replacing $\mathbf{z}_1 = \mathbf{z}_4 = \mathbf{z}^{t+1}$ and $\mathbf{z}_2 = \mathbf{z}_3 = \mathbf{z}^t$ in the above inequalities and using $\nabla E_{vex}(\mathbf{z}^{t+1}) = -\nabla E_{cave}(\mathbf{z}^t)$ we obtain:

$$E_{vex}(\mathbf{z}^{t+1}) + E_{cave}(\mathbf{z}^{t+1}) \leq E_{vex}(\mathbf{z}^t) + E_{cave}(\mathbf{z}^t)$$

□

To solve the constrained minimization problem (9.14) we form the Lagrangian:

$$\mathcal{L}^K\Big(\{b_r(\mathbf{x}_r);\lambda_{rt}(\mathbf{x}_t);\kappa_r\,,\ t\prec r\in R\}\Big) := \sum_{r\in R} c_r\, b_r(\mathbf{x}_r)\log\big(\frac{b_r(\mathbf{x}_r)}{\beta_r(\mathbf{x}_r)}\big) + \sum_{r\in R}\sum_{s\in C(r)}\sum_{\mathbf{x}_s}\lambda_{rs}(\mathbf{x}_s)\big(b_s(\mathbf{x}_s)-\sum_{\mathbf{x}_{r\setminus s}} b_r(\mathbf{x}_r)\big) + \sum_{r\in R}\kappa_r\big(\sum_{\mathbf{x}_r} b_r(\mathbf{x}_r)-1\big) \quad \text{(H.2)}$$

where multipliers $\lambda_{rt}(\mathbf{x}_t)$ and κ_r enforce consistency and normalization constraints of Δ_R^K respectively (see equation (9.12)).

As suggested by Theorem H.1 we decompose $\mathcal{L}^K$ into convex and concave parts $\mathcal{L}^K_{vex}$ and $\mathcal{L}^K_{cave}$:

$$\begin{aligned}\mathcal{L}^K_{vex} := {} & c^*\sum_{r\in R}\sum_{\mathbf{x}_r} b_r(\mathbf{x}_r)\log\big(\frac{b_r(x_r)}{\beta_r(x_r)}\big) + \sum_{r\in R}\kappa_r\big(\sum_{\mathbf{x}_r} b_r(\mathbf{x}_r)-1\big) \\ & + \sum_{r\in R}\sum_{s\in C(r)}\sum_{\mathbf{x}_s}\lambda_{rs}(\mathbf{x}_s)\big(b_s(\mathbf{x}_s)-\sum_{\mathbf{x}_{r\setminus s}} b_r(\mathbf{x}_r)\big)\end{aligned} \quad \text{(H.3)}$$

$$\mathcal{L}^K_{cave} := -\sum_{r\in R}(c^*-c_r)\sum_{\mathbf{x}_r} b_r(\mathbf{x}_r)\log\big(\frac{b_r(x_r)}{\beta_r(x_r)}\big) \quad \text{(H.4)}$$

where $c^* := \max_{r\in R} c_r$ is the largest overcounting factor of any region.

Then Yuille's condition (H.1) is equivalent to minimizing the following convex functional of $\{b_r^{t+1}(\mathbf{x}_r),\, r\in R\}$, where the superscripts t are time indices:

$$\begin{aligned}E^{t+1}(\{b_r^{t+1}(\mathbf{x}_r)\}) := {} & \sum_{r\in R}\sum_{\mathbf{x}_r} b_r^{t+1}(\mathbf{x}_r)\cdot\frac{\partial\mathcal{L}^K_{cave}}{\partial b_r^t(\mathbf{x}_r)}\big(b_r^t(\mathbf{x}_r)\big) \\ & + c^*\sum_{r\in R}\sum_{\mathbf{x}_r} b_r^{t+1}(\mathbf{x}_r)\log\big(\frac{b_r^{t+1}(x_r)}{\alpha_r(x_r)}\big) + \sum_{r\in R}\kappa_r\big(\sum_{\mathbf{x}_r} b_r^{t+1}(\mathbf{x}_r)-1\big) \\ & + \sum_{r\in R}\sum_{s\in C(r)}\sum_{\mathbf{x}_s}\lambda_{rs}(\mathbf{x}_s)\big(b_s^{t+1}(\mathbf{x}_s)-\sum_{\mathbf{x}_{r\setminus s}} b_r^{t+1}(\mathbf{x}_r)\big)\end{aligned} \quad \text{(H.5)}$$

with

$$\frac{\partial\mathcal{L}^K_{cave}}{\partial b_r^t(\mathbf{x}_r)}\big(b_r^t(\mathbf{x}_r)\big) = -(c^*-c_r)\big(\log\big(\frac{b_r^t(x_r)}{\alpha_r(x_r)}\big)+1\big) \quad \text{(H.6)}$$

Setting partial derivatives $\frac{\partial E^{t+1}}{\partial b_r^{t+1}(\mathbf{x}_r)}$ equal to zero, the minimum of (H.5) is acheived with

$$b_r^{t+1}(\mathbf{x}_r) = b_r^t(\mathbf{x}_r)^{1-\frac{c_r}{c^*}}\,\beta_r(\mathbf{x}_r)^{\frac{c_r}{c^*}}\,e^{-\frac{c_r}{c^*}}\,e^{-\frac{1}{c^*}\left(\kappa_r+\sum_{s\in C(r)}\lambda_{rs}(\mathbf{x}_s)-\sum_{s\in P(r)}\lambda_{sr}(\mathbf{x}_r)\right)} \quad \text{(H.7)}$$

Equation (H.7) is the update rule for the *outer loop* of Yuille's algorithm and is indexed by the time index t. By duality then, the Lagrange multipliers $\{\lambda_{rs}(\mathbf{x}_s); \kappa_r,\ r \prec s \in R\}$ are constrained to maximize the *concave* dual energy:

$$\hat{E}^{t+1}(\{\lambda_{rs}(\mathbf{x}_s); \kappa_r\}) = \sum_{r \in R} \frac{\kappa_r}{c^*} - \sum_{r \in R} \sum_{\mathbf{x}_r} b_r^t(\mathbf{x}_r)^{1-\frac{c_r}{c^*}}\, \beta_r(\mathbf{x}_r)^{\frac{c_r}{c^*}}\, e^{-\frac{c_r}{c^*}}\, e^{-\frac{1}{c^*}\left(\kappa_r + \sum_{s \in C(r)} \lambda_{rs}(\mathbf{x}_s) - \sum_{s \in P(r)} \lambda_{sr}(\mathbf{x}_r)\right)} \quad \text{(H.8)}$$

Since $\hat{E}^{t+1}$ is a concave function of $\{\lambda_{rs}(\mathbf{x}_s); k_r\}$, solving the constraint equations one by one is guaranteed to converge to the unique maximum of (H.8). We therefore obtain the *inner loop* of the algorithm, which is indexed by time index τ:

$$e^{(\kappa_r^{\tau+1}/c^*)} = e^{-\frac{c_r}{c^*}} \sum_{\mathbf{x}_r} b_r^t(\mathbf{x}_r)^{1-\frac{c_r}{c^*}}\, \beta_r(\mathbf{x}_r)^{\frac{c_r}{c^*}}\, e^{-\frac{1}{c^*}\left(\sum_{s \in C(r)} \lambda_{rs}^{\tau}(\mathbf{x}_s) - \sum_{s \in P(r)} \lambda_{sr}^{\tau}(\mathbf{x}_r)\right)} \quad \text{(H.9)}$$

$$e^{\frac{2}{c^*}\lambda_{ru}^{\tau+1}(\mathbf{x}_u)} = \frac{\sum_{\mathbf{x}_{r \setminus u}} b_r^t(\mathbf{x}_r)^{1-\frac{c_r}{c^*}}\, \beta_r(\mathbf{x}_r)^{\frac{c_r}{c^*}}\, e^{-\frac{c_r}{c^*}}\, e^{-\frac{1}{c^*}\left(\kappa_r^{\tau} + \sum_{s \in C(r)\setminus\{u\}} \lambda_{rs}^{\tau}(\mathbf{x}_s) - \sum_{s \in P(r)} \lambda_{sr}^{\tau}(\mathbf{x}_r)\right)}}{b_u^t(\mathbf{x}_u)^{1-\frac{c_u}{c^*}}\, \beta_u(\mathbf{x}_u)^{\frac{c_u}{c^*}}\, e^{-\frac{c_u}{c^*}}\, e^{-\frac{1}{c^*}\left(\kappa_u^{\tau} + \sum_{s \in C(u)} \lambda_{us}^{\tau}(\mathbf{x}_s) - \sum_{s \in P(u)\setminus\{r\}} \lambda_{su}^{\tau}(\mathbf{x}_u)\right)}} \quad \text{(H.10)}$$

where equations (H.9) and (H.10) are respectively obtained from normalization and consistency constraints. Defining $m_{ru}(\mathbf{x}_u) := e^{\lambda_{ru}(\mathbf{x}_u)/c^*}$ and $A_r := e^{-(\kappa_r + c_r)/c^*}$, we can rewrite this algorithm more compactly:

Outer loop update equation:

$$b_r^{t+1}(\mathbf{x}_r) = A_r\, b_r^t(\mathbf{x}_r)^{1-\frac{c_r}{c^*}}\, \beta_r(\mathbf{x}_r)^{\frac{c_r}{c^*}} \Big(\prod_{s \in P(r)} m_{sr}(\mathbf{x}_r) \Big) \Big/ \Big(\prod_{s \in C(r)} m_{rs}(\mathbf{x}_s) \Big) \quad \text{(H.11)}$$

where $\{m_{rs}(\mathbf{x}_s); A_r,\ s \prec r \in R\}$ are the fixed points of the inner loop, with update equations:

$$A_r^{\tau+1} = \Big(\sum_{\mathbf{x}_r} b_r^t(\mathbf{x}_r)^{1-\frac{c_r}{c^*}}\, \beta_r(\mathbf{x}_r)^{\frac{c_r}{c^*}} \Big(\prod_{s \in P(r)} m_{sr}^{\tau}(\mathbf{x}_r) \Big) \Big/ \Big(\prod_{s \in C(r)} m_{rs}^{\tau}(\mathbf{x}_s) \Big) \Big)^{-1} \quad \text{(H.12)}$$

$$m_{ru}^{\tau+1}(\mathbf{x}_u) = \Bigg(\frac{\sum_{\mathbf{x}_{r \setminus u}} A_r\, b_r^t(\mathbf{x}_r)^{1-\frac{c_r}{c^*}}\, \beta_r(\mathbf{x}_r)^{\frac{c_r}{c^*}} \left(\prod_{s \in P(r)} m_{sr}(\mathbf{x}_r) \right) \Big/ \left(\prod_{s \in C(r)\setminus\{u\}} m_{rs}(\mathbf{x}_s) \right)}{A_u\, b_u^t(\mathbf{x}_u)^{1-\frac{c_u}{c^*}}\, \beta_u(\mathbf{x}_u)^{\frac{c_u}{c^*}} \left(\prod_{s \in P(u)\setminus\{r\}} m_{su}(\mathbf{x}_u) \right) \Big/ \left(\prod_{s \in C(u)} m_{us}(\mathbf{x}_s) \right)} \Bigg)^{1/2} \quad \text{(H.13)}$$

Bibliography

[1] A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, and C.K. Yap. "Finding minimal convex nested polygons", *Information and Computation*, 83:98–110, 1989.

[2] S.M. Aji and R.J. McEliece. "The generalized distributive law", *IEEE Transactions on Information Theory*, 46(2):325–343, March 2000.

[3] S.M. Aji and R.J. McEliece. "The generalized distributive law and free energy minimization". In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, pages 672–681, October 2001.

[4] F. Baccelli, G. Cohen, G.J. Olsder, and J.P. Quadrat. *Synchronization and Linearity.* Wiley, New York, NY, 1992.

[5] L.R. Bahl, J. Cocke, F. Jelinek, and J.Raviv. "Optimal decoding of linear codes for minimizing symbol error rate", *IEEE Transactions on Information Theory*, 20(2):284–287, March 1974.

[6] C. Berrou, A. Glavieux, and P. Thitimajshima. "Near shannon limit error-correcting coding and decoding: Turbo-codes". In *Proceedings of the IEEE International Conference on Communications*, number 2, pages 1064–1070, Geneva, May 1993.

[7] J.E. Cohen and U.G. Rothblum. "Nonnegative ranks, decompositions, and factorization of nonnegative matrices", *Linear Algebra and its Applications*, 190:149–168, 1993.

[8] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms.* McGraw-Hill, Cambridge, MA, 2001.

[9] T. Cover and J. Thomas. *Elements of Information Theory.* Wiley-Interscience, New York, NY, 1991.

[10] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Springer-Verlag, New York, NY, 1999.

[11] D. Divsalar, H. Jin, and R. McEliece. "Coding theorems for 'turbo-like' codes". In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, pages 201–210, October 1998.

[12] H. Federer. *Geometric Measure Theory.* Springer-Verlag, New York, NY, 1969.

[13] R.G. Gallager. *Low-Density Parity-Check Codes.* MIT Press, Cambridge, MA, 1963.

[14] P. Hall. "On representatives of subsets", *Journal of London Mathematical Society*, (10):26–30, 1935.

[15] R. Kikuchi. "A theory of cooperative phenomena", *Physical Review*, 6(81):988–1003, 1951.

[16] C. Kittel and H. Kroemer. *Thermal Physics.* New York, NY, 1980.

[17] F. Kschischang, B. Frey, and H.-A. Loeliger. "Factor graphs and the sum-product algorithm", *IEEE Transactions on Information Theory*, 47(2):498–519, February 2001.

[18] B. Kurkoski, P. Siegel, and J. Wolf. "Joint message-passing decoding of ldpc codes and partial-response channels", *IEEE Transactions on Information Theory*, 48(6):1410–1422, June 2002.

[19] M. Luby. "LT-codes". In *Proceedings of IEEE Symposium on the Foundations of Computer Science*, pages 271–280, November 2002.

[20] D.J.C. MacKay and R.M. Neal. "Good codes based on very sparse matrices". In *Cryptography and Coding: 5th IMA Conference*, number 1025, pages 100–111. Springer-Verlag, Berlin, 1995.

[21] R.J. McEliece. "On the BCJR trellis for linear block codes", *IEEE Transactions on Information Theory*, 42(4):1072–1092, July 1996.

[22] R.J. McEliece, D.J.C. MacKay, and J.F. Cheng. "Turbo decoding as an instance of Pearl's 'belief propagation' algorithm", *IEEE Journal on Selected Areas in Communications*, 16(2):140–152, February 1998.

[23] R.J. McEliece and M. Yildrim. "Belief propagation on partially ordered sets". In *Mathematical Systems Theory in Biology, Communications, Computation, and Finance*, pages 275–300. Springer, 2003.

[24] T. Morita. "Formal structure of the cluster variation method", *Progress of Theoretical Physics Supplement*, (115):27–39, 1994.

[25] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, U.C. Berkeley, Computer Science Division, July 2002.

[26] K. Murphy, Y. Weiss, and M. Jordan. "Loopy belief propagation for approximate inference: An empirical study". In *Proceedings of Uncertainty in Artificial Intelligence*, volume 15, pages 467–475. Morgan Kauffmann, 1999.

[27] M. Opper and O. Winther. "From naive mean field theory to the TAP equations". In *Advanced Mean Field Methods*, pages 7–20. MIT Press, Cambridge, MA, 2001.

[28] P. Pakzad and V. Anantharam. "Conditional independence for signed measures". Available online at http://inst.eecs.berkeley.edu/ payamp/signedci.pdf.

[29] P. Pakzad and V. Anantharam. "Estimation and marginalization using kikuchi based methods", *Submitted for publication in* Neural Computation.

[30] P. Pakzad and V. Anantharam. "Belief propagation and statistical physics". In *Proceedings of the Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, March 2002. Paper No. 225.

[31] P. Pakzad and V. Anantharam. "Minimal graphical representation of kikuchi regions". In *Proceedings of the 40th Allerton Conference on Communication, Control, and Computing*, pages 1585–1594, Urbana, Il, October 2002.

[32] P. Pakzad and V. Anantharam. "A new look at the generalized distributive law". To appear in *IEEE Transactions on Information Theory*, June 2004.

[33] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[34] T. Plefka. "Convergence condtition of the TAP equations for the infinite-ranged Ising spin glass model", *Journal of Physics A: Mathematical and General*, 15(6):1971–1978, June 1982.

[35] T. Richardson. "The geometry of turbo-decoding dynamics", *IEEE Transactions on Information Theory*, 46(1):9–23, January 2000.

[36] T. Richardson, A. Shokrollahi, and R. Urbanke. "Design of capacity-approaching irregular low-density parity-check codes", *IEEE Transactions on Information Theory*, 47:619–637, February 2001.

[37] T. Richardson and R. Urbanke. "The capacity of low-density parity-check codes under message-passing decoding", *IEEE Transactions on Information Theory*, 47:599–618, February 2001.

[38] G.R. Shafer and P.P. Shenoy. "Probability propagation", *Annals of Mathematics and Artificial Intelligence*, 2:327–352, 1990.

[39] D. Sherrington and S. Kirkpatrick. "Solvable model of a spin-glass", *Physical Review Letters*, 35(26):1792–1795, December 1975.

[40] R.P. Stanley. *Enumerative Combinatorics*, volume I. Wadsworth & Brooks/Cole, Monterey, CA, 1986.

[41] R.M. Tanner. "A recursive approach to low complexity codes", *IEEE Transactions on Information Theory*, (27):533–547, September 1981.

[42] D. Thouless, P. Anderson, and R. Palmer. "Solution of 'solvable model of a spin-glass"', *Philosophical Magazine*, 35(3):593–601, 1977.

[43] J.M. van den Hof and J.H. van Schuppen. "Positive matrix factorization via extremal polyhedral cones", *Linear Algebra and its Applications*, 293:171–186, 1999.

[44] M.J. Wainwright and M.I. Jordan. "Graphical models, exponential families, and variational inference". Technical Report 649, U.C. Berkeley, Dept. of Statistics, September 2003.

[45] J. Walrand and P. Varaiya. *High-Performance Communication Networks.* Morgan Kaufmann, San Francisco, CA, 1996.

[46] Y. Weiss. "Correctness of local probability propagation in graphical models with loops", *Neural Computation*, 12:1–41, 2000.

[47] M. Welling and Y.W. Teh. "Belief optimization for binary networks: a stable alternative to loopy belief propagation". In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, pages 554–561, 2001.

[48] S. Wicker. *Error Control Systems for Digital Communication and Storage.* Prentice Hall, Upper Saddle River, NJ, 1995.

[49] J. Yedidia, W. Freeman, and Y. Weiss. "Bethe free energy, Kikuchi approximations, and belief propagation algorithms". Technical Report TR2001-16, Mitsubishi Electronic Research Lab., May 2001.

[50] J. Yedidia, W. Freeman, and Y. Weiss. "Constructing free energy approximations and generalized belief propagation algorithms". Technical Report TR2002-35, Mitsubishi Electronic Research Lab., August 2002.

[51] J. Yedidia and A. Georges. "The fully frustrated ising model in infinite dimensions", *Journal of Physics A: Mathematical and General*, 23(11):2165–2171, June 1990.

[52] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam. "High throughput low-density parity-check architectures". In *Proceedings of IEEE Globecom2001*, pages 3019–3024, San Antonio, TX, November 2001.

[53] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam. "Vlsi architectures for iterative decoders in magnetic recording channels", *IEEE Transactions on Magnetics*, (2):748–755, March 2001.

[54] A.L. Yuille. "CCCP algorithms to minimize the bethe and kikuchi free energies: Convergent alternatives to belief propagation", *Neural Computation*, (14):1691–1722, 2002.

[55] N.L. Zhang and D. Poole. "Exploiting causal independence in Bayesian network inference", *Journal of Artificial Intelligence Research*, 5:301–328, 1996.