Belief Propagation

by

Dennis Kao

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Belief Propagation

Dennis Kao

Master of Science

Graduate Department of Computer Science

University of Toronto

2005

There are a wide assortment of descriptions of the belief propagation algorithm for marginalisation because of its vast applicability. Hence the following thesis aims to use consistent notation first to describe the crux of graphical models, in particular the relationship between Markov random fields, Bayesian networks, and factor graphs. Secondly, to illustrate the fundamentals and preliminary analyses of belief propagation, namely its relevance to Bethe free energy and LDPC codes, and a precursory empirical investigation. Finally, to discuss the application of belief propagation to satisfiability, culminating in survey propagation, one of belief propagation's promising progeny.

Acknowledgements

Firstly, thanks to both of my supervisors, Toni Pitassi and Alasdair Urquhart, for being there, even if I didn't launch any full-scale bombardments of questions and comments. Profuse appreciation to all of my friends for distracting me with wondrous adventures and marvelous revelations of web sites, music, television, and film. Penultimately, boundless gratitude to my family for being supportive from a distance. Finally, I owe my recent sustenance to the fine grocery stores and restaurants that deliver, without which I would surely have starved to death years ago.

Contents

Chapter 1

Introduction

In 1986, Judea Pearl formulated belief propagation, an approximate message-passing algorithm designed to solve marginalisation and other inference problems for probability distributions with graph representations called graphical models [Pea88]. In some respects, Gallager's somewhat similar decoding algorithm for error-correcting codes preceded belief propagation, which he devised in 1963 along with his celebrated low-density parity-check codes [Gal63]. The greatest testament to the versatility and generality of belief propagation is the countless descriptions of the algorithm in a variety of mathematical disciplines since its inception, including coding theory [MMC98, MN96, Mac03], physics [YFW02, BMZ03], data mining [BK02], and of course statistics and artificial intelligence [Pea88, Fre98, KFL01, Nea90]. Unfortunately, its diverse applicability also results in a wide range of expository approaches that lack a universal standard in notation. Thus some sort of fusion of some of these sundry portrayals of belief propagation is desirable. There have been many generalisations of belief propagation from the basic message-passing framework [AM97, AM00] to many complex approaches [BMZ03, WJW03, YFW02].

Moreover, belief propagation can be implemented using Markov random fields

[Wei01], Bayesian networks [Pea88], and factor graphs [KFL01] often with unique notation even amongst various discussions of the same implementation of belief propagation. Hence delving into the particulars of belief propagation first necessitates a clear understanding of the underlying graphical models. Upon describing these models and belief propagation itself, a closer examination of some of the theoretical advances pertaining to belief propagation is in order. Unfortunately, marginalisation and its relative approximation are NP-hard. Consequently, theoretical analysis of belief propagation and other algorithms for marginalisation will likely rely on particular types of graphs. Curiously, thus far, the related approximation technique of free energy approximation [Hes04, Hes03, YFW02] and the application of low density parity-check codes [RU01] provide most of theoretical work for belief propagation. Recently, advances towards sufficient conditions for the convergence of loopy belief propagation are promising [IIW05, MK05, TJ01]. However, the most significant result for correctness is rather unsatisfactory, namely the result for a single-cycle Markov network [Wei01]. Lastly, a recent, and somewhat mystifying, extension and application of belief propagation in the realm of satisfiability called survey propagation appears successful for some difficult instances of the satisfiability problem [BMZ03].

Chapter 2

Preliminaries

This chapter describes some basic mathematical knowledge used throughout the thesis. To ensure a clear understanding of all the notation used herein, there are sections describing definitions of basic probability, and graph theory.

2.1 Probability

Definition A *probability distribution* $P : S \to \mathbb{R}$ is a function over a finite sample space of possible outcomes Ω and an event set $S \subseteq \mathcal{P}(\Omega)$ which contains the trivial event Ω and empty event $\emptyset$ and is closed under union and complementation. P has three fundamental properties. For any events $s, t \in S$:

$P(s) \geq 0$, $P(\Omega) = 1$, and if $s \cap t = \emptyset$, then $P(s \cup t) = P(s) + P(t)$. Also, let $P(s, t) \equiv P(s \cap t)$.

Definition If $P(t) > 0$, the *conditional probability* of an event s given an event t is defined as $P(s|t) = P(s, t)/P(t)$. Bayes's Rule follows from this definition, in particular $P(s|t) = P(t|s)P(s)/P(t)$.

Definition Events s and t are *independent* in P denoted $(s \perp t)$ if $P(s|t) = P(s)$. We denote s and t being *conditionally independent* given u in P as $(s \perp t|u)$, which holds if $P(u) = 0$ or $P(s|t, u) = P(s|u)$.

Definition For a probability distribution P over a sample space Ω and event set S, a *discrete random variable* X is a function mapping Ω to some finite set of values such that the set $(X = x) \equiv \{v \in \Omega : X(v) = x\}$ is in the set of events S for all values x. Moreover, unless otherwise specified, $P(x)$ is shorthand for $P(X = x)$.

Definition The *joint distribution* of random variables $\boldsymbol{X} = \{X_1, X_2, \ldots, X_n\}$ is the distribution over all events involving all of these variables. This distribution is $P(X_1 = x_1, \ldots, X_n = x_n) \equiv P(\boldsymbol{X} = \mathbf{x}) \equiv P(\boldsymbol{x})$ for values $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbf{X}$, where $\mathbf{X}$ denotes the set of valid values for variables $\boldsymbol{X}$. Also, if $S \subseteq \boldsymbol{X}$, then $\mathbf{X_S}$ denotes the set of valid values for variables S. We may also use $P(\boldsymbol{X}) = P(X_1, \ldots, X_n)$ to denote the joint distribution.

Definition The *marginal distribution* over a random variable X_i is the distribution over all events involving X_i in some joint distribution of random variables $\boldsymbol{X}$, i.e. $P(X_i = x_i)$ for all possible values of x_i. It is typically computed from a joint distribution over several random variables, for instance from a joint distribution of two random variables, $P(x) = \sum_y P(x, y)$.

Definition Similarly, the *conditional distribution* over a random variable X given that Y is known is the distribution over all events involving X given that particular value of Y, denoted by $P(X|Y = y)$. $P(X|Y)$ denotes the set of all conditional distributions $P(X|Y = y)$.

Definition For random variables X, Y, Z, X is *independent* of Y given Z in P, denoted $X \perp Y|Z$, if $(X = x) \perp (Y = y)|(Z = z)$ for all values of X, Y, Z.

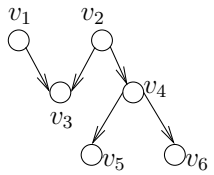

Fig. 2.1: A dag.

2.2 Graphs

The following graph definitions consider only simple graphs with no self-loops.

Definition A *graph* $G = (V, E)$ is a set of vertices/nodes V and a set of undirected edges E between pairs of distinct vertices. A *directed graph* is the same as a graph except that the edges are directed, i.e. $(u, v) \neq (v, u)$.

Definition There is a *path* from u to v in a graph G if the following edges are in $E(G)$, for some vertices $X_1, \ldots, X_n \in V(G)$: $uX_1, X_1X_2, \ldots, X_nv$. There is a *cycle* in G if there is a non-empty path from u to u. A graph G is *connected* if for any vertices $u, v \in V$, there is a path from u to v. The *distance* $d(u, v)$ between two vertices u, v is the length of, or the number of edges in the shortest path between u and v. The *diameter* $diam(G)$ of graph G is the maximum distance between two vertices in G.

Definition For any vertex $u \in V(G)$ of a graph G, $N(u) = \{v \in V | uv \in G\}$ is the set of *neighbours*. In a directed acyclic graph G, $Pa(u) = \{v \in V | (v, u) \in G\}$ is the set of *parents* and $Ch(u) = \{v \in V | (u, v) \in G\}$ is the set of *children*. Also, the set of *descendants* $Desc(u)$ is the set of vertices in the subgraph rooted at u, excluding u. The set of *non-descendents* is $NonDesc(u) = V(G) \setminus Desc(u) \setminus \{u\}$. For instance, in figure 2.1, $Desc(v_4) = \{v_5, v_6\}$ and $NonDesc(v_4) = \{v_1, v_2, v_3\}$.

Definition A graph G is a *tree* if G is connected and has no cycles. A graph G is *complete* if for all vertices $u, v \in V$, $uv \in E$. A *subgraph* $H = (V_H, E_H)$ of graph $G = (V, E)$ is a subset of vertices $V_H \subseteq V$ and a subset of edges $E_H \subseteq E$ where E_H is the set of all edges in E with both endpoints in V_H. An *induced subgraph* $G[A]$ is the subgraph on the subset of vertices $A \subseteq V(G)$ with edges $E(G[A])$ being all edges of G having both endpoints in A.

Definition A *clique* is a maximal complete subgraph of G, or the set of vertices in the said subgraph. That is, if any vertex in G is added to a clique, the resulting induced subgraph is no longer a clique. A *subclique* is any complete subgraph of G. An *X, Y-bipartite graph* is a graph $G = (V, E)$ where $V = X \cup Y$ and for all $uv \in E$, either $u \in X, v \in Y$ or $u \in Y, v \in X$.

Chapter 3

Graphical models

Three graphical models, Markov random fields, Bayesian networks, and factor graphs feature prominently in statistics, artificial intelligence, and physics, and each has their virtues. Though not identical, the design of both Markov random fields and Bayesian networks hinge on a desire to represent dependences of random variables in a graph. Highlighting this is the fact that the definitions of these two models centre on the independences of a joint probability distribution over these random variables. Interestingly, the form of the joint distribution for both of these models can be expressed as a product of simpler functions, which gives rise to the factor graph model, where the notion of representing a distribution is generalised to some generic global function $g(\mathbf{x})$ expressible as a product of simpler functions called factors.

Definition The set of *factors* is $F = \{f_1, \ldots, f_m\}$ and $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ is the set of *variables*. A factor $f_j : \mathbf{X_j} \to [0, k]$ is a function where $\boldsymbol{X_j} \subseteq \boldsymbol{X}$, for some $k \in \mathbb{R}, k > 0$, and $\mathbf{X_j}$ denotes the set of valid values for variables $\boldsymbol{X_j}$. As $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbf{X}$ are the values for all variables in $\boldsymbol{X}$, define $\mathbf{x_j} = (x_{i_1}, x_{i_2}, \ldots, x_{i_{|\mathbf{X_j}|}}) \in \mathbf{X_j}$ to be the values of the arguments $\boldsymbol{X_j}$ of factor f_j.

For instance, the factors of Bayesian networks are probabilities, so $k = 1$ in that case.

Example For global function $g(\mathbf{x}) = f_1(x_1, x_3)f_2(x_2, x_3) = f_1(\mathbf{x_1})f_2(\mathbf{x_2})$, the variables are $\boldsymbol{X} = \{X_1, X_2, X_3\}$, the factors are $F = \{f_1, f_2\}$, the arguments of f_1 are $\boldsymbol{X_1} = \{X_1, X_3\}$, and the arguments of f_2 are $\boldsymbol{X_2} = \{X_2, X_3\}$. □

The remainder of this section describes these pervasive models and their relationship without indulging in excessive detail. For both Markov random fields and Bayesian networks, we provide a basic understanding and fairly simple complete proofs regarding their joint distributions. This framework sufficiently motivates their generalisation to the factor graph model, which, in the context of belief propagation and inference, is an ideal model. Finally, we explore the relationship between these three models.

3.1 Markov random fields

Originating from the field of statistics [Ish81], Markov random fields (MRFs) are undirected graphs whose variables represent the random variables in $\boldsymbol{X}$ and edges represent dependences. The global function $g(\mathbf{x})$ is the joint probability distribution $P(\mathbf{x})$, and the factors are the potential functions of cliques, where $\boldsymbol{X_j} \in C$, where C is the set of cliques. A notable application of Markov random fields is computer vision. In 1984, Geman and Geman [GG84] helped initiate the usage of Markov random fields in computer vision, and thereafter, computer vision research involving MRFs has burgeoned.

Definition A *Markov random field* (MRF) is a pair (G, P) consisting of a graph $G = (\boldsymbol{X}, E)$ and a probability distribution P if each node $X_i \in \boldsymbol{X}$ is a random variable

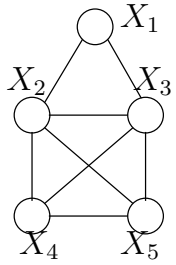

Fig. 3.1: Markov random field with $P(\mathbf{x}) = f_{C_1}(x_1, x_2, x_3) f_{C_2}(x_2, x_3, x_4, x_5)$.

and $P(\mathbf{x})$ satisfies the local Markov property for all $X_i \in \boldsymbol{X}$:

$$P(X_i | \boldsymbol{X} \setminus \{X_i\}) = P(X_i | N(X_i)). \tag{3.1}$$

In other words, every variable is independent of its non-neighbours given its neighbours, for example, see figure 3.1. One can also define a Markov random field using other Markov properties, but the local Markov property is the simplest one and these properties are equivalent precisely for positive distributions [KF02, pp. 168-178]. Positive distributions are also sufficient to express the distribution of an MRF in the following convenient form.

Definition A probability distribution P is a *Gibbs distribution* over G if it has the following form:

$$P(\mathbf{x}) = \alpha \prod_{A \in C} f_A(\mathbf{x_A}), \tag{3.2}$$

where C is the set of all cliques in G and α is the normalisation constant. Another form of the Gibbs distribution is the *log-linear model*, where if $f_A(\mathbf{x_A}) = \exp(-h_A(\mathbf{x_A}))$, then:

$$P(\mathbf{x}) = \alpha \exp\left(- \sum_{A \in C} h_A(\mathbf{x_A}) \right). \tag{3.3}$$

The equivalence of the distribution of an MRF and the Gibbs distribution given a positive distribution is known as the Hammersley-Clifford theorem [Pea88, Lau96,

KF02]. The Gibbs distribution can be parameterised in yet another form to facilitate the proof of this theorem. Note that for simplicity, the following definition uses 0 for fixed variables, but in fact any fixed assignment of variables is possible. The following standard proof, stated slightly more concisely, first shows that this form is equivalent to the Gibbs distribution, which subsequently helps prove the Hammersley-Clifford theorem [Lau96, pp. 32-37].

Definition Let $q : \mathbf{X} \to \mathbb{R}$ be a function where 0 is a valid value for each variable. For each subset of variables $A \subseteq \boldsymbol{X}$ define $q_A(\mathbf{x}) = q(\mathbf{y_A})$ where $y_{A,i} = \begin{cases} x_i & \text{if } X_i \in A; \\ 0 & \text{otherwise.} \end{cases}$ The *inclusion-exclusion function* (or canonical energy function [KF02]) is

$$h_A(\mathbf{x}) = \sum_{B \subseteq A} (-1)^{|A \setminus B|} q_B(\mathbf{x}). \tag{3.4}$$

The reason for calling this function the inclusion-exclusion function is its similarity to the inclusion-exclusion principle in combinatorics. Moreover, the following key proposition contains a variant of the Möbius inversion lemma, which is useful for proving the Hammersley-Clifford theorem.

Proposition 3.1.1 *The inclusion-exclusion function has the following properties.*
(i) The function h_A depends only on values of variables in A.
(ii) If $x_i = 0$ for some $X_i \in A$, then $h_A(\mathbf{x}) = 0$.
(iii) (Möbius Inversion Lemma) $q(\mathbf{x}) = \sum_{A \subseteq X} h_A(\mathbf{x})$.

Proof (i) follows from the definition: any value for a variable not in A is replaced with the fixed assignment 0. For (ii), consider some variable $X_i \in A$ and divide the subsets of A into sets A_i and $\bar{A}_i$ where the former is the set of subsets of variables in A containing X_i and the latter are those that do not contain X_i. Clearly $A_i \cup \bar{A}_i$ is the set of all subsets of A and every subset $\bar{B} \in \bar{A}_i$ has a counterpart $B \in A_i$ where

$B = \bar{B} \cup \{X_i\}$. Furthermore, $q_B(\mathbf{x}) = q_{\bar{B}}(\mathbf{x})$ because $x_i = 0$ and they have opposite sign in the summation of $h_A(\mathbf{x})$, so each such pair cancels, yielding $h_A(\mathbf{x}) = 0$.

(iii) follows by examining the sum over subsets B of functions $q_B(\mathbf{x})$ that arises from the definition of $h_A(\mathbf{x})$ in (3.4):

$$\sum_{A \subseteq X} h_A(\mathbf{x}) = \sum_{A \subseteq X} \sum_{B \subseteq A} (-1)^{|A \setminus B|} q_B(\mathbf{x}) = \sum_{B \subseteq X} \alpha_B q_B(\mathbf{x}).$$

Note that the coefficient α_B of each $q_B(\mathbf{x})$ is

$$\alpha_B = \sum_{\{A : B \subseteq A \subseteq X\}} (-1)^{|A \setminus B|} = \sum_{E \subseteq X \setminus B} (-1)^{|E|}.$$

$\alpha_X = 1$ because only the empty set contains $X \setminus X$. For any subset $B \subsetneq X$, the number of even subsets of $X \setminus B$ is equal to the number of odd subsets by a simple binomial coefficient argument (using $(1-1)^n$). Hence all coefficients are 0 except for α_X, and by definition $q_X(\mathbf{x}) = q(\mathbf{x})$. ■

With these tools in hand, it is easy to show that for any Markov random field with a positive distribution, this distribution must be a Gibbs distribution. The following proposition shows that the distribution is similar to the log-linear form of the Gibbs distribution. It is not quite the log-linear form because it depends on all subcliques rather than cliques, but it is easy to rearrange the terms to attain the desired factors of cliques.

Proposition 3.1.2 *Let P be a positive distribution of a Markov random field (G, P), and S be the set of all subcliques in G. Let $q(\mathbf{x}) = \log P(\mathbf{x})$. Then*

$$P(\mathbf{x}) = \exp\left(\sum_{A \in S} h_A(\mathbf{x})\right).$$

Proof Firstly, Proposition 3.1.1 (iii) yields $P(\mathbf{x}) = \exp\left(\sum_{A \subseteq X} h_A(\mathbf{x})\right)$. Thus it suffices to show that for any subset $A \subseteq \boldsymbol{X}$, if A is not a subclique, then $h_A(\mathbf{x}) = 0$.

Let $A \subseteq \boldsymbol{X}$ not be a subclique, and consider non-neighbours $X_i, X_j \in A$. Consider subsets $B, \bar{B} \subseteq A$ where $X_i \notin \bar{B}$ and $B = \bar{B} \cup \{X_i\}$. Note that the set of all such subsets B and $\bar{B}$ is the set of all subsets of A. Since the size of B and $\bar{B}$ differs by one, they appear as $\pm(q_B(\mathbf{x}) - q_{\bar{B}}(\mathbf{x}))$ in $h_A(\mathbf{x})$. Let $\mathbf{y}$ be the assignment of all variables except X_i and X_j, where $y_k = \begin{cases} x_k & \text{if } X_k \in A; \\ 0 & \text{otherwise.} \end{cases}$

Thus:

$$\begin{aligned} q_B(\mathbf{x}) - q_{\bar{B}}(\mathbf{x}) &= \log \tfrac{P(X_i=x_i, X_j=x_j, \mathbf{y})}{P(X_i=0, X_j=x_j, \mathbf{y})} \\ &= \log \tfrac{P(X_i=x_i | X_j=x_j, \mathbf{y}) P(X_j=x_j, \mathbf{y})}{P(X_i=0 | X_j=x_j, \mathbf{y}) P(X_j=x_j, \mathbf{y})} \\ &= \log \tfrac{P(X_i=x_i | X_j=0, \mathbf{y})}{P(X_i=0 | X_j=0, \mathbf{y})}. \end{aligned}$$

The last equality follows from the local Markov property: since $X_j \notin N(X_i)$, it follows that $X_i \perp X_j | N(X_i)$, hence the value of X_j is irrelevant. This holds for all subsets $B, \bar{B}$ of A, which constitute all terms in the summation of $h_A(\mathbf{x})$.

Hence with slight abuse of notation,

$$h_A(X_i = x_i, X_j = x_j, \mathbf{y}) = h_A(X_i = x_i, X_j = 0, \mathbf{y})$$

and since $X_j \in A$, Proposition 3.1.1 (ii) applies, so $h_A(\mathbf{x}) = 0$. ■

Finally, the main theorem requires one simple property of conditional independence.

Fact For a positive distribution P and disjoint sets of variables $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ with corresponding values $\mathbf{x} \in \mathbf{X}, \mathbf{y} \in \mathbf{Y}, \mathbf{z} \in \mathbf{Z}$, $\boldsymbol{X} \perp \boldsymbol{Y} | \boldsymbol{Z}$ iff there exist functions f, g such that $P(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}, \mathbf{z}) g(\mathbf{y}, \mathbf{z})$.

Proving the only if direction is trivial, and the if direction is true because

$$P(\mathbf{x}, \mathbf{z}) = f(\mathbf{x}, \mathbf{z}) \sum_{\mathbf{y}} g(\mathbf{y}, \mathbf{z}), P(\mathbf{y}, \mathbf{z}) = g(\mathbf{y}, \mathbf{z}) \sum_{\mathbf{x}} f(\mathbf{x}, \mathbf{z}), P(\mathbf{z}), \text{ so,}$$

$$P(\mathbf{x}, \mathbf{z}) = \sum_{\mathbf{x}} f(\mathbf{x}, \mathbf{z}) \sum_{\mathbf{y}} g(\mathbf{y}, \mathbf{z}), \text{ thus:}$$

$$P(\mathbf{x}|\mathbf{y},\mathbf{z})P(\mathbf{y}|\mathbf{z}) = P(\mathbf{x},\mathbf{y},\mathbf{z}) = f(\mathbf{x},\mathbf{z})g(\mathbf{y},\mathbf{z}), \text{ so}$$
$$f(\mathbf{x},\mathbf{z})g(\mathbf{y},\mathbf{z}) = \frac{f(\mathbf{x},\mathbf{z})\sum_{\mathbf{x}} f(\mathbf{x},\mathbf{z})g(\mathbf{y},\mathbf{z})\sum_{\mathbf{y}} g(\mathbf{y},\mathbf{z})}{P(\mathbf{z})} = P(\mathbf{x}|\mathbf{z})P(\mathbf{y},\mathbf{z}),$$

and so $P(\boldsymbol{X}|\boldsymbol{Y},\boldsymbol{Z}) = P(\boldsymbol{X}|\boldsymbol{Z})$.

Theorem 3.1.3 *(Hammersley-Clifford) Let (G,P), $G = (\boldsymbol{X},E)$ be an MRF and the distribution P be positive over $\boldsymbol{X}$. G is a Markov random field with distribution P iff P is a Gibbs distribution over G.*

Proof The only if direction requires the results involving the inclusion-exclusion function. Let $q(\mathbf{x}) = \log P(\mathbf{x})$, which is valid because P is positive. By Proposition 3.1.2, $P(\mathbf{x}) = \exp\left(\sum_{A\in S} h_A(\mathbf{x})\right)$. By Proposition 3.1.1 (i), h_A depends only on variables in A. Thus $h_A(\mathbf{x}) = h_A(\mathbf{x_A})$. Then

$$\begin{aligned} P(\mathbf{x}) &= \exp\left(\textstyle\sum_{B\in S} h_B(\mathbf{x_B})\right) \\ &= \textstyle\prod_{B\in S\setminus C} \exp(h_B(\mathbf{x_B})) \prod_{A\in C} \exp(h_A(\mathbf{x_A})) \\ &= \textstyle\prod_{A\in C} f_A(\mathbf{x_A}). \end{aligned}$$

By definition, each subclique $B \in S \setminus C$ is contained in some clique $A \in C$, so every B can be grouped with a clique in the second product to form the factors, validating the last equality. For instance, one factor could include all of its subcliques: $f_A(\mathbf{x_A}) = \prod_{\{B\in S: B\subseteq A\in C\}} \exp(h_A(\mathbf{x_A}))$. Therefore P is a Gibbs distribution.

The if direction requires only the conditional independence fact: Consider variable X_i and its neighbours $N(X_i)$. Obviously any clique c containing X_i must only contain variables in $N(X_i)$ because X_i must be connected to all variables in c. Thus separating the Gibbs distribution P into two products of factors $\boldsymbol{Y} = \boldsymbol{X} \setminus (\{X_i\} \cup N(X_i))$ and $\boldsymbol{X} \setminus \boldsymbol{Y}$ produces:

$$f(x_i, \mathbf{x_N}) = \prod_{A\in C: X_i\in A} f_A(\mathbf{x_A}),\ g(\mathbf{x_Y}, \mathbf{x_N}) = \prod_{A\in C: X_i\notin A} f_A(\mathbf{x_A})$$
$$P(\mathbf{x}) = \alpha f(x_i, \mathbf{x_N}) g(\mathbf{x_Y}, \mathbf{x_N}),$$

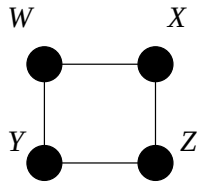

Fig. 3.2: Markov random field with non-positive distribution.

where $\mathbf{x} \in \mathbf{X}$, $\mathbf{x_N} \in \mathbf{X_{N(X_i)}}$, and $\mathbf{x_Y} \in \mathbf{X_Y}$. Thus $X_i \perp \boldsymbol{Y} | N(X_i)$, namely X_i is independent of non-neighbours given $N(X_i)$ hence verifying the local Markov properties defined in (3.1). ■

Lastly, a simple example reveals that if a Markov random field has a non-positive distribution P, then P might not be a Gibbs distribution.

Example [Lau96, p. 37] Consider the graph G in Figure 3.2 with binary variables W, X, Y, Z and the following distribution. $P(w, x, y, z) = 0$ except in the following cases: $(0, 0, 0, 0)$, $(0, 0, 0, 1)$, $(0, 0, 1, 1)$, $(0, 1, 1, 1)$, $(1, 1, 1, 1)$, $(1, 0, 0, 0)$, $(1, 1, 0, 0)$, and $(1, 1, 1, 0)$, where $P(w, x, y, z) = 1/8$.

Then it is easy to check that $P(W|X, Z) = P(W|X, Y, Z)$ so $W \perp Y | X, Z$. Similarly $P(X|W, Y) = P(X|W, Y, Z)$ so $X \perp Z | W, Y$. Thus G is a Markov random field. Using the same argument in Lauritzen's book, assume that P is a Gibbs distribution:

$$1/8 = P(0, 0, 0, 0) = f_{WX}(0, 0) f_{XZ}(0, 0) f_{YZ}(0, 0) f_{WY}(0, 0)$$

$$0 = P(0, 0, 1, 0) = f_{WX}(0, 0) f_{XZ}(0, 0) f_{YZ}(1, 0) f_{WY}(0, 1)$$

$$1/8 = P(0, 0, 1, 1) = f_{WX}(0, 0) f_{XZ}(0, 1) f_{YZ}(1, 1) f_{WY}(0, 1)$$

Thus $f_{YZ}(1, 0) f_{WY}(0, 1) = 0$ from the first two equalities. From the third equality, $f_{WY}(0, 1) \neq 0$ so $f_{YZ}(1, 0) = 0$. Finally,

$$1/8 = P(1, 1, 1, 0) = f_{WX}(1, 1) f_{XZ}(1, 0) f_{YZ}(1, 0) f_{WY}(1, 1)$$

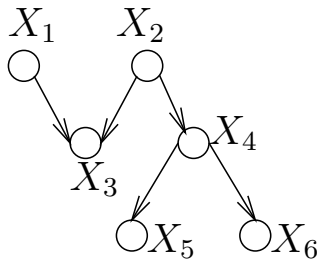

Fig. 3.3: Joint distribution $P(\mathbf{x}) = P(x_1)P(x_2)P(x_3|x_1, x_2)P(x_4|x_2)P(x_5|x_4)P(x_6|x_4)$ of a Bayesian network

which means $f_{YZ}(1, 0) \neq 0$, hence achieving a contradiction, so P is not a Gibbs distribution. □

3.2 Bayesian networks

As with Markov random fields, good theoretical analysis pertaining to Bayesian networks is bountiful [KF02, Nea90, Pea88].

Bayesian networks are directed acyclic graphs where variables represent the random variables in $\boldsymbol{X}$ and edges represent dependences. Bayesian networks are particularly common in artificial intelligence, where they are used in expert systems [Nea90, LS88].

Like Markov random fields, the global function $g(\mathbf{x})$ is the joint probability distribution $P(\mathbf{x})$, but unlike Markov random fields, the factors are probability functions that depend on a variable and its parents, i.e. $f_j(\mathbf{x_j}) = P(x_j|\mathbf{pa_j})$ where $\mathbf{x_j} \in \mathbf{X_j}$ gives the values for $\boldsymbol{X_j} = \{X_j, Pa(X_j)\}$ and $\mathbf{pa_j} \in \mathbf{X_{Pa(X_j)}}$ gives the values for $Pa(X_j)$. For instance, see Figure 3.3.

Definition A *Bayesian network* is a pair (G, P) consisting of a probability distribution P, and a directed acyclic graph $G = (\boldsymbol{X}, E)$ where the nodes represent random

variables and the directed edges represent dependencies, that is if $(X_1, X_2) \in E$, then X_1 influences directly X_2. More specifically, $(X_i \perp NonDesc(X_i)|Pa(X_i))$, for each variable X_i.

Bayesian networks also have a factorised distribution similar to the Gibbs distribution in Markov random fields under slightly more general conditions.

Definition For any DAG $G = (\boldsymbol{X}, E)$, a distribution P *factorises according to* G if $P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|\mathbf{pa_i}) = \prod_{i=1}^{n} f_i(\mathbf{x_i})$.

Theorem 3.2.1 *For any distribution P and DAG $G = (\boldsymbol{X}, E)$, (G, P) is a Bayesian network iff P factorises according to G.*

Proof The only if direction [KF02, Pea88] follows immediately from the definition of a Bayesian network. First order the variables so that all descendants X_j of a variable X_i satisfy $j > i$. Such an ordering is valid because G is acyclic. Then using basic facts of probability,

$$P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|x_1, \ldots, x_{i-1}) = \prod_{i=1}^{n} P(x_i|\mathbf{pa_i}).$$

The last equality arises from the given independences $X_i \perp NonDesc(X_i)|Pa(X_i)$ because our ordering means that the variables $X_1, \ldots X_{i-1}$ contain all parents of X_i, and no descendants of X_i.

Proof of the if direction requires that the basic Bayesian network independences hold, that is, $P(X_i|\boldsymbol{X} \setminus \{X_i\} \setminus Desc(X_i)) = P(X_i|Pa(X_i))$.

$$
\begin{aligned}
P(X_i|\boldsymbol{X} \setminus \{X_i\} \setminus Desc(X_i)) &= \frac{P(\boldsymbol{X} \setminus Desc(X_i))}{P(X \setminus \{X_i\} \setminus Desc(X_i))} \\
&= \frac{\sum_{Desc(X_i)} \prod_{j=1}^{n} P(X_j|Pa(X_j))}{\sum_{X_i} \sum_{Desc(X_i)} \prod_{j=1}^{n} P(X_j|Pa(X_j))} \\
&= \frac{P(X_i|Pa(X_i)) \sum_{Desc(X_i)} \prod_{\{j:X_j \in Desc(X_i)\}} P(X_j|Pa(X_j))}{\sum_{X_i} \sum_{Desc(X_i)} P(X_i|Pa(X_i)) \prod_{\{j:X_j \in Desc(X_i)\}} P(X_j|Pa(X_j))} \\
&= \frac{P(X_i|Pa(X_i))}{\sum_{X_i} P(X_i|Pa(X_i))} = P(X_i|Pa(X_i)).
\end{aligned}
$$

The second equality holds because P factorises according to G. The last equality holds because summing over all values of X_i means $\sum_{X_i} P(X_i|Pa(X_i)) = 1$. Note that if X_i is a leaf, $Desc(X_i)$ is empty so there is no summation over values of variables in $Desc(X_i)$. Otherwise, it suffices show that the second-last equality holds by induction, that is $\sum_{Desc(X_i)} \prod_{\{j:X_j \in Desc(X_i)\}} P(X_j|Pa(X_j)) = 1$.

If all children of X_i are leaves (i.e. max distance from X_i is $d = 1$), exactly one factor in the product contains $X_j \in Desc(X_i)$ for each descendant. Thus the summation can be separated:

$$
\sum_{Desc(X_i)} \prod_{\{j:X_j \in Desc(X_i)\}} P(X_j|Pa(X_j)) = \prod_{\{j:X_j \in Desc(X_i)\}} \sum_{X_j} P(X_j|Pa(X_j)) = 1.
$$

For max distance d, we can apply this idea for the leaves $L \subseteq Desc(X_i)$:

$$
\begin{aligned}
&\textstyle\sum_{Desc(X_i)} \prod_{\{j:X_j \in Desc(X_i)\}} P(X_j|Pa(X_j)) \\
&= \textstyle\sum_{Desc(X_i) \setminus L} \left[\prod_{\{j:X_j \in Desc(X_i) \setminus L\}} P(X_j|Pa(X_j)) \sum_{L} \prod_{\{k:X_k \in L\}} P(X_k|Pa(X_k)) \right] \\
&= \textstyle\sum_{Desc(X_i) \setminus L} \prod_{\{j:X_j \in Desc(X_i) \setminus L\}} P(X_j|Pa(X_j)).
\end{aligned}
$$

The remaining summation is for all descendants of max distance $d-1$, which is 1 by induction. ■

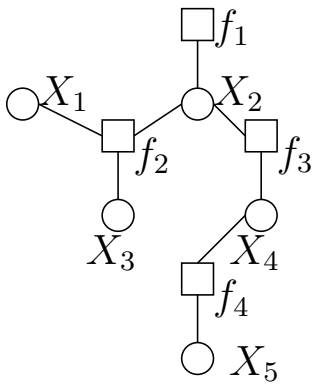

Fig. 3.4: Factor graph with $P(\mathbf{x}) = f_1(x_2)f_2(x_1, x_2, x_3)f_3(x_2, x_4)f_4(x_4, x_5)$.

3.3 Factor graphs

The notion of representing a factorised distribution is a common thread between Bayesian networks and Markov random fields, and factor graphs seek to generalise this idea. In factor graphs, the factorised distribution is built into the definition. Moreover, instead of limiting the model to represent just probability distributions, factor graphs represent any global function that is a product of local factors. Thus this model is a convenient and tidy formulation for a multitude of applications that includes both Bayesian networks and Markov random fields. Kschischang et al. provides an excellent survey of factor graphs that discusses many applications in detail [KFL01].

Definition For a set of variables $\boldsymbol{X}$ and a set of factors F, a factor graph G is an $\boldsymbol{X}, F$-bipartite graph that represents the factorisation

$$g(\mathbf{x}) = \prod_{j=1}^{m} f_j(\mathbf{x_j}). \tag{3.5}$$

Furthermore, for any variable X_i, factor f_j, $X_i f_j \in E(G)$ iff X_i is an argument of factor f_j. Hence the variables $\boldsymbol{X_j} = N(f_j)$ are arguments of f_j. For instance, see Figure 3.4.

If the global function is a distribution then it can also called a *factorised distribution* to distinguish it from the more specific case of a Gibbs distribution for Markov random fields. As noted, the factorised distribution is the definition of factor graphs, whereas the basis of the definitions of the other graphical models centres on their independences. Hence it is interesting to consider which independences hold in a factor graph distribution. Recall the following fact used to prove the Hammersley-Clifford theorem, which would be used as the starting point to determine the independences in a factor graph:

$$P(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}, \mathbf{z})g(\mathbf{y}, \mathbf{z}) \text{ iff } \mathbf{x} \perp \mathbf{y} | \mathbf{z}.$$

If the graph is a tree, then for any variable X_i, and factors $f_j, f_k \in N(X_i)$, notice that $\boldsymbol{X_j} \setminus \{X_i\} \perp \boldsymbol{X_k} \setminus \{X_i\} | X_i$. This yields the following form for the joint distribution of a factor tree.

Proposition 3.3.1 *For any factor tree G, the joint distribution is the following equation:*

$$P(\mathbf{x}) = \frac{\prod_{f_j \in F} P(\mathbf{x_j})}{\prod_{X_i \in \boldsymbol{X}} (P(x_i))^{|N(X_i)|-1}}.$$

Proof If all variables are of degree 1, this proposition is trivial. Otherwise, suppose $X_1, X_2, \ldots, X_d$ are variables of degree at least 2. Observe that every factor must be adjacent to such a variable because trees are connected, so $\cup_{i=1}^{d} N(X_i) = F$. If S is a set of factors, let $\mathbf{x_S}$ be values for all the arguments of the factors in S. Let $f_{j_1}, \ldots, f_{j_{|N(X_1)|}}$ be the factors containing X_1 and let $\mathbf{y_j}$ be values for variables in $N(f_j) \setminus \{X_1\}$. Then the right-hand side of the equation in the proposition equals

$$\left(P(\mathbf{y_{j_1}}, x_1) \prod_{f_j \in N(X_1) \setminus \{f_{j_1}\}} P(\mathbf{y_j} | x_1) \right) \frac{\prod_{f_j \in F : X_1 \notin N(f_j)} P(\mathbf{x_j})}{\prod_{X_i \in \boldsymbol{X} : i \neq 1} P(x_i)^{|N(X_i)|-1}}$$

$$= P(\mathbf{x_{N(x_1)}}) \frac{\prod_{f_j \in F : X_1 \notin N(f_j)} P(\mathbf{x_j})}{\prod_{X_i \in \boldsymbol{X} : i \neq 1} P(x_i)^{|N(X_i)|-1}}$$

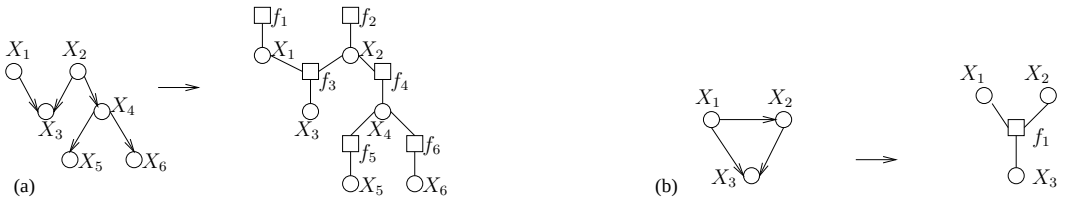

Fig. 3.5: (a) Converting a Bayesian network to a factor graph with one factor per variable. (b) Converting a Bayesian network to a factor graph with a clumped factor node

This equality is because $\mathbf{y_{j_k}} \perp \mathbf{y_{j_l}} | x_1$, where $k \neq l$. It is not hard to see that the continual application of this notion for each X_i will produce $P(\mathbf{x_F})$ because $\cup_{i=1}^{d} N(X_i) = F$.

■

3.4 Relationship between models

These three models can be more or less similar depending on whether we want to consider the precise independences of a graph or not. If we want only to maintain the global function from a particular model, factor graphs encompass both Markov random fields and Bayesian networks. The central difficulty with representing all independences of a Bayesian network in a corresponding factor graph is that a factor graph cannot capture the directed dependencies in a Bayesian network. Similarly, Bayesian networks cannot always represent all the non-directional independences of undirected graphical models because they are DAGs. Numerous works study these characteristics in detail [KF02, Pea88]. However, for computing the marginals, the only property that the graph needs to maintain is the factorised joint distribution or global function. Since there are three models, there are six possible conversions, and the following discusses them roughly in order of complexity.

Bayesian network→ factor graph

Firstly, Figure 3.5 (a) illustrates a simple procedure for converting a Bayesian network to a factor graph, thus preserving the precise factorisation given in the Bayesian network.

Create one factor node $f_i(\mathbf{x_i})$ for every variable X_i and connect it to X_i and the parents of X_i, so $f_i(\mathbf{x_i}) = P(x_i|\mathbf{pa_i})$ and this preserves the factorised distribution. Note that since the factors in factor graphs are more general than in Bayesian networks, one can also clump several factors from Bayesian networks into a single factor in a factor graph. For instance, in Figure 3.5 (b), the Bayesian network with distribution $P(x_1, x_2, x_3) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)$ becomes $g(x_1, x_2, x_3) = f_1(x_1, x_2, x_3)$, where $f_1(x_1, x_2, x_3) = P(x_1, x_2, x_3)$. This technique is sometimes advantageous because it removes loops and cycles, which often cause difficulty in computing marginals in Section 4.1. Unfortunately, it does so at the cost of amplifying the complexity of the individual factors. In any case, converting a Bayesian network to a factor graph is clearly efficient, because each node in the Bayesian network only needs to be examined a constant number of times.

Markov random field→ factor graph

Similarly, factor graphs can represent all Markov random fields, for instance see Figure 3.6. The procedure simply calls for one factor node f_{C_i} for each clique C_i with edges to each variable in C_i. It is apparent that while the number of vertices increases, this procedure will usually reduce the connectedness of the graph. Assuming that the set of cliques of the Markov random field is known, this conversion is obviously efficient.

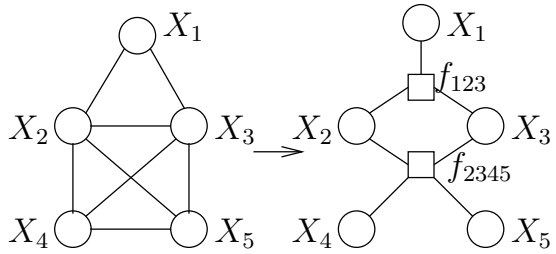

Fig. 3.6: Converting a Markov random field to a factor graph

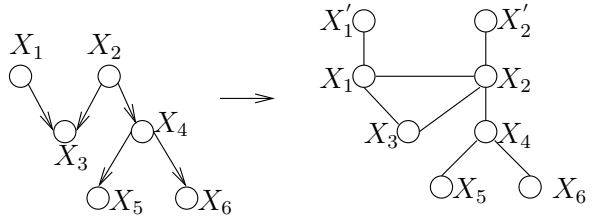

Fig. 3.7: Converting a Bayesian network to a Markov random field, from distribution $P(x_1)P(x_2)P(x_3|x_1,x_2)P(x_4|x_2)P(x_5|x_4)P(x_6|x_4)$ to distribution $f_1(x_1,x_1^{'})f_2(x_2,x_2^{'})f_{123}(x_1,x_2,x_3)f_{24}(x_2,x_4)f_{45}(x_4,x_5)f_{46}(x_4,x_6)$.

Bayesian network→ Markov random field

This conversion requires each variable and its parents to become cliques, and each leaf node to be its own clique. Hence the former case involves the addition of edges between the parents of each node to form cliques $C = \{X_i\} \cup Pa(X_i)$. The latter situation needs a dummy node $X_i^{'}$ attached to a leaf node X_i, with its factor set to $f_i(x_i,x_i^{'}) = P(x_i)$ if $x_i = x_i^{'}$, and 0 otherwise. Again, efficiency is apparent. For an example of this method, see Figure 3.7.

Factor graph → Markov random field

The Gibbs distribution property of MRFs is an obstacle in the conversion, and resolving these main issues results in the algorithm. To help illustrate this process, see Figure 3.8. In general, create a subclique for the variables of each factor. Then con-

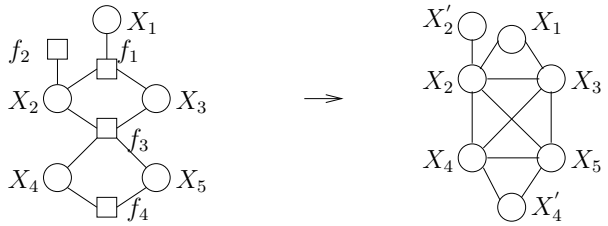

Fig. 3.8: Converting a factor graph to a Markov random field from distribution $f_1(x_1, x_2, x_3)f_2(x_2)f_3(x_2, x_3, x_4, x_5)f_4(x_4, x_5)$ to distribution $f_{123}(x_1, x_2, x_3)f_2(x_2, x_2')f_{2345}(x_2, x_3, x_4, x_5)f_{45}(x_4, x_4', x_5)$.

sider the case of single-node factors, for which there are no corresponding subcliques. In this situation, create a dummy node X_i' connected to each single-node factor node $f_j(x_i)$. The other problem is when $|F| \neq |C|$, which hinders the aim of converting each factor into a subclique. Observe that $|F| < |C|$ is impossible because each factor node represents a subclique, so the number of cliques is at most the number of factors. Hence $|F| > |C|$. Each excess factor f_j introduces a subclique C contained in some larger clique, so again create a dummy node X_i' of node $X_i \in C$ and connect it to every node in C, including X_i. In both the single-node factor case and the subclique case, set the factor $f_C(\mathbf{x_C}, x_i') = f_j(\mathbf{x_j})$ if $x_i = x_i'$, and 0 otherwise. The only efficiency issue is that the resulting joint function for the new Markov random field must be normalised because the global function of the factor graph might not be normalised.

Markov random field→ Bayesian network

The conversion from an MRF to a Bayesian network can be done by creating a node X_C representing clique C in the set of cliques of the MRF and making every $X_i \in C$ a child of X_C, for instance see Figure 3.9. The factor $P(\mathbf{x_C})$ corresponds to the MRF factor $f_C(\mathbf{x_C})$, and the "probability" $P(x_i|\mathbf{pa_i}) = 1$ when the value x_i of X_i is the same in each clique containing X_i, and 0 otherwise. Unfortunately, due to the nature

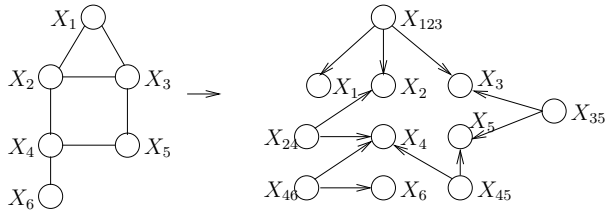

Fig. 3.9: Converting an MRF into a Bayesian network, from $f_{123}(x_1, x_2, x_3)f_{24}(x_2, x_4)f_{35}(x_3, x_5)f_{45}(x_4, x_5)f_{46}(x_4, x_6)$ to $P(x_{123})P(x_{24})P(x_{35})P(x_{45})P(x_{46})P(x_1|x_{123})P(x_2|x_{123}, x_{24})P(x_3|x_{123}, x_{35})$ $P(x_4|x_{24}, x_{46})P(x_5|x_{35}, x_{45})P(x_6|x_{46})$.

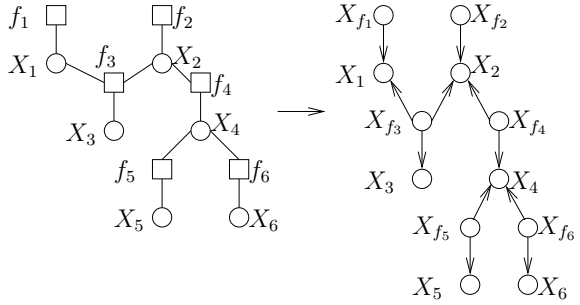

Fig. 3.10: Converting a factor graph to a Bayesian network, from $f_1(x_1)f_2(x_2)f_3(x_1, x_2, x_3)f_4(x_2, x_4)f_5(x_4, x_5)f_6(x_4, x_6)$ to $P(x_{f_1})P(x_{f_2})P(x_{f_3})P(x_{f_4})P(x_{f_5})P(x_{f_6})P(x_1|x_{f_1}, x_{f_3})P(x_2|x_{f_2}, x_{f_3}, x_{f_4})P(x_3|x_{f_3})$ $P(x_4|x_{f_4}, x_{f_5}, x_{f_6})P(x_5|x_{f_5})P(x_6|x_{f_6})$

of MRF factors, the resulting Bayesian factors might not be normalised.

Factor graph → Bayesian network

To convert a factor graph into a Bayesian network, create new nodes X_{f_j} representing the factors with children $X_i \in N(f_j)$, with the new node factor $P(\mathbf{x_j})$ corresponding to $f_j(\mathbf{x_j})$, and $P(x_i|\mathbf{pa_i})$ with the same definition as the $MRF \rightarrow BN$ conversion. For clarity, see the illustration in Figure 3.10. As in the previous conversion, the Bayesian factors might not be normalised if the factor graph factors were not normalised.

Chapter 4

Solving marginalisation

The task of marginalisation is a fundamental problem in probability. For instance, it is often the main subtask of inference problems which encompass a wide range of applications. Why is the computation of the marginal function important? Given some joint probability distribution of many variables, we would often desire to ascertain the likelihood of particular subsets of variables, like in the oft-cited example of medical diagnosis expert systems [Pea88, LS88]. This section presents the problem and chiefly discusses one algorithmic solution called belief propagation that works on graphical models.

4.1 Marginalisation

Recall that the global function is

$$g(\mathbf{x}) = \prod_{f_j \in F} f_j(\mathbf{x_j}).$$

Computing a marginal distribution, i.e. probabilities, requires a normalisation

constant $Z = \sum_{\mathbf{x}} \prod_{f_j \in F} f_j(\mathbf{x_j})$ to yield the distribution

$$P(\mathbf{x}) = g(\mathbf{x})/Z.$$

Definition The *marginal function* and *distribution* of X_i are, respectively

$$z(x_i) = \sum_{\mathbf{x}: X_i = x_i} g(\mathbf{x}),$$

$$P(x_i) = \sum_{\mathbf{x}: X_i = x_i} P(\mathbf{x}) = z(x_i)/Z.$$

Moreover, the *marginal function of a factor* f_j, i.e. the marginal of $N(f_j)$, is the sum of the global function over all possible values of variables in $\boldsymbol{X} \setminus N(f_j)$. The notation $N(f_j) = \mathbf{x_j}$ means that $\mathbf{x_j}$ gives an assignment to each variable in $N(f_j)$. So the marginal of f_j is

$$z(\mathbf{x_j}) = \sum_{\mathbf{x}: N(f_j) = \mathbf{x_j}} g(\mathbf{x}).$$

The easiest method of marginalisation is by brute force. Unfortunately, it is apparent that this method requires exponential time in the number of variables. For instance, if the variables are binary, then computing the marginal requires $2^{|X|-1}$ computations of the global function $g(\mathbf{x})$.

In fact, complexity analysis does not offer any respite. Complexity analysis shows that computing $P(X = x)$ over a graphical model is NP-hard [Coo90] which uses a reduction from 3SAT. In fact, the decision version of this problem is PP-complete [LMP01] and similarly, the functional version is #P-complete [Rot96]. Even more unfortunate is the extension of this result by Dagum and Luby [DL93] that shows approximate inference within some relative error in Bayesian networks is also NP-hard. It is not difficult to see that these results also apply to factor graphs. One simply converts the Bayesian network construction used in the reduction from 3SAT into a corresponding factor graph construction. Then one generalises the conditional

probability definitions in the Bayesian network into factor definitions and precisely the same result follows. The construction of a Bayesian network for a CNF formula will be discussed in Section 4.4 for the inference problem of maximisation. The construction is the same as the one used in marginalisation, and because the hardness proof for marginalisation is much more common, this section includes a similar proof for the slightly different problem of maximisation.

Despite this bad news, there are several techniques that can often improve performance in a factorised distribution. Some exact methods are variable elimination [KF02], belief propagation [Mac03], and the junction-tree algorithm [LS88]. There are also many approximate algorithms including variational methods [YFW01], loopy belief propagation [MWJ99], and tree reparameterisations [WJW03].

4.2 Belief propagation in factor graphs

Known as belief propagation, probability propagation, and the sum-product algorithm, this method is primarily used to compute the marginal function of a global function that can be expressed as a product of local factors. The algorithm functions by passing real-valued functions called *messages* between neighbouring nodes, thereby exploiting the structure of the global function. Belief propagation is an important algorithm that has demonstrated empirical success in solving inference problems like marginalisation [HM97, MWJ99]. Being successful in a general probabilistic setting makes belief propagation particularly useful for a multitude of applications. These include turbocodes [MMC98], LDPC codes [RU01], the forward-backward algorithm, the Viterbi algorithm, the Kalman filter [KFL01], free energy approximation [YFW02], and the satisfiability problem [BMZ03].

Some preliminary work [IIW05, MK05, TJ01, Wei01] has been done to determine

the graphs on which belief propagation works well, but much remains to be done. The following discusses the algorithm in the factor graph model, first in the tree case for which the algorithm was initially designed and then extend it for usage in any factor graph. Subsequently, accounts of some basic correctness results, related problems, and belief propagation in Markov random fields and Bayesian networks ensue.

4.2.1 Algorithm for trees

`TreeBeliefProp`

Output: set of messages

begin

while *there is an unsent message* **do**

Send all unsent messages $u \to v$ that can be sent (i.e. u has received messages from all of its other neighbours), according to the message definitions (4.1) and (4.2).

end

end

In the tree case, start by sending messages from leaf nodes. An internal node u sends a message to a neighbour v if u has already received messages from all of its other neighbours. The message passing component terminates when there are no remaining messages to be sent. Moreover, the message definitions are fairly simple and in trees, the algorithm is provably correct and efficient. The next two definitions describe the messages that are passed between neighbours.

Definition The *variable message* $X_i \to f_j(x_i)$, or simply $X_i \to f_j$ with the value of x_i

implicit, is the message from a variable X_i to factor f_j (For leaf X_i, $X_i \to f_j(x_i) = 1$):

$$X_i \to f_j(x_i) = \prod_{f_k \in N(X_i) \setminus \{f_j\}} f_k \to X_i(x_i). \tag{4.1}$$

Definition The *factor message* $f_j \to X_i(x_i)$ or $f_j \to X_i$ is the message from a factor f_j to variable X_i (For leaf f_j, $f_i \to X_i(x_i) = f_j(x_i)$):

$$f_j \to X_i(x_i) = \sum_{\mathbf{x_j}: X_i = x_i} f_j(\mathbf{x_j}) \prod_{X_k \in N(f_j) \setminus \{X_i\}} X_k \to f_j(x_k). \tag{4.2}$$

Observe that for binary variables, the factor message is a summation of $2^{|N(f_j)|-1}$ terms. Thus in the worst case, transmitting every message once could potentially be as adverse as brute force marginalisation. However, such a network with a hefty set of neighbours is unlikely in the graphical model setting. Typically, it is acceptable to assume that factors have a tolerable number of neighbours, for instance when $|N(f_j)| \in O(\log|\boldsymbol{X}|)$, so that transmitting every message once can be done in polynomial-time in the number of variables. In trees, this algorithm will converge once every possible message has been sent so it is certainly efficient. But convergence in a single iteration is not always certain in general, thus suggesting the introduction of notation to represent the approximation to the marginal function. In trees, this approximation is precisely the marginal function.

Definition The *belief propagation approximation to the marginal function* $z(x_i)$ is

$$b(x_i) = \prod_{f_m \in N(X_i)} f_m \to X_i.$$

Definition The *approximation to the marginal function* $z(\mathbf{x_j})$ is $b(\mathbf{x_j})$:

$$b(\mathbf{x_j}) = f_j(\mathbf{x_j}) \prod_{X_i \in N(f_j)} X_i \to f_j$$

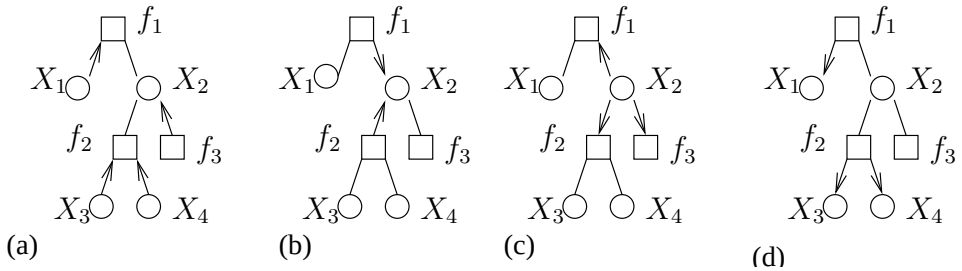

Fig. 4.1: Belief propagation example: (a) Step 1, (b) step 2, (c) step 3, and (d) step 4.

Thus one can roughly think of the variable message $X_n \to f_m$ as the known marginal of X_n without f_m and the factor message $f_m \to X_n$ as the known marginal function of X_n so far.

Definition The *belief propagation approximation to the marginal distribution* $P(x_i)$ is $B(x_i)$, and similarly, $B(\mathbf{x_j})$ approximates $P(\mathbf{x_j})$.

Correctness of belief propagation in trees is the simplest and most established correctness result. A simple, though somewhat lengthy, induction proof verifies this fact. First however, the ensuing example illustrates belief propagation in action and should provide some explanation for why belief propagation works in trees.

Example Given the graph G from Figure 4.1 and binary variables, belief propagation performs the following steps:

Step 1: Messages from leaves

$$(X_1 \to f_1) = (X_3 \to f_2) = (X_4 \to f_2) = 1, \quad (f_3 \to X_2) = f_3(x_2).$$

Step 2: Messages from non-leaves with at most one non-leaf neighbour

$f_1 \to X_2(x_2) = \sum_{\mathbf{x_1}:X_2=x_2} f_1(x_1, x_2)$,

$f_2 \to X_2 = \sum_{\mathbf{x_2}:X_2=x_2} f_2(x_2, x_3, x_4)$

Step 3: $X_2 \to f_1 = (f_2 \to X_2)(f_3 \to X_2) = \sum_{\mathbf{x_2}:X_2=x_2} f_2(x_2, x_3, x_4) f_3(x_2)$,

$X_2 \to f_2 = (f_1 \to X_2)(f_3 \to X_2) = \sum_{\mathbf{x_1}:X_2=x_2} f_1(x_1, x_2) f_3(x_2)$,

$$\begin{aligned} X_2 \to f_3 &= (f_1 \to X_2)(f_2 \to X_2) \\ &= \textstyle\sum_{\mathbf{x_1}:X_2=x_2} f_1(x_1, x_2) \sum_{\mathbf{x_2}:X_2=x_2} f_2(x_2, x_3, x_4) \\ &= \textstyle\sum_{\mathbf{x}:X_2=x_2} f_1(x_1, x_2) f_2(x_2, x_3, x_4) \end{aligned}$$

Step 4: $f_2 \to X_3 = \sum_{\mathbf{x_2}:X_3=x_3} f_2(\mathbf{x_2})(X_2 \to f_2) = \sum_{\mathbf{x}:X_3=x_3} f_1(\mathbf{x_1}) f_2(\mathbf{x_2}) f_3(x_2)$,

$f_2 \to X_4 = \sum_{\mathbf{x_2}:X_4=x_4} f_2(\mathbf{x_2})(X_2 \to f_2) = \sum_{\mathbf{x}:X_4=x_4} f_1(\mathbf{x_1}) f_2(\mathbf{x_2}) f_3(x_2)$,

$f_1 \to X_1 = \sum_{\mathbf{x_1}:X_1=x_1} f_1(x_1, x_2)(X_2 \to f_1) = \sum_{\mathbf{x}:X_1=x_1} f_1(\mathbf{x_1}) f_2(\mathbf{x_2}) f_3(x_2)$.

The marginal of X_1 is $b(x_1) = f_1 \to X_1$.

The marginal of X_2 is $b(x_2) = (f_1 \to X_2)(f_2 \to X_2)(f_3 \to X_2)$.

The marginal of X_3 is $b(x_3) = f_2 \to X_3$ and the marginal of X_4 is $b(x_4) = f_2 \to X_4$.

□

One of the notable features present in this computation involves the distribution of sums and products, alluding to the subsequent minor lemma.

Lemma 4.2.1 *(distributivity) For finite sets $S_1, \ldots, S_k \subset \mathbb{N}$,*

$$\prod_{i=1}^{k} \sum_{v_i \in S_i} v_i = \sum_{(v_1,\ldots,v_k)\in(S_1,\ldots,S_k)} \prod_{i=1}^{k} v_i.$$

Proof Induction. Base $k = 2$ (i.e. $x_1 + \cdots + x_k)(y_1 + \cdots + y_l) = x_1 y_1 + \ldots x_k y_l$):

$$\left(\sum_{v_1 \in S_1} v_1\right)\left(\sum_{v_2 \in S_2} v_2\right) = \sum_{(v_1,v_2)\in(S_1,S_2)} v_1 v_2.$$

Suppose the lemma holds for $k-1$. Then

$$\begin{aligned} \textstyle\prod_{i=1}^{k} \sum_{v_i \in S_i} v_i &= (\textstyle\prod_{i=1}^{k-1} \sum_{v_i \in S_i} v_i) \sum_{v_k \in S_k} v_k \\ &= (\textstyle\sum_{(v_1,\ldots,v_{k-1}\in(S_1,\ldots,S_{k-1})} \prod_{i=1}^{k-1} v_i) \sum_{v_k \in S_k} v_k, \text{ by induction,} \\ &= \textstyle\sum_{((v_1,\ldots,v_{k-1}),v_k)\in((S_1,\ldots,S_{k-1}),S_k)} \prod_{i=1}^{k} v_i, \text{ by base case.} \end{aligned}$$

■

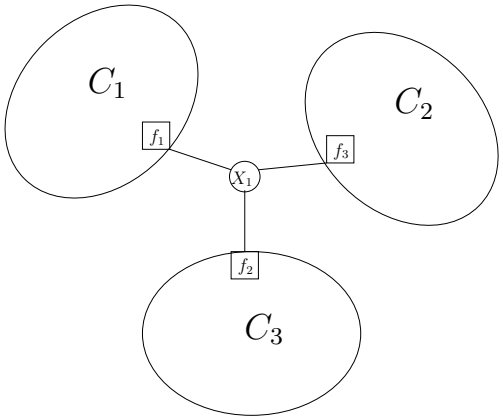

Fig. 4.2: Factor tree example with components C_1, C_2, C_3 with X_1 as a non-leaf

This simple lemma helps prove the following theorem, which states that belief propagation solves single variable marginalisation for factor trees.

Theorem 4.2.2 *For any factor tree G, the marginal of X_n is $z(x_n) = b(x_n)$. That is, $z(x_n) = \prod_{f_m \in N(X_n)} f_m \to X_n$ upon completion of message-passing.*

Proof Induction on $|F|$. Base case $|F| = 1$. All variables are leaves neighbouring f_1, so for any leaf X_i, $X_i \to f_1 = 1$. Then for any leaf X_n,

$$f_1 \to X_n = \sum_{\mathbf{x_1}: X_n = x_n} f_1(\mathbf{x_1}) = \sum_{\mathbf{x}: X_n = x_n} g(\mathbf{x}) = z(x_n).$$

This also holds when the factor is a leaf. Suppose the theorem holds for $< |F|$ factors. For simplicity, consider X_1.

Case 1: X_1 is not a leaf variable.

Let C_i be the component of $G \setminus \{X_1\}$ containing $f_i \in N(X_1)$, and let C_i' be C_i with the vertex X_1 and edge $X_1 f_i$. The vertices $V(C_i) = X(C_i) \cup F(C_i)$, which respectively, are the variables and factors in C_i. Finally, let $\mathbf{c_i}, \mathbf{c_i'}$ be an assignment to all variables in $X(C_i), X(C_i')$ respectively.

Fact All $V(C_i)$s are disjoint and cover all vertices in G except X_1 because G is a tree, i.e. $V(G) \setminus \{X_1\} = \cup_{f_i \in N(X_1)} V(C_i)$. Figure 4.2 depicts an example of these components.

So consider such a message $f_i \to X_1$ in C_i'. By induction, $(f_i \to X_1)_{C_i'} = z_{C_i'}(x_1)$. Note that $f_i \to X_1$ in G depends solely on messages sent in C_i' (G is a tree) and factors in C_i' retain the same domain as their counterparts in G. Thus

$$f_i \to X_1 = (f_i \to X_1)_{C_i'} = z_{C_i'}(x_1) = \sum_{\mathbf{c_i'}:\mathbf{X_1}=\mathbf{x_1}} \prod_{f_j \in F(C_i)} f_j(\mathbf{x_j}).$$

So from this,

$$\begin{aligned}
\textstyle\prod_{f_i \in N(X_1)} f_i \to X_1 &= \textstyle\prod_{f_i \in N(X_1)} \sum_{\mathbf{c_i'}:X_1=x_1} \prod_{f_j \in F(C_i)} f_j(\mathbf{x_j}) \\
&= \textstyle\sum_{\{\mathbf{c_i}'\}_{f_i \in N(X_1)}:X_1=x_1} \left(\prod_{f_i \in N(X_1)} \prod_{f_j \in F(C_i)} f_j(\mathbf{x_j}) \right) \\
&= \textstyle\sum_{\mathbf{x}:X_1=x_1} \prod_{f_j \in F} f_j(\mathbf{x_j}) \\
&= z(x_1).
\end{aligned}$$

The second equality holds by Lemma 4.2.1, and the Fact validates the third equality.

Case 2: X_1 is a leaf variable. For simplicity, let f_1 be the neighbour of X_1 and let $N(f_1) = \{X_1, X_2, \ldots, X_j\}$.

Let D_i be the component of $G \setminus \{f_1\}$ containing $X_i \in N(f_1)$, and the vertices $V(D_i) = X(D_i) \cup F(D_i)$, which respectively, are the variables and factors in D_i. Finally, let $\mathbf{d_i}$ be an assignment to all variables in $X(D_i)$.

Fact All $V(D_i)$s are disjoint and cover all vertices in G except f_1 because G is a tree, i.e. $V(G) \setminus \{f_1\} = \cup_{X_i \in N(f_1)} V(D_i)$. Figure 4.3 depicts an example of these components.

Consider the marginal of $X_i \in N(f_1)$ in D_i. Thus

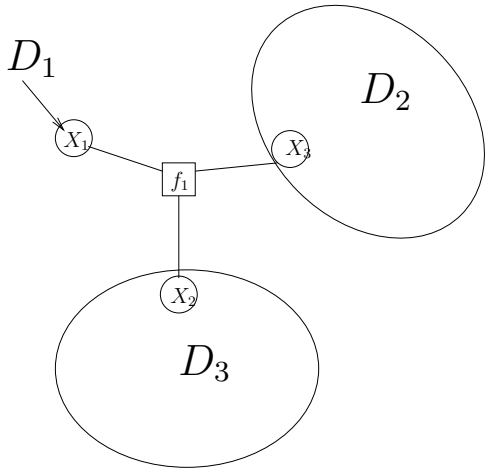

Fig. 4.3: Factor tree example with components D_1, D_2, D_3 with X_1 as a leaf

$$\begin{aligned} X_i \to f_1 &= \prod_{f_m \in N(X_i)-\{f_1\}} f_m \to X_i \\ &= \prod_{f_m \in N(X_i)-\{f_1\}} (f_m \to X_i)_{D_i} = z_{D_i}(x_i) \\ &= \sum_{\mathbf{d_i}: X_i = x_i} \prod_{f_j \in F(D_i)} f_j(\mathbf{x_j}). \end{aligned}$$

The second equality holds because the messages forming the first product in G depend only on messages in D_i, so these messages are the same in D_i. Induction justifies the third equality. Then since X_1 is a leaf, it suffices to show $f_1 \to X_1 = z(x_1)$:

$$\begin{aligned} f_1 \to X_1 &= \sum_{\mathbf{x_1}: X_1 = x_1} f_1(\mathbf{x_1}) \prod_{X_i \in N(f_1)\setminus\{X_1\}} X_i \to f_1 \\ &= \sum_{\mathbf{x_1}: X_1 = x_1} f_1(\mathbf{x_1}) \prod_{X_i \in N(f_1)\setminus\{X_1\}} \sum_{\mathbf{d_i}: X_i = x_i} \prod_{f_j \in F(D_i)} f_j(\mathbf{x_j}) \\ &= \sum_{\mathbf{x_1}: X_1 = x_1} f_1(\mathbf{x_1}) (\sum_{\{\mathbf{d_i}: X_i = x_i, X_i \in N(f_1)\setminus\{X_1\}\}} \prod_{X_i \in N(f_1)} \prod_{f_j \in F(D_i)} f_j(\mathbf{x_j})) \\ &= \sum_{\mathbf{x_1}: X_1 = x_1} f_1(\mathbf{x_1}) (\sum_{\{\mathbf{d_i}: X_i = x_i, X_i \in N(f_1)\setminus\{X_1\}\}} \prod_{f_j \in F - \{f_1\}} f_j(\mathbf{x_j})) \\ &= \sum_{\mathbf{x}: X_1 = x_1} \prod_{f_j \in F} f_j(\mathbf{x_j})) \\ &= z(x_1). \end{aligned}$$

(4.2) upholds the first equality, the second equality is a consequence of the preceding discussion, the third equality is true because of Lemma 4.2.1, and the Fact establishes the fourth equality. Thus, the proof is complete at last. ■

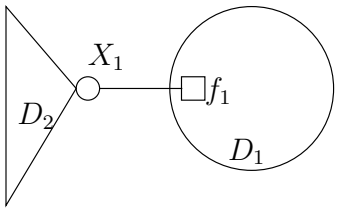

Fig. 4.4: Factor tree with components D_1, D_2 and $C_1 = D_1 \cup \{X_1\}, C_2 = D_2 \cup \{X_2\}$.

Additionally, belief propagation can also compute the marginal function of a factor f_m.

Corollary 4.2.3 *For any factor tree G and any factor $f_m \in F$, $z(\mathbf{x_m}) = b(\mathbf{x_m})$, that is $z(\mathbf{x_m}) = f_m(\mathbf{x_m}) \prod_{X_j \in N(f_m)} X_j \to f_m$.*

Proof Consider f_1 and suppose that its neighbours are $X_1, \ldots, X_d$. If f_1 has only one neighbour X_1 and $H = G \setminus \{f_1\}$, then

$$\begin{aligned} b(x_1) = f_1(x_1)(X_1 \to f_1) &= f_1(x_1) \textstyle\prod_{f_m \in N(X_1) \setminus \{f_1\}} f_m \to X_1 \\ \text{By above Theorem:} &= f_1(x_1) z_H(x_1) \\ &= f_1(x_1) \textstyle\sum_{\mathbf{x}: X_1 = x_1} \prod_{f_j \in F(H)} f_j(\mathbf{x_j}) \\ &= \textstyle\sum_{\mathbf{x}: X_1 = x_1} \prod_{f_j \in F} f_j(\mathbf{x_j}) = z(x_1). \end{aligned}$$

Otherwise, apply induction on $|F|$. For the base case $|F| = 1$, $X_i \to f_1 = 1$ and $\mathbf{x_1} = \mathbf{x}$ so the marginal of f_1 is the marginal of all variables, i.e. the global function: $b(\mathbf{x}) = f_1(\mathbf{x}) = g(\mathbf{x})$.

Now consider the general case. Since G is a tree, removing X_1 will leave two disjoint subgraphs: D_1, which contains f_1 and its neighbours, and D_2, which contains the rest of the neighbours of X_1. Now let $C_1 = D_1 \cup \{X_1\}$ and $C_2 = D_2 \cup \{X_1\}$ and let $\mathbf{c_1}, \mathbf{c_2}$ denote the assignments to variables in these two subgraphs (Figure 4.4). Then

$$\begin{aligned}
b(\mathbf{x_1}) &= f_1(\mathbf{x_1}) \textstyle\prod_{X_j \in N(f_1)} X_j \to f_1 \\
&= (f_1(\mathbf{x_1}) \textstyle\prod_{X_j \in N(f_1)\setminus\{X_1\}} X_j \to f_1) X_1 \to f_1 \\
&= (\textstyle\sum_{\mathbf{c_1}:N(f_1)=\mathbf{x_1}} \prod_{f_i \in F(C_1)} f_i(\mathbf{x_i}))(\prod_{f_j \in N(X_1)\setminus\{f_1\}} f_j \to X_1) \\
&= (\textstyle\sum_{\mathbf{c_1}:N(f_1)=\mathbf{x_1}} \prod_{f_i \in F(C_1)} f_i(\mathbf{x_i})) z_{C_2}(x_1) \\
&= (\textstyle\sum_{\mathbf{c_1}:N(f_1)=\mathbf{x_1}} \prod_{f_i \in F(C_1)} f_i(\mathbf{x_i}))(\sum_{\mathbf{c_2}:X_1=x_1} \prod_{f_i \in F(C_2)} f_i(\mathbf{x_i})) \\
&= \textstyle\sum_{\mathbf{x}:N(f_1)=\mathbf{x_1}} \prod_{f_i \in F(C_1)} f_i(\mathbf{x_i})) \prod_{f_i \in F(C_2)} f_i(\mathbf{x_i})) \\
&= \textstyle\sum_{\mathbf{x}:N(f_1)=\mathbf{x_1}} \prod_{f_i \in F} f_i(\mathbf{x_i})) = z(\mathbf{x_1}).
\end{aligned}$$

Induction gives the second equality, and Theorem 4.2.2 accounts for the third equality. The sixth equality holds because $V(C_1) \cap V(C_2) = \{X_1\}$ and $V(C_1) \cup V(C_2) = \boldsymbol{X}$. Note that the initial assignment $\mathbf{x_1}$ for b already includes the assignment for X_1, so both subgraphs only fix the same assignment to the lone shared variable X_1. Also, fixing $N(f_1)$ for C_2 is valid because no neighbour of f_1 besides X_1 is in C_2. Since this covers all variables in G, the combined summation is over assignments of $\boldsymbol{X}$.

The second-last equality holds because $F(C_1) \cap F(C_2) = \emptyset$ and $F(C_1) \cup F(C_2) = F$. Thus combining the two products has no overlap and covers the whole set of factors. ■

Lastly, one must verify that message-passing will complete across all edges.

Proposition 4.2.4 *For any tree G, at any step during belief propagation, there is always a node in $S \subseteq V$ that can send a message, where $S = \{s_1, \ldots, s_k\}$ is the set of all nodes with messages to send at this step.*

Proof Suppose that at some point during the execution of the algorithm, no $s_i \in S$ can send a message. Then every $s_i \in S$ is waiting for messages from at least one other node. Note that all unsent messages must originate from nodes in S by definition.

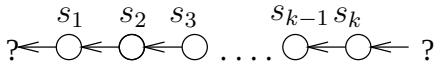

Fig. 4.5: Nodes that cannot send messages

Without loss of generality, assume that s_1 awaits a message from s_2, which awaits a message from s_3 and so forth, as depicted in Figure 4.5. Then all nodes in S except s_1 and s_k have at least one message to send and cannot do so. So s_k must await a message from some node in S, but then there is a cycle in S achieving a contradiction. ■

4.2.2 Loopy belief propagation

The above algorithm does not quite work in general graphs, since there may be no leaves. A simple modification of the above algorithm will allow it to run on general graphs. The given algorithm converges when a message is sufficiently close to its previous message, for sufficiently many consecutive iterations.

The modification is to initialise all messages $X_n \to f_m = 1$. This converges to a set of messages consistent with the message definitions (4.1) and (4.2) in the tree case.

This method achieves convergence to the messages in the previous algorithm in $diam(G)$ iterations for any factor tree G. For instance, if $diam(G) = 1$, there is one factor and one variable, so the messages transmitted in the first iteration are already the desired messages. If $diam(G) = 2$, say there are two factors and one variable. Since the factors are leaves, they transmit the correct message in the first iteration, but the variable requires a second iteration to transmit the correct message.

Proposition 4.2.5 *For any factor tree G and any message from u to v, belief propagation converges in i iterations, where i is the maximum distance to a leaf without*

`BeliefProp`

Output: set of messages

begin

Initialise all variable messages $X_n \to f_m = 1.$;

while *there is an unconverged factor message* **do**

for *each* $X_n \in X$ **do**

transmit $X_n \to f_m$ for all $f_m \in N(X_n)$.

end

for *each* $f_m \in F$ **do**

Compute $f_m \to X_n$ according to (4.2) for all $X_n \in N(f_m)$.

if $|f_m \to X_n^{new} - f_m \to X_n^{old}| < \epsilon$ **then**

$f_m \to X_n$ has potentially converged, so increment a counter ctr.

if $ctr = k$ **then**

Message $f_m \to X_n$ has converged.

end

end

else

$f_m \to X_n$ has not converged, so reset $ctr = 0$

end

end

Compute all new messages $X_n \to f_m$ according to (4.1) for the next iteration.

end

end

going through v.

Proof Induction on i. $i = 0$ (u is a leaf) is trivial.

Suppose the proposition holds for $\leq i$. Consider a message from u to v where the max distance to a leaf without going through v is $i+1$. Now consider a message from $w \in N(u) \setminus \{v\}$ to u. The max distance to a leaf without going through u for any such message is at most i. So by induction, each of the messages affecting the computation of $u \to v$ converges within i iterations, so the message from u to v computed in the iteration $i+1$ must converge appropriately. e.g. If u is a variable x_n and v is f_m, then by the end of iteration i, then for any $m_0 \in N(x_n) \setminus \{m\}$, $f_{m_0} \to X_n$ is exactly as defined in the original algorithm, so in iteration $i+1$, the computation of $X_n \to f_m$ must also correspond exactly to the required message. Moreover, it is apparent that once a message becomes the "correct" message, it will always remain correct because G is a tree. ■

Then the convergence result follows easily:

Corollary 4.2.6 *For any factor tree G, all messages converge within $diam(G)$ iterations.*

There is no such guarantee of convergence in the general case, but as mentioned before, empirical and analytical work suggests that belief propagation does work well in some specialised settings. Much of the literature explaining the success of loopy belief propagation in marginalisation is empirical and non-rigorous [HM97, MWJ99], although recent work gives some interesting sufficient conditions for the convergence of loopy belief propagation [IIW05, MK05, TJ01]. Consequently, the question of the quality of the belief propagation approximation in marginalisation is still open. Since approximate inference is NP-hard, the best one can hope for is an understanding of the types of graphs on which belief propagations works best.

4.3 Belief propagation in other graphical models

4.3.1 Belief propagation in Bayesian networks

In his detailed book [Pea88], Judea Pearl introduced the notion of belief propagation in Bayesian networks for the particular inference problem of marginalisation. Its description however, is less compact than the same algorithm in factor graphs.

Recall that in Bayesian networks, $\mathbf{x_i}$ specifies values of X_i and $Pa(X_i)$, and factor $f_i(\mathbf{x_i}) \equiv P(x_i|\mathbf{pa_i})$, where $\mathbf{pa_i}$ are values for all variables in $Pa(X_i)$.

Definition For every $X_d \in Pa(X_s)$, the *"backward" message* (from child to parent) is

$$\lambda_{X_s}(x_d) = \sum_{\mathbf{x_s}:X_d=x_d} \left(f_s(\mathbf{x_s}) \prod_{X_i \in Ch(X_s)} \lambda_{X_i}(x_s) \prod_{X_j \in Pa(X_s)\setminus\{X_d\}} \pi_{X_s}(x_j) \right)$$

Definition For every $X_d \in Ch(X_s)$, the *"forward" message* (from parent to child) is

$$\pi_{X_d}(x_s) = \prod_{X_i \in Ch(X_s)\setminus\{X_d\}} \lambda_{X_i}(x_s) \left(\sum_{\mathbf{x_s}:X_s=x_s} f_s(\mathbf{x_s}) \prod_{X_j \in Pa(X_s)} \pi_{X_s}(x_j) \right)$$

Definition The *belief* of variable X_s computed by this version of belief propagation is

$$BEL(x_s) = \prod_{X_i \in Ch(X_s)} \lambda_{X_i}(x_s) \left(\sum_{\mathbf{x_s}:X_s=x_s} f_s(\mathbf{x_s}) \prod_{X_j \in Pa(X_s)} \pi_{X_s}(x_j) \right)$$

In the conversion, create one factor node $f_i(\mathbf{x_i})$ for every variable X_i (recall Figure 3.5 in Section 3.4). One can derive the factor graph version of belief propagation from Pearl's algorithm. This can be done by simply dissecting the message definitions for Bayesian networks.

Essentially, $\pi_{X_d}(x_s) = X_s \to f_d$ and $\lambda_{X_s}(x_d) = f_s \to X_d$. First consider $\pi_{X_d}(x_s)$. When converting the Bayesian network into a factor graph, the new link from X_s to

X_d is from X_s to f_d, so it would be desirable for the forward message to correspond precisely to $X_s \to f_d$. Similarly, since the backward message $\lambda_{X_s}(x_d)$ needs to transmit information from X_s to X_d, it would be ideal for the backward message to correspond to the message from the factor of X_s to X_d.

$$\begin{aligned}
\pi_{X_d}(x_s) &= \prod_{X_i \in Ch(X_s)\setminus\{X_d\}} \lambda_{X_i}(x_s) \left(\sum_{\mathbf{x_s}: X_s = x_s} f_s(\mathbf{x_s}) \prod_{X_j \in Pa(X_s)} \pi_{X_s}(x_j) \right) \\
&= \prod_{X_i \in Ch(X_s)\setminus\{X_d\}} (f_i \to X_s) \left(\sum_{\mathbf{x_s}: X_s = x_s} f_s(\mathbf{x_s}) \prod_{X_j \in Pa(X_s)} (X_j \to f_s) \right) \\
&= \prod_{X_i \in Ch(X_s)\setminus\{X_d\}} (f_i \to X_s)(f_s \to X_s) \\
&= \prod_{f_i \in N(X_s)} (f_i \to X_s).
\end{aligned}$$

To explain the above, the product of $\lambda_{X_i}(x_s)$ for $X_i \in Ch(X_s)$ is the product of messages from children of X_s. When inserting factor nodes, these are the messages from $f_i \to X_s$. This accounts for all messages that $X_s \to f_d$ needs except $f_s \to X_s$. Thus set $f_s \to X_s$ to be the remaining summation over values $\mathbf{x_s}$. Note that this message accounts for all messages from the parents of X_s, which is are all of the neighbours of f_s except X_s. Consequently, this accounts for all the necessary messages to compute $X_s \to f_d$.

$$\begin{aligned}
\lambda_{X_s}(x_d) &= \sum_{\mathbf{x_s}: X_d = x_d} \left(f_s(\mathbf{x_s}) \prod_{X_i \in Ch(X_s)} \lambda_{X_i}(x_s) \prod_{X_j \in Pa(X_s)\setminus\{X_d\}} \pi_{X_s}(x_j) \right) \\
&= \sum_{\mathbf{x_s}: X_d = x_d} \left(f_s(\mathbf{x_s}) \prod_{X_i \in Ch(X_s)} (f_i \to X_s) \prod_{X_j \in Pa(X_s)\setminus\{X_d\}} (X_j \to f_s) \right) \\
&= \sum_{\mathbf{x_s}: X_d = x_d} \left(f_s(\mathbf{x_s})(X_s \to f_s) \prod_{X_j \in Pa(X_s)\setminus\{X_d\}} (X_j \to f_s) \right) \\
&= \sum_{\mathbf{x_s}: X_d = x_d} f_s(\mathbf{x_s}) \prod_{X_j \in N(f_s)} (X_j \to f_s).
\end{aligned}$$

The product of $\lambda_{X_i}(x_s)$ composes of messages from the children of X_s, that is messages $f_i \to X_s$. This product yields precisely the message $X_s \to f_s$ because the set of neighbours is $N(X_s) = \{f_s\} \cup \{f_i : X_i \in Ch(X_s)\}$. Finally, the product of $\pi_{X_s}(x_i)$ composes of messages from the parents of X_i, that is $X_i \to f_s$. Since the set of parents excludes X_d, this message is precisely $f_s \to X_d$. Hence the message definitions from the factor graph model are consistent with those in the Bayesian network model.

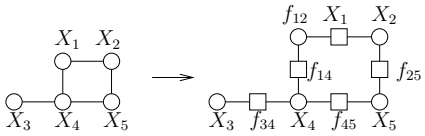

Fig. 4.6: Converting a pairwise Markov random field to a factor graph

Then it is easy to check that $BEL(x_s) = B(x_s)$.

$$\begin{aligned} BEL(x_s) &= \prod_{X_i \in Ch(X_s)} \lambda_{X_i}(x_s) \left(\sum_{\mathbf{x_s}: X_s = x_s} f_s(\mathbf{x_s}) \prod_{X_j \in Pa(X_s)} \pi_{X_s}(x_j) \right) \\ &= \prod_{X_i \in Ch(X_s)} (f_i \to X_s) \, (f_s \to X_s) \\ &= \prod_{f_i \in N(X_s)} f_i \to X_s = B(x_s). \end{aligned}$$

Since we can always convert Bayesian networks into factor graphs, and the same algorithm applies for both models, it is reasonable to consider the simpler message definitions given for the factor graph model.

4.3.2 Belief propagation in Markov random fields

Most literature describes belief propagation for Markov random fields under a specialised type of Markov random fields called pairwise Markov random fields [Wei01, YFW01]. These are simply Markov random fields with a maximal clique size of 2, i.e. there is no subclique with 3 nodes. Since the edges are undirected and all nodes are variables, this model only has one message:

$$X \xrightarrow{M} Y(y) = \sum_x f_{XY}(x, y) \prod_{Z \in N(X) \setminus \{Y\}} Z \xrightarrow{M} X(x)$$

The computation of the belief is $b(x) = \prod_{Z \in N(X)} Z \xrightarrow{M} X(x)$, the same as in the factor graph case.

As in the Bayesian network case, it is easy to transform this problem into the factor graph case and vice versa when applicable. Figure 4.6 depicts a conversion from a

pairwise Markov random field to a factor graph. This case is more straightforward: Set

$$X \xrightarrow{M} Y = f_{XY} \to Y,$$

and then from the above message definition, Consequently,

$$\prod_{Z \in N(X) \setminus \{Y\}} Z \xrightarrow{M} X(x) = X \to f_{XY}.$$

Thus the message definitions of factor graphs are not nearly as simple as that of pairwise Markov random fields. On the other hand, the difficulty with this model is that pairwise Markov random fields are quite restrictive without first clustering cliques in general Markov random fields. Also, factor graphs are certainly more general than pairwise Markov random fields.

4.4 Related inference problems

An inference problem related to marginalisation called the maximisation or most probable explanation (MPE) problem has a solution similar to belief propagation known as the min-sum, Viterbi, or max-product algorithm [Mac03, p. 339]. This algorithm is also the inspiration for the warning propagation algorithm [BMZ03] which will be discussed in Section 6.2. The algorithm is precisely the same as belief propagation except one replaces all instances of $\sum$ with max. This yields the max-product algorithm and all the results from before apply. For instance, it is clear that the distributivity lemma 4.2.1 would also apply when stated as such:

$$\prod_{i=1}^{k} \max_{v_i \in S_i} v_i = \max_{(v_1,\ldots,v_k) \in (S_1,\ldots,S_k)} \prod_{i=1}^{k} v_i.$$

In the base case, $(\max_{v_1 \in S_1} v_1)(\max_{v_2 \in S_2} v_2) = \max_{(v_1,v_2) \in (S_1,S_2)} v_1 v_2$ is clearly true. Similarly, the min-sum algorithm is merely the max-product algorithm in the negative

log-likelihood domain (or replace max with min, and $\prod$ with $\sum$) [Mac03, pp. 339-340]. Unlike marginalisation, which requires the precise value of $g(\mathbf{x})$, maximisation requires only the values $\mathbf{x}$ such that $\max_{\mathbf{x}} g(\mathbf{x})$. So maximisation seems as though it may be easier than marginalisation, and some work suggests that belief propagation works especially well for maximisation [Wei01, WF01]. Still, solving maximisation is NP-complete [KF02, Par02].

To prove NP-completeness of maximisation, we use the following formulations of the problem [PD04]. Note that it is straightforward to show that BN-MPE is NP-hard if BN-MPE-D is NP-complete.

BN-MPE-D: Given a Bayesian network G over variables $\boldsymbol{X}$, evidence $\mathbf{e}$ for a subset of variables $E \subset \boldsymbol{X}$, and a number k, does there exist an assignment $\mathbf{w}$ to $W = \boldsymbol{X} \setminus E$ such that $P(\mathbf{w}, \mathbf{e}) > k$?

BN-MPE: Given a Bayesian network G over variables $\boldsymbol{X}$ and evidence $\mathbf{e}$ for a subset of variables $E \subset \boldsymbol{X}$, solve $\arg\max_{\mathbf{x}} P(\mathbf{w}, \mathbf{e})$, where $W = \boldsymbol{X} \setminus E$.

Proposition 4.4.1 *BN-MPE-D is NP-complete.*

Proof Firstly, the given evidence $\mathbf{e}$ and guess $\mathbf{w}$, yields a full assignment to the joint distribution, so it is easy to check if $P(\mathbf{w}, \mathbf{e}) > k$. To show it is NP-hard, reduce 3SAT to BN-MPE-D. This proof mirrors Cooper's NP-hardness proof for marginalisation [Coo90].

Consider the 3CNF formula F with n variables $Q = \{Q_1, Q_2, \ldots, Q_n\}$ and m clauses, and let a Boolean assignment of the variables be denoted by $\mathbf{q} = \{q_1, q_2, \ldots, q_n\}$. So show that F is satisfiable iff there exists a number k and an instantiation $\mathbf{w}$ to the variables of some Bayesian network G such that $P(\mathbf{w}, \mathbf{e}) > k$, for some instantiation $\mathbf{e}$ of some evidence E. The ensuing proof uses the standard Bayesian network simulation of a formula first presented in Cooper's hardness proof for marginalisation [Coo90].

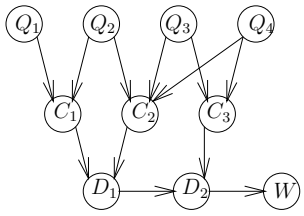

Fig. 4.7: Bayesian network construction of $F = (Q_1 \vee \neg Q_2) \wedge (Q_2 \vee Q_3 \vee \neg Q_4) \wedge (\neg Q_3 \vee Q_4)$. The vertex W is used for the modified MPE problem without evidence described after this proof.

The set of vertices is as follows: one vertex for each variable Q_i, one vertex for each clause C_j, and one vertex for each AND connective D_k and also denote $D_{m-1} = Y$. For edges, $(Q_i, C_j) \in E(G)$ iff clause C_j contains Q_i. To link the clauses, initially use $(C_1, D_1) \in E(G), (C_2, D_1) \in E(G)$, which will represent $D_1 = C_1 \wedge C_2$, and subsequently use $(D_{j-1}, D_j) \in E(G), (C_{j+1}, D_j) \in E(G)$, $j \geq 2$, which will represent $D_j = D_{j-1} \wedge C_{j+1}$. Clearly, $Y = D_{m-1}$ will serve to represent the entire formula F. For a simple example, see Figure 4.7.

Let each variable Q_i have a uniform prior probability (i.e. $P(Q_i = 1) = 0.5$). The rest of the vertices are essentially indicator variables set based on which connective they represent in the formula F, namely every C_j is an OR connective and every D_k is an AND connective. Thus $P(C_j = 1 | Q_{j1} = q_{j1}, Q_{j2} = q_{j2}, Q_{j3} = q_{j3}) = 1$ if the given assignment of variables satisfies C_j, and 0 otherwise. Similarly, if $c_k = d_{k-1} = 1$, then $P(D_k = 1 | C_k = c_k, D_{k-1} = d_{k-1}) = 1$ and 0 otherwise (note the slightly different case for D_1). This construction is obviously attainable in time which is polynomial in the size of the formula. The network has size $O(|F|)$ and the conditional probability tables are all small: the largest set of probabilities required are for the clauses which are given at most three variables.

Note that the joint probability $P(\mathbf{x})$ can be condensed because of the nature of

the vertices in G that represent connectives. Given some assignment $\mathbf{q}$, the above construction already determines the remainder of the conditional probabilities, and there is exactly one assignment of values to the C_j and D_k vertices with positive probability. Thus clearly, $Y = 1$ iff an assignment $\mathbf{q}$ satisfies F, so Y to represent all of the intermediary connectives that essentially form F.

So $P(Q = \mathbf{q}, Y = 1) = 1/2^n$ if $\mathbf{q}$ satisfies F and 0 otherwise. Now set the evidence $E = \{Y\}$ with $\mathbf{e} = 1$, that is let the evidence be $Y = 1$, and take $k = 1/2^{n+1}$. Thus $P(\mathbf{q}, \mathbf{e}) = 1/2^n > k$ if $\mathbf{q}$ satisfies F and $P(\mathbf{q}, \mathbf{e}) = 0 \not> k$ if $\mathbf{q}$ doesn't satisfy F. Hence: F is satisfiable iff for some Bayesian network G, evidence E, and number k, there exists an instantiation $\mathbf{w}$ to variables $W = \boldsymbol{X} \setminus E$ such that $P(\mathbf{w}, \mathbf{e}) > k$. ■

Solving the complexity of same decision problem with no evidence requires a slight modification to the construction in a reduction from 3SAT. The problem is now as follows:

Given a Bayesian network G over variables $\boldsymbol{X}$, and a number k, does there exist an assignment $\mathbf{x}$ to $\boldsymbol{X}$ such that $P(\mathbf{x}) > k$?

Shimony shows a reduction from vertex cover [Shi94] and Park and Darwiche demonstrate a reduction from a satisfiability problem to prove a similar inference problem without evidence [PD04]. However, only one vertex needs to be added to the above construction to achieve the desired result:

Use the same construction as before with the additional vertex W and edge (D_{m-1}, W) (Figure 4.7), and set its conditional probability

$$P(W = 1 | D_{m-1} = d_{m-1}) = \begin{cases} 1 & \text{if } d_{m-1} = 1; \\ 1/2 & \text{otherwise.} \end{cases}$$

Then it is clear that there is an assignment $\mathbf{x}$ such that $P(\mathbf{x}) > 1/2^{n+1}$ iff F is satisfiable.

Another inference problem that is probably more difficult than marginalisation and maximisation is the generalisation of maximisation called the maximum a posteriori (MAP) problem. The goal of MAP is to determine values $\mathbf{s}$ of variables in some subset $S \subseteq \boldsymbol{X}$ satisfying $\max_{\mathbf{s}} P(\mathbf{s})$. MAP is difficult because it combines marginalisation ($P(\mathbf{s}) = \sum_{\{\mathbf{x}:S=\mathbf{s}\}} P(\mathbf{x})$) and maximisation, and it is $\mathrm{NP}^{\mathrm{PP}}$-complete [Par02, PD04], a class believed to be substantially harder than NP. Coupled with the hardness of approximating marginalisation, it seems that inference in general is a difficult problem, yet there is still hope for approximation algorithms on particular classes of graphs. Some work has already begun in the analyses of such algorithms, and chiefly for belief propagation.

Chapter 5

Some insight and analysis into belief propagation

Several threads of research demonstrate the theoretical and empirical value of belief propagation. For instance, Weiss shows the convergence and an error bound of belief propagation in a single cycle graph [AHM98, Wei01], Yedidia et al. and Heskes unveiled the relationship between belief propagation and free energy [Hes04, YFW02], and Richardson and Urbanke prove the convergence and concentration of the algorithm in LDPC codes [RU01]. Finally, active research on the convergence of loopy belief propagation has aided in understanding the algorithm through various mathematical techniques [IIW05, MK05, TJ01]. This section explore these avenues of free energy and LDPC codes, and implements a rudimentary belief propagation algorithm to explore some simple classes of cyclic factor graphs.

5.1 Free energy

Discussing free energy necessitates an apparent diversion to statistical mechanics. However, maintaining notation consistent with previous sections will help to illustrate the intersection between free energy approximation and belief propagation.

The set of values $\mathbf{x} \in \mathbf{X}$ is sometimes called the set of realisations for the n discrete-valued random variables in $\boldsymbol{X}$, or the state of the system in statistical mechanics. Recall the factorised joint distribution $P(\mathbf{x}) = \frac{1}{Z} \prod_{j=1}^{m} f_j(\mathbf{x_j})$.

5.1.1 Definitions and motivation

One of the centerpieces of statistical mechanics is Boltzmann's Law, which is based on the belief that physical systems tend to be in states of low energy. For a physical explanation, see http://hyperphysics.phy-astr.gsu.edu/hbase/quantum/disfcn.html

Definition For a given temperature T and energy function $E : \mathbf{X} \to \mathbb{R}$, the *Boltzmann distribution* is the following equation:

$$P(\mathbf{x}) = e^{-E(\mathbf{x})/T}/Z, \tag{5.1}$$

where the normalisation constant or *partition function* is $Z = \sum_{\mathbf{x} \in \mathbf{X}} e^{-E(\mathbf{x})/T}$.

This distribution is a generalisation of the Gibbs and factorised distribution mentioned earlier. In fact, throughout this section, assume that the system is set up as a factor graph with the factorised distribution and the constant $T = 1$. Thus in this situation, the energy is simply a function of the factors of G:

$$E(\mathbf{x}) = -\sum_{f_j \in F} \ln f_j(\mathbf{x}_j). \tag{5.2}$$

The major obstacle in the computation of the Boltzmann distribution lies in the partition function Z, because the naïve method requires a summation over all possible

values of $\mathbf{X}$. Traditionally, the *Helmholtz free energy* F_H of a system is the desired quantity because computing it will also yield Z:

$$F_H = -\ln Z.$$

One method of approximation is the variational approach, which approximates the joint distribution $P(\mathbf{x})$ with a simpler distribution $B(\mathbf{x})$. This method relies on the variational free energy defined in terms of B and its close connection to the Helmholtz free energy.

Definition The *Gibbs free energy* or *variational energy* $F(B)$ is a function of $B(\mathbf{x})$ expressed in terms of the *variational average energy* $U(B)$ and the *variational entropy* $H(B)$. They are defined as follows, where the belief $B(\mathbf{x})$ is an approximation of the joint distribution of the system $P(\mathbf{x})$:

$$F(B) = U(B) - H(B) = \sum_{\mathbf{x}\in\mathbf{X}} B(\mathbf{x})E(\mathbf{x}) - \left(-\sum_{\mathbf{x}\in\mathbf{X}} B(\mathbf{x}) \ln B(\mathbf{x}) \right)$$

This apparently nebulous equation is in fact rather straightforward. The average energy is simply defined like the expected value of the energy function in the system. Similarly, the entropy is a measure of the average amount of information in the system. For a thorough discussion regarding its derivation, see works on information theory or thermodynamics [JJ00, Mac03]. In the context of energy, entropy can be construed as the inherent energy present in the system, so the free energy is precisely the difference of total average energy and the intrinsic energy of the system. Another quantity known as the Kullback-Leibler divergence aids in expressing the relationship between the variational energy and the Helmholtz free energy.

Definition The *Kullback-Leibler divergence* or *relative entropy* measures the difference between distribution B and P, and is defined as

$$KL(B, P) = \sum_{\mathbf{x} \in \mathbf{X}} B(\mathbf{x}) \ln \frac{B(\mathbf{x})}{P(\mathbf{x})}.$$

Immediately from these definitions, it follows that $F(B) = F_H + KL(B, P)$. Moreover, $KL(B, P)$ is nonnegative and zero iff $B(\mathbf{x}) = P(\mathbf{x})$. Consequently, $F(B) \geq F_H$ and equality iff $B(\mathbf{x}) = P(\mathbf{x})$. So because $P(\mathbf{x}) = \arg\min_{B(\mathbf{x})} F(B)$, the desired goal is to compute $\min_{B(\mathbf{x})} F(B)$. Unfortunately, minimising over $B(\mathbf{x})$ is also intractable, so the actual goal of this process is to approximate the minimisation of the variational energy in a reasonably efficient manner.

One variational approach, called mean field approximation, attempts to minimise $F(B)$ over a restricted class of probability distributions B. For example, instead of minimising over all beliefs, one can consider beliefs of the form

$$B(\mathbf{x}; \mathbf{a}) = \exp\left(\sum_{\{X_i \in X : X_i = x_i\}} a_i x_i \right),$$

where the minimisation is over $\mathbf{a}$. Mackay outlines how this restriction, one of many viable constructions of $B(\mathbf{x})$, yields a tractable minimisation of the variational free energy [Mac03, pp. 424-425].

5.1.2 Region-based approximations

Yedidia et al. introduced another approximation technique, dubbed region-based approximations, by minimising over the region-based free energy rather than the variational free energy [YFW02]. The crux of the method is to first compute the free energy of smaller, more manageable pieces, then combining them to form the region-based free energy, and finally minimising this combined quantity. Figure 5.1

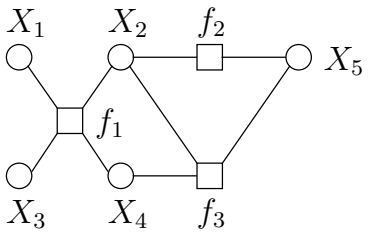

Fig. 5.1: Factor graph. If $F_R = \{f_2, f_3\}$, then $\{X_2, X_4, X_5\} \subseteq X_R$. In a Bethe approximation, the regions are $R_{f_1} = \{f_1, X_1, X_2, X_3, X_4\}, R_{f_2} = \{f_2, X_2, X_5\}, R_{f_3} = \{f_3, X_2, X_4, X_5\}$, and $R_{X_i} = \{X_i\}$ for each $X_i \in \boldsymbol{X}$.

illustrates an example of regions in a factor graph. The definitions for the marginal and energy in a region mirror those for the entire system.

Definition A *region* $R = X_R \cup F_R$ where $X_R \in \boldsymbol{X}$ is a subset of variables and F_R is a subset of factors under the constraint that if $f_j \in F_R$, then all variables neighbouring f_j are in X_R. Moreover, $\mathbf{x_R}$ is an assignment to the variables in R.

Definition The *marginal distribution over region* R is $P(\mathbf{x}_R)$ and belief $b_R(\mathbf{x}_R)$ or $B_R(\mathbf{x}_R)$ (B_R is a distribution, whereas b_R might not be) approximate $P(\mathbf{x}_R)$. The *region energy of* R is

$$E_R(\mathbf{x}_R) = -\sum_{f_j \in F_R} \ln f_j(\mathbf{x}_j).$$

The *region free energy of* R is

$$F_R(b_R) = U_R(b_R) - H_R(b_R) = \sum_{\mathbf{x}_R} b_R(\mathbf{x}_R) E_R(\mathbf{x}_R) - \left(-\sum_{\mathbf{x}_R} b_R(\mathbf{x}_R) \ln b_R(\mathbf{x}_R) \right).$$

The goal is to divide the factor graph into regions so that the overall free energy is the sum of the free energies of all the regions. Let $\mathcal{R}$ be a set of regions and c_R be the counting number of region R. The constants c_R roughly determine how often the approximation counts each variable and factor. The region-based free energy $F_{\mathcal{R}}$ represents this approximation to the overall free energy:

Definition The *region-based free energy* or *approximation* is

$$F_{\mathcal{R}}(\{b_R\}) = U_{\mathcal{R}}(\{b_R\}) - H_{\mathcal{R}}(\{b_R\}) = \sum_{R\in\mathcal{R}} c_R U_R(b_R) - \sum_{R\in\mathcal{R}} c_R H_R(b_R).$$

Definition A *valid region-based approximation* satisfies

$$\sum_{R\in\mathcal{R}} c_R I_{F_R}(f_j) = \sum_{R\in\mathcal{R}} c_R I_{X_R}(X_i) = 1,$$

where $I_S(x) = 1$ if $x \in S$, 0 otherwise, for all variables X_i and factors f_j.

Definition A *constrained region-based free energy* is a region-based free energy with beliefs $B_R(\mathbf{x}_R)$ that are probability distributions, i.e. $F_{\mathcal{R}}(\{B_R\})$. Moreover, the marginal region beliefs $B(\mathbf{x}_S)$ are consistent for pairs of regions that both contain the set of variables $\boldsymbol{X}$. For instance, if $S \subseteq R_1, S \subseteq R_2$, then in both regions, the marginal of variables S is the same in both regions.

A valid region-based approximation has the subsequent properties that gives it credence as an approximation to the variational free energy.

Proposition 5.1.1 *For all regions $R \in \mathcal{R}$, if $b_R(\mathbf{x_R}) = P(\mathbf{x_R})$, then the average energy $U_{\mathcal{R}}(\{b_R\})$ of a valid region-based approximation is exact, namely that $U_{\mathcal{R}}(\{b_R\}) = U(P)$.*

Proof This proposition follows from the definition of a valid region-based approximation and the marginal $P(\mathbf{x_R})$, achieving the equality $U_{\mathcal{R}}(\{b_R\}) = U(P)$. The key observation is that every region satisfies $N(f_j) \subseteq X_R$ and extraneous variables in $X_R \setminus N(f_j)$ will sum out leaving the marginal of each factor f_j in F_R. ■

It is easy to see that such a convenient result does not follow for the region-based entropy. Although each variable will have $\sum_{R:X_i\in R} c_R = 1$, the regions with X_i may all have different variables, thus usually there is no way to consolidate the disparate sums involving the marginals $P(\mathbf{x_R})$.

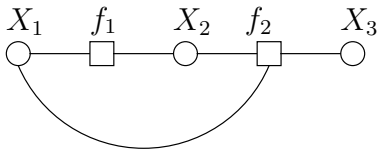

Fig. 5.2: Regions $R_1 = \{f_1, X_1, X_2\}, R_2 = \{f_2, X_1, X_2, X_3\}, R_3 = \{X_1\}, R_4 = \{X_2\}$ of a factor graph.

Example See Figure 5.2, whose graph has regions are given in the figure. Suppose it has counting numbers $c_{R_1} = c_{R_2} = -c_{R_3} = -c_{R_4} = 1$, so that this region-based approximation is valid. Behold that when $b_R(\mathbf{x_R}) = P(\mathbf{x_R})$, the region-based entropy is
$$H_{\mathcal{R}} = -\Big(\sum_{x_1,x_2} P(x_1,x_2) \ln P(x_1,x_2) + \sum_{\mathbf{x}} P(\mathbf{x}) \ln P(\mathbf{x}) - \sum_{x_1} P(x_1) \ln P(x_1) - \sum_{x_2} P(x_2) \ln P(x_2) \Big).$$

In general, this does not equal the exact entropy $H(P) = -\sum_{x_1,x_2,x_3} P(\mathbf{x}) \ln P(\mathbf{x})$. However, in the special case when the joint distribution $P(\mathbf{x})$ is uniform, $H_{\mathcal{R}} = H(P)$, because $H(P) = -\ln P(\mathbf{x})$. Similarly, $\sum_{\mathbf{x_{R_i}}} P(\mathbf{x_{R_i}}) \ln P(\mathbf{x_{R_i}}) = \ln P(\mathbf{x_{R_i}})$, so

$$H_{\mathcal{R}} = -\ln P(x_1,x_2) - \ln P(\mathbf{x}) + \ln P(x_1) + \ln P(x_2) = -\ln \frac{1}{n_1 n_2 n_3} = -\ln P(\mathbf{x}) = H(P),$$

where n_i is the number of values that X_i can take. It is apparent that equality arises in this case because the approximation is valid. For instance if R_3 is removed, the region-based entropy is not exact when the joint distribution is uniform. □

Proposition 5.1.2 *For all $R \in \mathcal{R}$, if $b_R(\mathbf{x_R}) = P(\mathbf{x_R})$, and if $P(\mathbf{x})$ is an equiprobable distribution, then the entropy $H_{\mathcal{R}}(\{b_R\})$ is exact, that is $H_{\mathcal{R}}(\{b_R\}) = H(P)$.*

Proof The proof follows from the simple generalisation of the above example. ■

Ideally, the region-based entropy would achieve its maximum when the joint distribution is uniform, thus imitating this key property of exact entropy. This would also yield a desirable region-based approximation because of the above propositions

and the ultimate goal of minimising the region-based free energy would likely imply near-maximum entropy. This suggests the ensuing definition.

Definition A *maxent-normal* constrained region-based approximation is one that is valid and its region-based entropy is maximised when all beliefs $B_R(\mathbf{x}_R)$ are uniform.

Clearly, this type of approximation is open to many specific region and counting number assignments that could prove fruitful in the pursuit of a better approximation. Moreover, many older approximation ideas can be expressed in this convenient form.

5.1.3 Bethe free energy

A classic example is the *Bethe method*, which renowned physicist Hans Bethe devised in 1935 [Bet35]. Moreover, it is a region-based approximation where the set of regions $\mathcal{R} = \mathcal{R}_F \cup \mathcal{R}_X$. There are M regions in $\mathcal{R}_F$ each representing a factor node and its neighbouring variables, and $c_R = 1$ if $R \in \mathcal{R}_F$. Also, there is one region for each variable node in $\mathcal{R}_X$, and $c_{R_{X_i}} = 1 - |N(X_i)|$ if $R_{X_i} \in \mathcal{R}_X$.

Definition The *Bethe free energy* is a function of the beliefs $b_i(x_i), b_j(\mathbf{x}_j)$:

$$F_{Bethe} = U_{Bethe} - H_{Bethe},$$

$$U_{Bethe} = -\sum_{f_j \in F} \sum_{\mathbf{x_j}} b_j(\mathbf{x_j}) \ln f_j(\mathbf{x_j}),$$

$$H_{Bethe} = -\sum_{f_j \in F} \sum_{\mathbf{x_j}} b_j(\mathbf{x_j}) \ln b_j(\mathbf{x_j}) + \sum_{X_i \in \boldsymbol{X}} (|N(X_i)| - 1) \sum_{x_i} b_i(x_i) \ln b_i(x_i).$$

Definition The *constrained Bethe free energy* is Bethe free energy with beliefs (denoted B, signifying that they are probability distributions) abiding the following constraints for all variables X_i and factors f_j:

- *Normalisation*: $\sum_{x_i} B_i(x_i) = \sum_{\mathbf{x}_j} B_j(\mathbf{x}_j) = 1$,
- *Consistency*: $B_i(x_i) = \sum_{\{\mathbf{x_j}: X_i = x_i\}} B_j(\mathbf{x}_j)$, and
- *Inequality*: $B_i(x_i), B_j(\mathbf{x}_j) \geq 0$.

Theorem 5.1.3 *For all regions R in the Bethe method, if $B_R(\mathbf{x_R}) = P(\mathbf{x_R})$ then the exact variational FE is equal to the Bethe FE for factor trees.*

Proof By Proposition 5.1.1, the exact average energy U equals the Bethe average energy, since Bethe FE is valid. Similarly, by Proposition 3.3.1, the Bethe entropy is also exact because in trees,

$$P(\mathbf{x}) = \left(\prod_{f_j \in F} P(\mathbf{x}_j) \right) / \left(\prod_{X_i \in \boldsymbol{X}} P(x_i)^{|N(X_i)|-1} \right).$$

Then applying this equation to $H(P)$ equates to H_{Bethe}. ■

Proposition 5.1.4 *For the constrained Bethe approximation, the global maximum of the region-based entropy H_{Bethe} is when the beliefs $B_R(\mathbf{x_R})$ are uniform for all regions $R \in \mathcal{R}$, i.e. the Bethe approximation is maxent-normal.*

Proof

$$\begin{aligned} H_{Bethe} &= \left(\sum_{f_j \in F} H_j(B_j) - \sum_{X_i \in \boldsymbol{X}} |N(X_i)| H_i(B_i) \right) + \sum_{X_i \in \boldsymbol{X}} H_i(B_i) \\ &= \sum_{f_j \in F} \left(H_j(B_j) - \sum_{X_i \in N(f_j)} H_i(B_i) \right) + \sum_{X_i \in \boldsymbol{X}} H_i(B_i). \end{aligned}$$

Expanding the contents of the summation over factors and using the consistency constraint yields

$$\sum_{\mathbf{x_j}} B_j(\mathbf{x_j}) \ln B_j(\mathbf{x_j}) - \sum_{\mathbf{x_j}} B_j(\mathbf{x_j}) \ln B_k(x_k) - \sum_{X_i \in N(f_j) \setminus \{X_k\}} H_i(B_i).$$

This is at most $-\sum_{X_i \in N(f_j) \setminus \{X_k\}} H_i(B_i) \leq 0$ because $KL(B_j, B_k) \geq 0$ and entropy is non-negative. Clearly, setting all beliefs to have a uniform distribution results in equality for this sum. It also maximises the sum over variables because of the well-known fact that entropy $H(P)$ is maximised when P is uniform. Hence the Bethe entropy is maxent-normal. ■

5.1.4 Bethe free energy and belief propagation

Why would Bethe free energy and belief propagation be connected? Both find distributions $B(x_i), B(\mathbf{x_j})$ which approximate the respective marginal distribution of the joint distribution $P(\mathbf{x})$. Furthermore, both techniques exploit the structure of the graph and perform computations on smaller regions that cumulatively, facilitates in the approximation of some global function. However, the two approximation techniques reach this distribution using slightly different approaches, so certain adjustments need to be made to corollate the two approximations. Yedidia proves the subsequent theorem [YFW02], and uses Lagrange multipliers as described in Appendix A. Also observe that the constraints of constrained Bethe free energy are linear, so Theorem A.4.2 applies.

Theorem 5.1.5 *The interior stationary points of the constrained Bethe free energy are BP fixed points.*

Proof To determine when Bethe free energy is minimised, construct a Lagrangian L of F_{Bethe} and its constraints, which produces the desired stationary points. For all variables X_i and factors f_j, the conditions are

- normalisation $n_i(\mathbf{x}) = \sum_{x_i} B_i(x_i) - 1, n_j(\mathbf{x}) = \sum_{\mathbf{x_j}} B_j(\mathbf{x_j}) - 1$,
- consistency $c_{ji}(\mathbf{x}) = B_i(x_i) - \sum_{\{\mathbf{x_j}: X_i = x_i\}} B_j(\mathbf{x}_j)$, and
- inequality $g_j(\mathbf{x}) = -B_j(\mathbf{x_j})$.

The inequality conditions are all inactive when considering interior points, so their Lagrange multipliers are necessarily zero (see Theorem A.3.1). Then the Lagrangian L is a function of the beliefs B_i, B_j and multipliers $\gamma_{X_i}, \gamma_{f_j}, \lambda_{ji}$:

$$\begin{aligned} L &= \Big(-\sum_{X_i \in \boldsymbol{X}} (|N(X_i)| - 1) \sum_{x_i} B_i(x_i) \ln B_i(x_i) \\ &+ \sum_{f_j \in F} \sum_{\mathbf{x_j}} B_j(\mathbf{x_j}) \ln B_j(\mathbf{x_j}) - \sum_{f_j \in F} \sum_{\mathbf{x_j}} B_j(\mathbf{x_j}) \ln f_j(\mathbf{x_j}) \Big) \\ &+ \sum_{X_i \in \boldsymbol{X}} \gamma_{X_i} n_i(\mathbf{x}) + \sum_{f_j \in F} \gamma_{f_j} n_j(\mathbf{x}) \\ &+ \sum_{X_i \in \boldsymbol{X}} \sum_{f_j \in N(X_i)} \sum_{x_i} \lambda_{ji}(x_i) [B_i(x_i) - \sum_{\{\mathbf{x_j} : X_i = x_i\}} B_j(\mathbf{x}_j)]. \end{aligned}$$

Then $\frac{\partial L}{\partial B_i(x_i)} = 0, \frac{\partial L}{\partial B_j(\mathbf{x_j})} = 0$ yield equations for the beliefs at the stationary points that are proportional to the BP fixed-point beliefs for consistency Lagrange multipliers when $\lambda_{ji}(x_i) = \ln(X_i \to f_j(x_i))$:

$$B(\mathbf{x_j}) = f_j(\mathbf{x_j}) \exp\left(-\gamma_{f_j} - 1 + \sum_{X_i \in N(f_j)} \lambda_{ji}(x_i) \right),$$

$$B(x_i) = \exp\left(\frac{1}{|N(X_i)| - 1} \left[\gamma_{f_j} + \sum_{f_j \in N(X_i)} \lambda_{ji}(x_i) - 1 \right] \right).$$

One technicality is that the beliefs of leaf variables are excluded from the analysis because they do not influence the Bethe free energy, but it is a minor detail because leaf variable beliefs are easy to compute from the belief of their neighbouring factor.

■

Theorem 5.1.6 *The BP fixed points that are positive for all* $\mathbf{x}$ *are also interior stationary points of the constrained Bethe free energy.*

Proof Given the convergence of belief propagation to yield beliefs, recall the messages and belief equations (assuming everything is normalised):

- $X_n \to f_m(x_n) = \prod_{f_{m'} \in N(X_n) \setminus \{f_m\}} f_{m'} \to X_n(x_n)$,
- $f_m \to X_n(x_n) = \sum_{\mathbf{x_m} : X_n = x_n} f_m(\mathbf{x_m}) \prod_{X_{n'} \in N(f_m) \setminus \{X_n\}} X_{n'} \to f_m(x_{n'})$,
- $B(x_i) = \prod_{f_m \in N(X_i)} f_m \to X_i$,
- $B(\mathbf{x_j}) = f_j(\mathbf{x_j}) \prod_{X_i \in N(f_j)} X_i \to f_j$.

As in Theorem 5.1.5, the crucial connection between Bethe free energy and belief propagation is via $\lambda_{ji}(x_i) = \ln X_i \rightarrow f_j(x_i)$. Substituting this into the messages produces

$$X_i \rightarrow f_j(x_i) = \exp(\lambda_{ji}(x_i)),$$

$$f_j \rightarrow X_i(x_i) = \exp\left(\frac{2 - |N(X_i)|}{|N(X_i)| - 1}\lambda_{ji}(x_i) + \frac{1}{|N(X_i)| - 1}\sum_{f_k \in N(X_i): k \neq j} \lambda_{ki}(x_i)\right).$$

Substituting these messages into the belief equations results in equations proportional to the interior stationary points achieved in Theorem 5.1.5.

$$\begin{aligned} B(\mathbf{x_j}) &= f_j(\mathbf{x_j}) \exp\left(\textstyle\sum_{X_i \in N(f_j)} \lambda_{ji}(x_i))\right), \\ B(x_i) &= \exp\left(\frac{1}{|N(X_i)|-1} \textstyle\sum_{f_j \in N(X_i)} [(2 - |N(X_i)|)\lambda_{ji}(x_i) + \lambda_{ji}(x_i)]\right) \\ &= \exp\left(\frac{1}{|N(X_i)|-1} \textstyle\sum_{f_j \in N(X_i)} \lambda_{ji}(x_i)\right). \end{aligned}$$

■

Theorem 5.1.7 *If all factors of a factor graph are positive, then all belief propagation beliefs are interior stationary points of the constrained Bethe free energy.*

Proof In fact, if all factors of a factor graph are positive, then all fixed points of belief propagation are always positive. This follows from the initialisation of the variable messages to 1 as described in Section 4.2.2 and the application of the messages defined in Section 4.2.1. If all the messages are always positive, then it holds that the fixed points, which is the computation of the beliefs, must also be positive because this computation is a product of converged messages. Consequently, they are all interior stationary points of Bethe free energy by Theorem 5.1.6. ■

Theorem 5.1.8 *If all factors are positive, then all local minima of constrained Bethe free energy are BP fixed points.*

Proof To prove this theorem, it suffices to show that all local minima of the constrained Bethe free energy are interior minima. Then Theorem 5.1.5 validates this theorem. This proof generalises Yedidia's Theorem 9 [YFW01].

Suppose that there is a local minimum of Bethe free energy that is not an interior minimum. That is, there is a region belief such that for some assignment of its variables, $B_R(\mathbf{x}_\mathbf{R}^*) = 0$. Assume that each variable $X_i \in \{1, 2, \ldots, n\}$. The basic idea is to first modify B_R by a small amount so that it is no longer an edge point, that is no assignment of variables makes B_R zero. Then show that this new B_R results in a lower free energy thus proving the original B_R could not have been a local minimum.

One case is that no variable region has belief $B_i(x_i) = 0$, so $B_j(\mathbf{x}_\mathbf{j}^*) = 0$ and let $|N(f_j)| = k \geq 2$ (dealing with $k = 1$ is trivial). The main issue in modifying B_j is that the consistency and normalisation constraints must be maintained. The two subcases are when $\mathbf{x}_\mathbf{j}^* = (c, c, \ldots, c)$ and when it is not on the "diagonal". All unspecified beliefs below remain unchanged from their old belief.

On diagonal

- $B_j(\mathbf{x}_\mathbf{j}^*) = \epsilon(n^{k-1} - 1)$,
- $B_j(\mathbf{x_j}) = B_j^{old}(\mathbf{x_j}) - \epsilon$, if at least one variable $X_i = x_i^*$, and $\mathbf{x_j} \neq \mathbf{x_j^*}$,
- $B_j(\mathbf{x_j}) = B_j^{old}(\mathbf{x_j}) + \epsilon(n^{k-1} - (n-1)^{k-1})$, if $\mathbf{x_j} \neq \mathbf{x_j^*}$ is on the diagonal.

Off diagonal

- $B_j(\mathbf{x}_\mathbf{j}^*) = \epsilon((n-1)^{k-1} - 1)$,
- $B_j(\mathbf{x_j}) = B_j^{old}(\mathbf{x_j}) - \epsilon$, if at least one variable $X_i = x_i^*$ and $\mathbf{x_j} \neq \mathbf{x_j^*}$, and $\mathbf{x_j}$ is not on the diagonal,
- $B_j(\mathbf{x_j}) = B_j^{old}(\mathbf{x_j}) + \epsilon(n^{k-1} - (n-1)^{k-1})$, if $\mathbf{x_j}$ is on the diagonal and variable $X_i \neq x_i^*$.

- $B_j(\mathbf{x_j}) = B_j^{old}(\mathbf{x_j}) + \epsilon(n^{k-1} - (n-1)^{k-1} - 1)$, if $\mathbf{x_j}$ is on the diagonal and variable $X_i = x_i^*$.

Verifying normalisation: On the diagonal, there are $(n^k - 1 - (n-1)^k)$ instances of the second item (all possible assignments except those that do not contain any variable set $X_i = x_i^*$), $(n-1)$ of the third, and one of the first. Checking normalisation equates to summing all the epsilons to ensure they sum to 0:

$$(n^{k-1} - 1) + (n^k - n(n-1)^{k-1} - n^{k-1} + (n-1)^{k-1}) - (n^k - 1 - (n-1)^k) = 0.$$

Similarly off the diagonal, let d be the number of variables in $\mathbf{x_j^*}$ with a distinct assignment. Then there are $(n^k - 1 - (n-1)^k - d)$ instances of the second item, $n - d$ of the third, d of the fourth, and one of the first, so

$$((n-1)^{k-1} - 1) + (n-d)(n^{k-1} - (n-1)^{k-1}) + d(n^{k-1} - (n-1)^{k-1} - 1) - (n^k - 1 - (n-1)^k - d) = 0.$$

Verifying consistency: Likewise, checking consistency corresponds to ensuring the sum of epsilons for a given variable $X_i = x_i$ equals 0. In fact, the main modifications to the belief B_j arise from the aim to satisfy these constraints. Recall

$$B_i(x_i) = \sum_{\{\mathbf{x_j} : X_i = x_i\}} B_j(\mathbf{x_j}),$$

so there are n^{k-1} terms in the summation.

Firstly on the diagonal, if $X_i = x_i^*$, all terms have x_i^* in their assignment and exactly one term is $B_j(\mathbf{x_j^*})$, so the epsilons clearly sum to zero. Secondly, if $X_i \neq x_i^*$, $(n-1)^{k-1}$ terms have no variable set to x_r^*, for any $X_r \in N(f_j)$, so $n^{k-1} - (n-1)^{k-1}$ terms have at least one variable set to x_r^* (but not all since $X_i \neq x_i^*$). Moreover, there is exactly one term where $\mathbf{x_j} = (c, c, \ldots, c)$ so that its modification falls under the category of the third item because $X_i \neq x_i^* = 1$.

Off the diagonal, if $X_i = x_i^*$, then one term is $B_j(\mathbf{x_j^*})$, another is the diagonal (fourth item), and the rest have x_i^*, and simply combining the epsilons totals zero. If $X_i = x_l^* \neq x_i^*$, the diagonal term is like in the previous case, and $n^{k-1} - (n-1)^{k-1} - 1$ other terms also contain a variable in $\mathbf{x_j^*}$, which adds up splendidly to zero. Lastly, if $X_i = x_i$ is not a variable in $\mathbf{x_j^*}$, it has one diagonal term corresponding to the third item in the above list and $n^{k-1} - (n-1)^{k-1}$ terms that contain a variable in $\mathbf{x_j^*}$. Again, addition of the epsilons clearly nullifies the modification to satisfy consistency. Hence consistency holds.

Finally, the new belief B_j has a lower free energy than B_j^{old}, for sufficiently small $\epsilon > 0$. With a small ϵ, (1) the modifications to $B_j(\mathbf{x_j})$ where $\mathbf{x_j} \neq \mathbf{x_j^*}$ will negligibly affect their average energy and entropy terms. Most importantly, (2) a sufficiently small ϵ guarantees that the energy from the assignment $\mathbf{x_j^*}$ is less than its entropy, yielding a decrease in free energy from the old belief B_j^{old}. As is clear, if $f_j(\mathbf{x_j}) = 0$, gaining an infinite average energy is likely, thus the condition of positive factors is necessary. Also observe that the above modification does not work if there is another $B_j(\mathbf{x_j}) = 0$ where at least one $X_i = x_i^*$ because inequality will fail. Basically, the extension entails setting the belief of each such assignment to the same assignment as $B_j(\mathbf{x_j^*})$, adjusting the other modifications appropriately, and all of the other affected belief assignments. Suffice to say that this is even messier and the effect on the overall free energy is similar. Furthermore, it is clear that this process can continue *ad nauseam*, since each subsequent zero assignment involving the new considered variables must also be included. For instance, consider $k = 2$ and two zero assignments $(x_i^*, y_i), (x_i^*, z_i)$.

- $B_j(\mathbf{x_j}) = (n-4)\epsilon$, if $B_j^{old}(\mathbf{x_j}) = 0$,
- $B_j(\mathbf{x_j}) = B_j^{old}(\mathbf{x_j}) - 2\epsilon$, if $\mathbf{x_j} = (x_i^*, \alpha)$, $\alpha \neq x_i^*$,

- $B_j(\mathbf{x_j}) = B_j^{old}(\mathbf{x_j}) - \epsilon$, if $\mathbf{x_j} = (\alpha, c)$, $\alpha \neq x_i^*$, $c \in \{y_i, z_i\}$,
- $B_j(\mathbf{x_j}) = B_j^{old}(\mathbf{x_j}) + 2\epsilon$, if $\mathbf{x_j} = (l, l)$ for $l \in \{1, 2, 3, \ldots, n\}$,
- $B_j(\mathbf{x_j}) = B_j^{old}(\mathbf{x_j})$, otherwise.

Verification of the constraints and the decreasing free energy simply follows in the same manner as the simpler case discussed above. Also observe an example of the aforementioned problem of the repetitive application of this procedure: if there is yet another assignment involving z_i or y_i whose belief B_j is zero, this modification must be altered yet again. However, the main point is that the adjustments always add or subtract $O(\epsilon)$ from the old beliefs so that the above argument will always apply.

The second case is when a variable region has $B_i(x_i^*) = 0$. Also suppose only one variable has such a zero belief. A similar process can be used to derive the more general setting where several variables have each have at least one zero belief. Then for belief B_i,

$$B_i(x_i^*) = \epsilon;\ B_i(x_i) = B_i^{old}(x_i) - \epsilon/(n-1), \text{if } x_i \neq x_i^*.$$

Observe that for $f_j \in N(X_i)$, $B_j(\mathbf{x_j}) = 0$ if $X_i = x_i^*$, thus an adjustment of each B_j is also in order. For $|N(f_j)| = k \geq 2$,

- $B_j(\mathbf{x_j}) = \epsilon/(n^{k-1} - 1)$, if $X_i = x_i^*$ and $\mathbf{x_j}$ is on the diagonal,
- $B_j(\mathbf{x_j}) = \epsilon(n^{k-1} - 2)/(n^{k-1} - 1)^2$, if $X_i = x_i^*$, and $\mathbf{x_j}$ is not on the diagonal,
- $B_j(\mathbf{x_j}) = B_j^{old}(\mathbf{x_j}) - \epsilon\frac{n^{2k-3}-2n^{k-2}+1}{n^{k-2}(n-1)(n^{k-1}-1)^2}$, if some variable $X_l = x_i^*, l \neq i$,
- $B_j(\mathbf{x_j}) = B_j^{old}(\mathbf{x_j}) - \epsilon\frac{(n-1)^{k-1}(n^{2k-3}-2n^{k-2}+1)-n^{k-1}+n^{k-2}}{n^{k-2}(n-1)(n^{k-1}-1)^2}$, if $\mathbf{x_j} = (c, c, \ldots, c), c \neq x_i^*$.

Verifying normalisation: The normalisation constraint for $B_i(x_i)$ trivially holds. For the factors adjacent to X_i, there is one occurrence of the first item, $n^{k-1} - 1$ of the

second, $n^k-(n-1)^k-n^{k-1}$ of the third, and $(n-1)$ of the last. Summing the epsilons is simple though excruciating, but does in fact sum to zero, so the normalisation constraint holds for $B_j(\mathbf{x_j})$.

Verifying consistency: Recall $B_l(x_l) = \sum_{\{\mathbf{x_j}:X_i=x_i\}} B_j(\mathbf{x_j})$, which has n^{k-1} terms in the summation. If $l = i$ and $X_i = x_i^*$, the summation equals $B_i(x_i^*) = \epsilon$ because there is one occurrence of the first item in the list and the rest fall in the case of the second item, which means the sum of the epsilons trivially sums to ϵ. If $l = i$ and $X_i \neq x_i^*$, then one term falls under the case of the fourth item and $(n^{k-1}-(n-1)^{k-1})$ terms fit in domain of the third item. This situation demands that the summation yields $B_i(x_i) = B_i^{old}(x_i) - \epsilon/(n-1)$, which a bit of work reveals to be true.

Similarly, if $l \neq i$, the summation of epsilons must equal zero, because $B_l(x_l)$ maintains the same value as $B_l^{old}(x_l)$. Again, there are two subcases. If $X_l = x_i^*$, the first item has one term, the second item has $n^{k-2}-1$ terms, and the third item has the remainder of terms. Lastly, if $X_l \neq x_i^*$, the fourth item has one term, the second item has n^{k-2} terms and the third item has $(n_{k-1}-n^{k-2}-(n-1)^{k-1})$ terms. Showing that these epsilons sum to zero is a simple, though tedious exercise.

It remains to be shown that the overall free energy of the modification is less than the original free energy. With a sufficiently small $\epsilon > 0$, the change to B_i will add $(|N(X_i)|-1)O(\epsilon \ln \epsilon)$ to the entropy, which increases the overall free energy. Also, tweaking B_i forced revisions with each belief of X_i's neighbours, which further affects the free energy. It is easy to guarantee that upon applying these changes to B_j, the entropy will increase. The only beliefs $B_j(\mathbf{x_j})$ of note are those with $X_i = x_i^*$, which were originally zero, and each B_j applies the same modifications so

$$|N(X_i)|\left[\epsilon\frac{1}{n^{k-1}-1}(\ln\epsilon - \ln(n^{k-1}-1)) + \epsilon\frac{n^{k-1}-2}{n^{k-1}-1}\left(\ln\epsilon + \ln\left(\frac{n^{k-1}-2}{n^{k-1}-1}\right)\right)\right]$$

So with constant C, this equates to $|N(X_i)|\epsilon\ln\epsilon + C\epsilon < |N(X_i)|\epsilon\ln\epsilon - \epsilon\ln\epsilon$, for

sufficiently small $\epsilon > 0$. The right-hand side is the contribution from $B_i(x_i^*)$, so overall the entropy increases. Finally, the change in average energy is at most $D\epsilon$ for a constant D and for sufficiently small ϵ, this change is less than the change in entropy, so the overall free energy decreases. At last, this interminable proof is finally complete. The general case where there are more assignments yielding zero beliefs can follow the same labyrinthine template. ■

Theorem 5.1.9 *If all factors are positive, there is a BP fixed point.*

Proof This theorem is a consequence of previous theorem - Bethe free energy is bounded below because factors are nonnegative so there is a global minimum which by Theorem 5.1.7 means that there is a BP fixed point. ■

The consequence of this theorem is that for any factor graph whose factors are all positive, there is always potential for the convergence of belief propagation. Perhaps the clearest results thus far are Theorems 5.1.5 and 5.1.7, namely that interior stationary points of Bethe free energy are BP fixed points, and if the factors are positive, then BP fixed points are interior stationary points of Bethe free energy. The most significant snag rests in the situation when there exists a factor with some assignment such that $f_j(\mathbf{x_j}) = 0$. Heskes aids somewhat in the resolution of this issue, demonstrating that stable BP fixed points are local minima of Bethe free energy, and that the converse is false using stability analysis [Hes03]. Finally, subsequent work by Heskes clarifies some sufficient conditions for unique BP fixed points, associating them with Bethe free energy, with the ultimate goal of providing better sufficient conditions for the convergence of belief propagation [Hes04]. Based on this relationship, Yedidia introduces a generalised belief propagation algorithm that exploits the notion of region-based approximation which could potentially outperform belief propagation [YFW01].

5.2 LDPC Codes

Once again, we embark on an apparent diversion, this time to the preeminent application of belief propagation, the decoding of LDPC codes.

5.2.1 Definitions

In 1962, Gallager pioneered low-density parity-check (LDPC) codes, which are particularly good linear codes that use a sparse parity-check matrix [Gal63]. First, some basic definitions and intuition are necessary to describe and understand these codes.

Error correction of corrupted data is of primary interest in information theory. Not surprisingly, practical transmissive channels and storage devices are imperfect, and thus the distortion of data is a critical issue in need of solutions. Hence instead of merely sending the unmodified data, one should be inclined to include some form of additional information to assist in an unblemished exchange. In information theory, the set of messages that can be sent through a channel is called a code.

Definition A *channel* Q is denoted as $Q = \{A, O, P\}$, where A is the input alphabet, O is the output alphabet, and for any $x \in A, y \in O$, P includes the transition probability distribution $P(y|x)$, and the channel input distribution $P(x)$. Intuitively, the *capacity* of a channel Q is the most information that Q can transmit. For more information on mathematical details, refer to a text that covers information theory [Mac03].

Definition A *code* C is a subset of the set of all possible permutations of n input symbols, i.e. $C \subseteq A^n$, where A is the input alphabet (e.g. a binary code has $A = \mathbb{Z}_2$). Elements of C are called *codewords*. The *rate* of code C is

$$R = \frac{\log|C|}{n}.$$

Note that the definition for a channel applies for a single input symbol, so one must extend the channel definition to allow for the transmission of codewords.

Definition The *nth extension* Q_n of a channel consists of n independent copies of the original channel $Q = \{A, O, P\}$ so that $Q_n = \{A^n, O^n, P_n\}$, where A^n is the input alphabet, and O^n is the output alphabet. Lastly, assuming independence of the n copies of Q, $\mathbf{x} \in A^n$, and $\mathbf{y} \in O^n$,

$$P_n(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} P(y_i|x_i) \text{ and } P_n(\mathbf{x}) = \prod_{i=1}^{n} P(x_i).$$

In the nascence of information theory, Claude Shannon proved that there is a code that can transmit at a rate arbitrarily close to the capacity of any channel with an arbitrarily small error probability [Sha48]. Unfortunately, despite considerable effort, no one has uncovered such a remarkable and elusive code. Still, many types of codes are good because they can be used to transmit with arbitrarily small probability of error up to some rate close, sometimes arbitrarily close, to the capacity of some channels.

Definition A binary code C is a *linear code* if for codewords $\mathbf{u}, \mathbf{v} \in C$, and constant $c \in \mathbb{Z}_2$, $\mathbf{u} + \mathbf{v} \in C$, and $c\mathbf{u} \in C$ (modulo 2 arithmetic). Equivalently, one can define the linear code as the set of solutions $\mathbf{x}$ of the equation $H\mathbf{x}^T = 0^T$, where H is called a *parity-check matrix*. Each row of H is called a *parity-check equation*.

LDPC codes are an example of a sparse graph code, which consists of linear codes with sparse parity-check matrices, and is among several such codes displaying exceptional error-correction capabilities. It is worth noting that other sparse graph codes such as turbo codes can also be decoded using belief propagation [MMC98]. Finally, the following is a simple definition of LDPC codes.

Definition (d_v, d_c)-*regular LDPC code* is a binary linear code determined by the condition that every codeword bit participates in exactly d_v parity-check equations and every parity-check equation involves exactly d_c codeword bits. In other words, the parity-check matrix has exactly d_v ones in each column and exactly d_c ones in each row.

For the remainder of this discussion, assume that there are n codeword bits or variables and m parity-check equations or checks. Moreover, the definition for LDPC codes can be weakened slightly. In particular, one can define irregular LDPC codes in a similar fashion where each variable participates in a varying number of checks, and similarly, each check has a varying number of variables.

As mentioned earlier, the application of belief propagation to decode LDPC codes is one of the great successes of the message-passing algorithm. Richardson and Urbanke contribute some results that further the theoretical understanding of both LDPC codes and belief propagation [RU01]. As such, a few more definitions given by Richardson and Urbanke are useful in this analysis.

Definition For any graph G and vertex $u \in V(G)$, the *u-neighbourhood N_u^d of depth d* is the induced subgraph $G[A]$, where A contains *all* vertices such that for any $v \in V(G)$, $v \in A$ iff $d(u, v) \leq d$, so A can contain duplicate vertices.

Definition For any graph G and $e = uv \in E(G)$, the *undirected e-neighbourhood N_e^d of depth d* is defined as the set of edges whose endpoints are in $N_u^d \cup N_v^d$. For $\vec{e} = (u, v)$, where $uv \in E(G)$, the *directed $\vec{e}$-neighbourhood $N_{\vec{e}}^d$ of depth d* contains all edges and nodes in paths $\vec{e_1}\vec{e_2}\ldots\vec{e_d}$ starting from u where $\vec{e_1} \neq \vec{e}$ and $\vec{e_i}$ is a directed version of an edge in $E(G)$.

Definition If the induced subgraph of a neighbourhood is a tree, then the neighbourhood is *tree-like*. Clearly, a neighbourhood is tree-like iff all involved vertices are distinct.

Proposition 5.2.1 *Let $2 \leq d_v < d_c$. For any X, Y-bipartite graph where $deg(u_X) \leq d_v$ and $deg(u_Y) \leq d_c$ for $u_X \in X, u_Y \in Y$, the number of distinct vertices in any of the predefined neighbourhoods of depth $2l$ is upper bounded by $2(d_v d_c)^l$.*

Proof Follows by induction on l. Firstly, the neighbourhood has the most distinct vertices if it is a (d_v, d_c)-regular tree, so it suffices to consider the number of distinct vertices in this case. For instance, this is the proof for the u-neighbourhood N_u^{2l}. The base case is for depth 2, which contains $1 + d_v + d_v d_c \leq 2(d_v d_c)$ distinct vertices.

Suppose the proposition holds for depth $2l$. By induction and observation, there are at most $2d_v^l d_c^l + d_v^l d_c^{l+1} + d_v^{l+1} d_c^{l+1}$ distinct vertices in $N_u^{2(l+1)}$. This is at most $2d_v^{l+1} d_c^{l+1}$ vertices because $2 \leq d_v < d_c$, so the proposition holds. ■

5.2.2 Concentration and Convergence

The following is an attempt to generalise the proof of Richardson and Urbanke's theorems in Section 4 [RU01]. The basic strategy is to break up the two major components into two separate theorems: the first on the concentration around the expected value and the second on the convergence to the cycle-free case. The proofs are essentially equivalent to the ones given in the aforementioned paper. It is easiest to first consider the case in which both sides of the bipartite graph are regular, i.e. regular LDPC codes.

Definition For any iteration l, The *discrepancy* of an edge e in a labelled bipartite graph G is $\delta_l(e)$, where $0 \leq \delta_l(e) \leq 1$. The total discrepancy of G is $Z^l = \sum_{e \in E(G)} \delta_l(e)$.

Definition *Locality* holds if a message (and discrepancy) across an edge $\vec{e}$ depends only on the directed neighbourhood $N_{\vec{e}}^{2l}$ in the l^{th} iteration.

The first theorem relies on a message-passing algorithm with the Locality property. It may be possible to relax the definition for discrepancy. The theorem relies on Z being the sum over all edges of some measure of the messages across edges in order derive a suitable bound for Claim 1. This measure should bounded by some constant. In this case, the measure (discrepancy) is between 0 and 1. In the case of LDPC codes, the discrepancy is an indicator variable denoting whether the message from variable to constraint is wrong, which is sufficient to determine the validity of the message-passing algorithm.

Lemma 5.2.2 *(Azuma's inequality) If $Z_0, Z_1, \ldots$ is a martingale sequence such that for each $k \geq 1$, $|Z_k - Z_{k-1}| \leq \alpha_k$, then for all $l \geq 0$ and any $\lambda > 0$,*

$$\mathbf{Pr}[|Z_l - Z_0| \geq \lambda] \leq 2 \exp\left(\frac{-\lambda^2}{2\sum_{k=1}^{l} \alpha_k^2}\right).$$

For more information on this inequality and martingales, refer to a text covering probability theory [MR95].

Theorem 5.2.3 *For a given d_v, d_c, l: consider the probability space over all labelled (d_v, d_c)-regular graphs $G^n(d_v, d_c)$ with n variables and received values of all n variables R, let $Z = Z^l$. Then there exists a constant $\beta > 0$ such that for any $\epsilon > 0$,*

$$\mathbf{Pr}[|Z - \mathbf{E}[Z]| > nd_v\epsilon/2] \leq 2e^{-\beta\epsilon^2 n}.$$

Proof First construct a martingale on the random variable Z which is also a graph-theoretic function. Formally, a particular instance in the probability space should be denoted as (G, R). Since the first part of the proof deals only with G, R is dropped for the sake of clarity whenever viable.

As a brief preamble, consider the order of the edges of a graph in $G^n(d_v, d_c)$. Label the outgoing edges from variables from $1, \ldots, nd_v$ and similarly, label the outgoing edges from checks from $1, \ldots, md_c$. Thus, one can think of the edges as a pair of these labels. Let the first i edges refer to the edges connected with labels $1, \ldots, i$ from the variables.

Construction Define the equivalence between two graphs in the ensemble $G =_i H$ to mean that the first i edges in G and H are equivalent. Define the martingale sequence $Z_0, Z_1, \ldots, Z_{nd_v}$ by

$$Z_i(G) = \mathbf{E}[Z(H) : G =_i H].$$

Secondly, prove that the difference between consecutive random variables in the martingale sequence is at most some α_i and then apply Azuma's inequality.

Let $\mathcal{G}(G, i)$ be the set of graphs in $G^n(d_v, d_c)$ whose first i edges agree with those of G, that is $\mathcal{G}(G, i) = \{H : H =_i G\}$. Let $\mathcal{G}_j(G, i)$ be the subset of $\mathcal{G}(G, i)$ consisting of those graphs whose $i + 1^{th}$ edge go to the same check label j, Then

$$\begin{aligned} Z_i(G) &= E[Z(H) : H \in \mathcal{G}(G, i)] \\ &= \sum_j E[Z(H) : H \in \mathcal{G}_j(G, i)] \cdot Pr\{H \in \mathcal{G}_j(G, i) : H \in \mathcal{G}(G, i)\}. \end{aligned}$$

Define $\phi_{j,k} : \mathcal{G}_j(G, i) \to \mathcal{G}_k(G, i)$ as the switching operation described in Richardson and Urbanke which essentially switches two edges of the graph [RU01, p.614]. That is, this operation replaces the edge labelled $(i + 1, j)$ and some other edge labelled (x, k) with $(i + 1, k)$ and (x, j). This operation preserves probabilities because every graph occurs with uniform probability.

Claim 1 $|Z(H) - Z(\phi_{j,k}(H))| \leq 8(d_v d_c)^l$

Using Proposition 5.2.1 and locality (a message along any edge e is a function of its directed neighbourhood N_e^{2l}), it follows as that the switching operation affects at

most $8(d_v d_c)^l$ neighbourhoods. Hence Claim 1 follows.

Claim 2 $|\mathbf{E}[Z(H) : H \in \mathcal{G}_j(G,i)] - \mathbf{E}[Z(H) : H \in \mathcal{G}_k(G,i)]| < 8(d_v d_c)^l$

This follows from Claim 1 and because $|\mathbf{E}[W]| \leq \mathbf{E}[|W|]$, using the fact that

$$\mathbf{E}[Z(H) : H \in \mathcal{G}_k(G,i)] = \mathbf{E}[Z(\phi_{j,k}(H) : H \in \mathcal{G}_j(G,i)].$$

The desired inequality follows from this claim:

$$|Z_{i+1}(G) - Z_i(G)| \leq \max_{j,k} |\mathbf{E}[Z(H) : H \in \mathcal{G}_j(G,i)] - \mathbf{E}[Z(H) : H \in \mathcal{G}_k(G,i)]|.$$

Consequently, Azuma's inequality applies, thus completing the proof of the theorem for the exposed edges. For the n messages R received by variables, extend the martingale sequence to be denoted as $Z_1(G,R), \ldots, Z_{nd_v+1}(G,R), \ldots, Z_{(n+1)d_v}(G,R)$. Again by locality, each exposed message received by v affects at most the nodes in N_v^{2l} which is upper bounded by Proposition 5.2.1, appropriately bounding $|Z_{i+1} - Z_i|$ as required. ■

The second theorem requires only the requisite assumptions in the first theorem. The proof is unchanged with nothing important to note, hence there is no need to restate the proof. However, observe that relaxing the constraints on discrepancy would require some changes to the theorem. p is the expected discrepancy along an edge with a tree-like directed neighbourhood of depth at least $2l$. Hence $\mathbf{E}[Z_t] = nd_v p$ represents the expected total discrepancy if every edge has a tree-like directed neighbourhood of depth at least $2l$, i.e. the cycle-free case.

Theorem 5.2.4 *With the same assumptions as the previous theorem, and* $n > 2\gamma/\epsilon$,

$$|\mathbf{E}[Z] - \mathbf{E}[Z_t]| < nd_v\epsilon/2.$$

Corollary 5.2.5 *With the same assumptions as the previous theorem,*

$$\mathbf{Pr}[|Z - \mathbf{E}[Z_t]| > nd_v\epsilon] \leq 2e^{-\beta\epsilon^2 n}.$$

Theorem 5.2.3 states that the actual total discrepancy is concentrated around the expected total discrepancy. Theorem 5.2.4 reports that for a sufficiently large graph, the expected total discrepancy converges to the expected total discrepancy in the cycle-free case. Lastly, the corollary proclaims that the actual total discrepancy is concentrated around the expected total discrepancy of the cycle-free case.

The extension to irregular graphs is not difficult as mentioned in Richardson and Urbanke [RU01]. If the degrees are bounded, the concentration results still hold, but for different constants for the appropriate ensemble of graphs.

While the theorems themselves do not assume anything about the message-passing algorithm besides locality, knowledge of the algorithm is necessary to present something useful. Hence generalising this argument for a problem like k-SAT requires a suitable definition for discrepancy that pertains to the specific message-passing algorithm used for the problem. Moreover, the theorem in its current form demands that the algorithm operates on a labelled bipartite graph. So over graphs in any ensemble of bipartite graphs representing some random constraint satisfaction problem, the total discrepancy is concentrated around the expected total discrepancy in the cycle-free case.

5.3 Empirical performance in some factor graphs

Generally empirical research regarding belief propagation has focused on particular applications [MWJ99]. Instead, this basic implementation aims to assess the applicability of belief propagation on simple classes of graphs with the ultimate intention of analytically characterising on which graphs belief propagation works well. The algorithm is essentially the same as the one discussed in Section 4.2.2 with two minor changes. Firstly, the algorithm normalises the factors and messages to avoid

numerical underflow. Secondly, the algorithm assumes that the computation of the messages equates the message transmission. The algorithm designates convergence to mean that all messages are within a small tolerance (default is 10^{-4}) of their previous values, and non-convergence to mean that the algorithm has not converged within a specified number of iterations (default is 100). The C implementation of the algorithm was run on a large set of randomly generated factor graph classes with varying factor definitions through a simple Perl script. Belief propagation is compared to a slightly refined version of brute force marginalisation to determine the quality of its approximation. Computing the error of a function also requires some consideration. While it is reasonable that the error of a function $B(x_i)$ approximating $P(x_i)$ with value x_i is the traditional relative error $RE(B(x_i)) = |\frac{B(x_i)-P(x_i)}{P(x_i)}|$, it is not entirely clear what should be used to determine the overall error of such a function. For simplicity, equally weighting the relative error of $B(x_i)$ at each $X_i = x_i$ gives a crude characterisation of the overall error of $B(X_i)$. Assuming $|X_i|$ is the number of values X_i can take,

$$Error(B(X_i)) = \frac{1}{|X_i|} \sum_{x_i=0}^{|X_i|-1} RE(B(x_i)).$$

5.3.1 Results

Running belief propagation on three simple types of graphs reveals some of the potential capabilities of the algorithm. All of the performance graphs discussed herein are in Appendix B.

Grids

One easily-generated graph is a pairwise Markov random field converted into a factor graph, which resembles a grid. Briefly, grids are highly structured, where all factors

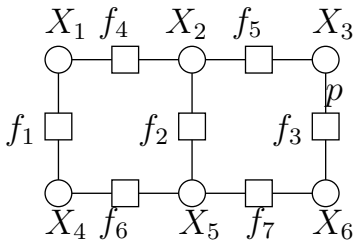

Fig. 5.3: A grid with width $w = 3$, height $h = 2$ and edge probability p.

have degree at most two and variables have degree at most four, and no two variables share more than one common factor. A $w \times h$ grid has wh variables, and a random grid G is simply a grid with the possible edges in $E(G)$ probability p. For instance, see Figure 5.3. The results for some small grids are perhaps surprisingly good, as evidenced in the low error and perfect convergence (see Appendix B). Both the number of variables wh and number of edges appear to be directly proportional to the error and convergence rate, while changing the variables from binary to quaternary has little effect on either measure. This observation is not as surprising because the increase in variables and edges implies an increase in dependencies and complexity of the overall system.

To further assess how limited this good performance is to this specific breed of graph, construct a full grid, that is the random grid with probability $p = 1$, and add all other possible non-grid edges with probability q. The effect on belief propagation's efficacy is drastic, as the error rises sharply even with $q = 0.1$ on small grids. Thus it seems likely that there is some property of a grid that allows belief propagation to function remarkably well.

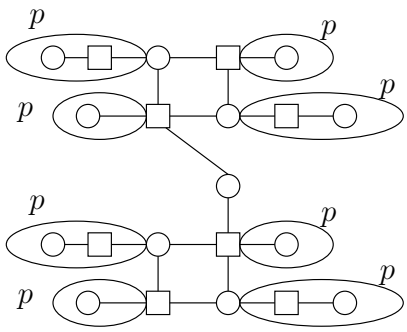

Fig. 5.4: Disjoint $2k$-cycle graph with 2 cycles, $k = 2$, and extra variable probability p.

Disjoint cycles

Another simple type of graph is a graph of disjoint cycles. Considering the only major result for belief propagation is for graphs with a single cycle, it is reasonable to consider the extension to a scenario with several disjoint cycles. For simplicity, each disjoint cycle in the graph has k variables, and thus k factors, and each vertex in cycle has probability p of having an additional variable, and a factor if necessary, attached to it. Intuitively, these graphs are trees of cycles. For instance, see Figure 5.4. Expectably, the results for these graphs are somewhat better than grids, and like grids, these graphs always converged. Since the number of additional leaves would not affect the overall error and convergence, $p = 0.3$, which is a reasonably small value, but large enough so that there will usually be at least one additional leaf. The same notion of dependencies and complexity discussed in relation to grids also applies for disjoint cycle graphs. Increasing the number of disjoint cycles adversely influences the error and convergence rate, as graphs with one cycle have roughly half the error of those with four cycles. More cycles means more dependencies, which tend to negatively impact the effectiveness of belief propagation. However, increasing the size of the cycles decreases the error, which also conforms with belief that larger

cycles have fewer dependencies between all variables. Interestingly, the trend for the convergence rate is not as clear, although the gap between convergence rate of graphs with different cycle size appears directly proportional to the number of cycles. Curiously, all graphs with only one cycle take close to five iterations to converge.

Random graphs

Lastly, one can consider the general case of random graphs, namely given the vertices of a graph $G = (V, E)$, an edge $(v_1, v_2) \in V \times V$ is in E with probability p. While not particularly helpful in isolating the specific types of graphs on which belief propagation runs well, it is still interesting to realise the effect of the number of edges, variables, factors, and values of variables.

Many interesting oddities crop up in the tested cases. Perhaps the simplest pattern in these graphs is that the convergence, unconvergence, and error of random graphs does not depend on whether a variable is binary or quaternary. Another trend in error is that for a factor-to-variable ratio in $[1, 3.5]$, the error grows steadily as the number of edges increases, hence supporting the notion that an increase in dependencies diminishes the effectiveness of belief propagation. For larger ratios, the only clear conclusion is that the error grows drastically as the ratio increases, although for five variables, it actually decreases for $p > 0.5$.

Regarding convergence, the general trends for convergence and unconvergence are similar, which is expected because higher convergence rates would suggest a higher number of unconverged instances. However, it is curious that increasing the number of variables alters the convergence rate drastically as the number of edges increases. For five variables, the convergence rate tends to increase as the number of edges increases, although for factor-variable ratios $[4, 6]$, the convergence rate levels out for $p > 0.4$, while the percentage of unconverged instances grows rapidly for $p > 0.4$.

However, the error remains relatively low when $p > 0.4$, at least compared to the corresponding error for eight and ten variables.

For eight and ten variables, the slowest instances and highest percentage of unconverged instances is around $p \in [0.4, 0.7]$ yielding graphs resembling parabolas, that is the rate of unconvergence is increasing up to some point, and decreasing past that point. Strikingly, as the number of variables increases, the unconvergence rate drops sharply for large p, and correspondingly the speed of convergence rises dramatically. However, the error is quite erratic at these values, which suggests that there is an edge probability p^* such that belief propagation past p^* is no longer reliable even with fast convergence, possibly resulting from how constrained the system is for many variables and large p, and a potential increase in the number of stationary point pitfalls. This p^* seems appears to be consistent for any ratio, although the effect becomes substantially worse for large p^*, and also seems to decrease as the number of variables increase. For instance, it seems $p^* > 0.9$ for five variables, but $p^* \in [0.6, 0.7]$ for eight variables and $p^* \in [0.4, 0.6]$ for ten variables. Still, it is not entirely apparent why belief propagation acts in such a manner for random graphs and this could warrant further investigation.

5.4 Analysis of belief propagation

Section 4.2.1 already proved convergence and exactness of belief propagation in trees, and the correctness of the program for factor trees reinforces the correctness of the theorem and the program. The program demonstrated empirical evidence of some of belief propagation's capabilities, but clearly lacks analytical muscle. Thus far, there are some attempts to determine good sufficiency conditions for convergence of belief propagation [Hes04, IIW05, MK05, TJ01], but most of these are rather inconvenient

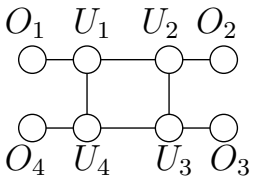

Fig. 5.5: A single-cycle pairwise MRF with variables $U_1, U_2, U_3, U_4, O_1, O_2, O_3, O_4$.

tests. Even less is known about the correctness of belief propagation, with the most notable result being for a single-cycle graph [Wei01].

5.4.1 Convergence

Though recent investigations regarding the convergence of belief propagation yield some preliminary analytical work and interesting avenues of attack, the results seem somewhat unsatisfying. Tatikonda and Jordan essentially use a unique Gibbs measure as sufficient condition for loopy belief propagation [TJ01]. Most recently, Mooij and Kappen give a stronger condition using Banach's fixed-point theorem and analysis pertaining to it [MK05].

5.4.2 Correctness

Aside from the results for a single-cycle graph [Wei01], there is apparently no analytical progress on this front. One possible approach may be to re-examine the belief propagation algorithm through its origins in probability theory. More specifically, this entails how the assumed independences in the belief propagation algorithm affect performance in various graphs. Crudely speaking, it is unsurprising that the algorithm tends to produce more precise results when there are fewer edges relative to the number of vertices because these graphs have a propensity for fewer dependences. In another sense, they are more tree-like than their edge-crammed brethren.

The result of Weiss relies on a pairwise Markov random field (G, P) with $2n$ variables $\mathbf{X} = \{U_1, \ldots, U_n, O_1, \ldots O_n\}$, where $U_1, \ldots, U_n$ are in the cycle and leaf variable O_i neighbours U_i, for $i = 1, \ldots, n$ (see Figure 5.5). His key observation is that in a single cycle, one can express the new message received by a variable in the cycle as a function of the old message received by the same variable. Recall that the MRF message is

$$X \to Y(y) = \sum_x f_{XY}(x, y) \prod_{Z \in N(X) \setminus \{Y\}} Z \to X(x).$$

Thus the MRF message $X \to Y(y)$ can be expressed as a vector $X \to Y$ whose ith value equates to $X \to Y(i)$. Moreover, the MRF message can also be written as a matrix operation, where $\odot$ denotes componentwise multiplication (e.g. $(u_1, u_2) \odot (v_1, v_2) = (u_1 v_1, u_2 v_2)$) and $M_{X \to Y}(i, j) = f_{XY}(j, i)$:

$$X \to Y = M_{X \to Y} \bigodot_{Z \in N(X) \setminus \{Y\}} Z \to X.$$

Then it is not hard to see that the following holds:

$$U_n \to U_1^{new} = C_{U_n \to U_1} \left(U_n \to U_1^{old} \right).$$

To determine $C_{U_n \to U_1}$, note that

$$U_n \to U_1 = M_{U_n \to U_1}(O_n \to U_n \odot U_{n-1} \to U_n) = M_{U_n \to U_1} D_n(U_{n-1} \to U_n),$$

where $D_n(i, j) = O_n \to U_n(i)$ if $i = j$ and 0 otherwise. Repeating this process for $U_{n-1} \to U_n, \ldots, U_1 \to U_2$, which ultimately yields the desired matrix $C_{U_n \to U_1}$. Now the theorem can be stated, and its proof is in the original paper, and also relevant to this discussion, Aji et al. proved a similar result for a single cycle graph [AHM98, Wei01].

Theorem 5.4.1 *[Wei01] For a pairwise Markov random field consisting of a single cycle as specified above, then if all elements of $C_{U_n \to U_1}$ are nonzero, then*

- *$U_n \to U_1$ converges to the principal eigenvector of $C_{U_n \to U_1}$.*
- *$U_2 \to U_1$ converges to the principal eigenvector of $D_1^{-1} C_{U_n \to U_1} D_1$.*
- *The ratio λ_1/λ_2 of the largest eigenvalue λ_1 of $C_{U_n \to U_1}$ and the second-largest eigenvalue λ_2 governs the convergence rate of the messages.*
- *The elements $C_{U_n \to U_1}(i,i)$ give the correct posterior marginal probabilities $P_{U_1}(i) \propto C_{U_n \to U_1}(i,i)$.*
- *The belief $B(U_1)$ is related to $P(U_1)$ by: $B(U_1) = \beta P(U_1) + (1-\beta) Q(U_1)$, where β is the ratio of λ_1 to the sum of all eigenvalues of $C_{U_n \to U_1}$, and $Q(U_1)$ depends on the eigenvectors of $C_{U_n \to U_1}$.*

Chapter 6

Propagation for satisfiability

In recent years, there has been some interest in using message-passing algorithms to solve the venerable propositional satisfiability problem, and this section reviews the framework and message-passing solvers for random 3-SAT [BMZ03, MMW05].

6.1 Definitions

Definition A *literal* is a propositional variable or its negation and a *clause* is a disjunction of literals. *k-CNF formula* $\mathcal{F}$ (conjunctive normal form) on n variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a formula of propositional logic consisting of a conjunction of m clauses $F = \{f_1, \ldots, f_m\}$, each with at most k literals. Let $r = m/n$ be the *clause-to-variable ratio*. Also, $\mathbf{x}$ is an assignment to the variables $\mathbf{X}$, and $\mathbf{x_j}$ is an assignment of the variables $\mathbf{X_j}$, where $\mathbf{X_j} = \{X_i \in \mathbf{X} : \text{either } X_i \text{ or } \neg X_i \text{ is in } f_j\}$ is the set of variables in clause f_j. Finally, another representation of the assignment to variables is the *truth assignment function* $\tau : \{0,1\}^n \to \{0,1\}$.

Note that one can express a formula as a Boolean function $\mathcal{F} : \{0,1\}^n \to \{0,1\}$, where $\mathcal{F}(\mathbf{x}) = 1$ if the assignment $\mathbf{X} = \mathbf{x}$ satisfies the formula $\mathcal{F}$, and $\mathcal{F}(\mathbf{x}) = 0$

otherwise. The Boolean function of a clause is defined in a similar fashion: $f_j : \{0,1\}^{|\mathbf{X_j}|} \rightarrow \{0,1\}$. As a result, a CNF formula can be written in the following familiar form:

$$\mathcal{F}(\mathbf{x}) = \prod_{j=1}^{m} f_j(\mathbf{x_j}).$$

Definition The *k-SAT problem*, or k-satisfiability, can be formulated as the following decision problem: given an instance $\mathcal{F}$ of k-CNF with variables $\mathbf{X}$ and clauses F, does there exist an assignment $\mathbf{x}$ of variables such that $\mathcal{F}(\mathbf{x}) = 1$?

Typically, for the k-SAT problem, it is easier to just consider the k-CNF instances where every clause has exactly k literals. One can also define methods for randomly generating k-CNF formulas.

Definition An instance of *uniform random k-SAT* is a k-CNF formula given n variables and m clauses, or just a ratio r, such that each clause includes k literals chosen at random from the $2n$ possible literals, where the probability of selecting a literal is $1/2n$. A clause is acceptable if they contain at most one literal of any variable. Otherwise, one discards it and tries again.

Intuitively, a small ratio implies many copies of every variable hence yielding an under-constrained instance with many solutions, whereas a large ratio implies an over-constrained instance, where proving unsatisfiability is usually easy. Hence the central goal is to determine a tight interval of *phase transition* $[r^k_{sat}, r^k_{unsat}]$ such that for ratios $r < r^k_{sat}$, almost all k-SAT instances are satisfiable, and for ratios $r > r^k_{unsat}$, almost all k-SAT instances are unsatisfiable. Informally, the difficult instances are usually the ones within this interval, where the problem might have only a few solutions. It appears empirically that the hard region of random 3-SAT occurs at a threshold ratio around 4.2, and some recent progress has managed to tighten the transition

interval around this ratio [AP04, CK02]. Moreover, some evidence maintains that some message-passing algorithms work quite well in this hard region, especially the survey propagation algorithm for 3-SAT [BMZ03].

Belief propagation in factor graphs can also be modified for the satisfiability problem. In particular, each variable is a variable node and each clause is a factor node with a labelled edge to a variable node if the clause contains this variable. The label is simply an indicator for the negation of a variable in a clause. All of the following algorithms rely on message-passing from variable-to-clause (variable messages $X_i \to f_j$) and clause-to-variable (clause messages $f_j \to X_i$). The following formulation parameterises messages in such a way that eliminates the argument $X_i = x_i$, so to distinguish them from the original message definitions, the originals will always contain the argument, i.e. $f_j \to X_i(x)$.

In a factor graph, the first requirement is that the graph represents the global function (3.5), and the representation of the CNF formula $\mathcal{F}$ as a Boolean function maintains this property. Thus belief propagation undergoes two major modifications. One is that the edges are labelled, to account for the presence of negated variables, and the second is that the factors are Boolean. Thus running belief propagation to solve single-variable marginals equates to solving how likely the formula is true given one assigned variable.

Currently, one goal in this area is to understand survey propagation, an extension of belief propagation for the satisfaction problem. Empirical results reveal good performance for belief propagation up a clause-to-variable ratio to around 3.921, and even better performance of survey propagation, which seems to work well for random 3-SAT instances in the aforementioned hard region [BMZ03, MMW05].

6.2 Warning propagation

Warning propagation [BMZ03] is a naïve method of propagation algorithm for satisfiability that serves as an excellent intuitive introduction to propagation algorithms for satisfiability.

6.2.1 Message definitions

The set of neighbours $N(X) = N_+(X) \cup N_-(X)$ consists of clauses containing X or $\neg X$ (the clause is in $N_+(X)$ or $N_-(X)$ respectively). Let indicator functions $I_{>0}(x) = 1$ if $x > 0$ and 0 otherwise and $I_{<0}(x) = 1$ if $x < 0$ and 0 otherwise.

$$X_i \to f_j = \sum_{f_k \in N_+(X_i) \setminus \{f_j\}} f_j \to X_i - \sum_{f_k \in N_-(X_i) \setminus \{f_j\}} f_j \to X_i. \tag{6.1}$$

$$f_j \to X_i = \prod_{X_k \in N_+(f_j) \setminus \{X_i\}} I_{<0}(X_k \to f_j) \prod_{X_k \in N_-(f_j) \setminus \{X_i\}} I_{>0}(X_k \to f_j). \tag{6.2}$$

$$B_i = \sum_{f_j \in N_+(X_i)} f_j \to X_i - \sum_{f_j \in N_-(X_i)} f_j \to X_i. \tag{6.3}$$

$$c_i = 1, \text{ if } \left(\sum_{f_j \in N_+(X_i)} f_j \to X_i \right) \left(\sum_{f_j \in N_-(X_i)} f_j \to X_i \right) > 0 \tag{6.4}$$

$$= 0, \text{ otherwise.} \tag{6.5}$$

The variable-to-clause message (6.1) sends the clause information regarding what value the variable wants to be. More specifically, the variable tells the clause whether it wants to be true or false if a majority of other clauses containing the underlying variable want it to be true or false respectively. If no majority is reached, then the literal does not have information to yearn for truth or falsehood. Thus equation (6.1) is positive if the variable wants to be true, negative if it wants to be false, and zero if it is uncertain.

The clause message $f_j \to X_i$ is an indicator determining whether f_j desires the literal of X_i to be true. It can also be interpreted as a warning sent to a literal demanding that it will be set to be true. Thus transmitting an affirmative indicator, a warning, requires the clause to believe that all its other literals desire to be false. The equation (6.2) reflects this idea. For variable $X_k \in N_+(f_j)$, $I_{<0}((X_k \to f_j)$ is 1 if there are more clauses telling X_k to be false, and 0 otherwise. Similarly, for variable $X_k \in N_-(f_j)$, $I_{>0}((X_k \to f_j)$ is 1 if there are more clauses Hence the product over all other literals of c equates to a warning $(c \to x = 1)$ if and only if all these literals want to be false (all indicators are 1).

The belief computation B_i conveys the same information as $X_i \to f_j$, except it uses all neighbours of X_i, that is it prefers X_i true if $B_i > 0$ and $X_i = 0$ if $B_i < 0$. Finally, the contradiction number c_i determines if a variable has a clause that wants X_i to be true and another that wants $\neg X_i$ to be true, which suggests a contradiction.

Resemblance to the min-sum algorithm

This formulation is a natural extension of the min-sum algorithm, whose message definitions are as follows:

$$X_i \to f_j(x_i) \;=\; \textstyle\sum_{f_k \in N_+(X_i)\setminus\{f_j\}} f_k \to X_i(x_i) - \sum_{f_k \in N_-(X_i)\setminus\{f_j\}} f_k \to X_i(x_i).$$

$$f_j \to X_i(x_i) \;=\; \min_{\mathbf{x_j}: X_i = x_i} f_j(\mathbf{x_j})\Big(\textstyle\sum_{X_k \in N_+(f_j)\setminus\{X_i\}} X_k \to f_j(x_k) - \sum_{X_k \in N_-(f_j)\setminus\{X_i\}} X_k \to f_j(x_k)\Big).$$

$$b(x_i) \;=\; \textstyle\sum_{f_j \in N_+(X_i)} f_j \to X_i(x_i) - \sum_{f_j \in N_-(X_i)} f_j \to X_i(x_i).$$

The negation of messages from negative literals distinguishes these messages from their positive counterparts. On a high level, WP treats the computation of both beliefs of X_i, $X_i = 0$ and $X_i = 1$, is unnecessary. To see this, one can think of the

messages as indicators of the desirability of $X_i = x_i$. Since $f_j(\mathbf{x_j}) = 1$ if $\mathbf{x_j}$ satisfies f_j and 0 otherwise, $f_j \rightarrow X_i(x_i) \leq 0$. Obviously, if $f_j \rightarrow X_i(x_i) = 0$, $X_i = x_i$ would be the least desirable assignment, so f_j must prefer $X_i = 1 - x_i$. The min-sum variable message has the same meaning as the WP variable message: $X_i \rightarrow f_j(x_i)$ tells f_j that either the other clauses containing X_i or the other clauses containing $\neg X_i$ want $X_i = x_i$. The WP clause message (6.2) simplifies the min-sum clause message and treats the incoming desirability information of each $X_k \rightarrow f_j(x_k)$ equally, and effectively determines if the clause wants $X_i = 1$ or not.

6.2.2 Algorithm

As with most propagation algorithms, WP is correct for factor trees. But it can fail when there is a cycle. One minor consolation is that it is in some sense sound when applying a recursive process based on WP (Warning Inspired Decimation or WID).

6.2.3 Correctness

Theorem 6.2.1 *Warning propagation converges and is correct for any factor tree G representing a formula $\mathcal{F}$, that is WP($\mathcal{F}$) returns no contradictory messages iff $\mathcal{F}$ is satisfiable.*

Proof [BMZ03] Convergence follows in the same vein as Proposition 4.2.5 and Corollary 4.2.6. Then consider the set of converged messages $c \rightarrow X$, $X \rightarrow c$ in order to prove correctness. First prove the direction "if WP($\mathcal{F}$) returns a contradictory message, then $\mathcal{F}$ is unsatisfiable". For any edge $cX \in E(G)$, let H_{cX} be the subgraph containing c of the graph G with the edge cX removed.

Claim 1 if $c \rightarrow X = 1$, then c is not true in the subgraph H_{cX} containing c induced by the removal of edge cX.

WP (formula $\mathcal{F}$)

Output: The set of all clause messages and their values (empty if unconverged).

begin
Build graph G out of formula $\mathcal{F}$;
Randomly initialise clause messages to 0 or 1;
converged $\leftarrow$ **false** ;
for $t \leftarrow 1$ **to** T **do**
Randomly order the edges of G and update clause messages in that order;
if *for all clause messages* $f_j \rightarrow_t X_i = f_j \rightarrow_{t-1} X_i$ **then**
converged $\leftarrow$ **true** ;
exit loop.
end
end
if *converged* = **false then**
return $\emptyset$;
else
return set of clause messages.
end
end

WID (formula $\mathcal{F}$, partial assignment τ)

begin

Replace $\mathcal{F}$ with the simplified formula of $\mathcal{F}$ using the partial assignment τ;

clausemessages $\leftarrow$ WP ($\mathcal{F}$);

if *clausemessages* $= \emptyset$ **then**
return "unconverged".

end

Compute the desired value of variables B_i;

Compute the contradiction numbers c_i;

if *there exists* $c_i \neq 0$ **then**
return "unsatisfiable".

else
Extend the assignment τ to τ' as follows: for all variables with a

non-zero H_i, assign $x_i = 1$ if H_i positive and $x_i = 0$ if it is negative;

if *No such variable exists* **then**
Pick an unassigned variable at random and randomly assign it.

end

end

If all variables are fixed, check τ' on $\mathcal{F}$ and return "unsat" or τ';

Otherwise run WID(F, τ');

end

This is an induction on the maximum distance of c from a leaf. The base case is vacuous. Suppose the claim is true for a max distance $< d$, and consider c with max distance d, and $c \to X = 1$. By (6.2), c receives only nonzero messages from neighbours $Y \in N(c) \setminus \{X\}$. If $Y \in N_+(c)$, then $Y \to c < 0$. Hence there is some $b \in N_-(Y)$ such that $b \to Y = 1$. b is closer to a leaf than c in H_{bY}, so by induction, b is violated in H_{bY}. Thus to satisfy b, Y must be set to false, which does not satisfy clause c. A symmetric argument follows if $Y \in N_-(c)$, so it follows that c must be violated in H_{cX}, validating Claim 1.

Then suppose that there is some $c_i = 1$. By definition, there must be a clause $c \in N_+(X_i)$ and clause $b \in N_-(X_i)$ such that $c \to X_i = b \to X_i = 1$. By Claim 1, c is violated in H_{cX_i} and b is violated in H_{bX_i}, so it is impossible to satisfy both clauses.

Proving the converse: It suffices to show that if WP($\mathcal{F}$) returns no contradictory messages, then $\mathcal{F}$ is satisfiable. Firstly, the lack of contradictory messages is equivalent to stating that no variable X receives warnings from clauses containing X and clauses containing $\neg X$. Using induction on $|X|$, the base case is trivial. Consider a non-leaf variable X_1 neighbouring clause c. Let the subgraph H have $V(H) = V(H_{cX_1}) \cup \{X_1\}$ and $E(H) = V(H_{cX_1}) \cup \{cX_1\}$. Note that the messages sent from the leaves of H towards the root X_1 are the same messages sent in the original graph G, so these messages must satisfy the condition of no contradictory messages. However, messages sent from X_1 towards the leaves of H might differ from their values in G. Thus it remains to verify that there are no contradictory messages in the subtree H. If this is true, applying induction, H is satisfiable. The same reasoning applies for each neighbour of X_1, so each such subtree is satisfiable. Since there are no contradictory messages in G, there is no pair of clauses b, c such that b needs $X_1 = 1$ and c needs $X_1 = 0$ by Claim 1. Subsequently, $\mathcal{F}$ must be satisfiable.

Claim 2 If there are no contradictory messages in G, then no variables in H have

contradictory messages in H.

Denote messages in H as $u \to_H v$ and messages in G as $u \to v$. Prove this Claim by induction in the distance $2d$ of variable Y to X_1. For the base case, it is obvious X_1 has no contradictory message since it only has one neighbour in H. For $d = 1$, because X_1 is a leaf in H, $X_1 \to_H c = c \to_H Y = 0$. The other incoming messages $b \to_H Y = b \to Y$, so there are no contradictory messages for Y. Suppose the claim holds for distance $2(d-1)$, and consider a variable Y of distance $2d$ from X_1. Consider the parent b of Y in the subtree rooted at X_1, and the parent Z of b.

If $Y \to b = 0$, then the messages from the children of Y to Y are all zero, so the incoming message $b \to_H Y$ is the only possible nonzero incoming message for Y, so Y has no contradictory message. If $Y \to b > 0$ and $Y \in N_+(b)$ (or $Y \to b < 0$ and $Y \in N_-(b)$), then the message $b \to_H Y$ will obviously not yield a contradiction. Lastly, if $Y \to b > 0$ and $Y \in N_-(b)$ (or its symmetric case): If $b \in N_+(Z)$ (or symmetrically $b \in N_-(Z)$), then $Z \to_H b \geq 0$ because by induction, Z must only receive non-zero messages from clauses containing the non-negated Z. Consequently, $b \to_H Y = 0$, affirming that Y has no contradictory message. Hence Claim 2 holds. ■

Theorem 6.2.2 *WID is sound. That is, WID($\mathcal{F}$) returns a contradiction or does not converge if $\mathcal{F}$ is unsatisfiable.*

Proof This is trivial because when a provided with a full assignment, the algorithm checks if this assignment satisfies $\mathcal{F}$. Thus if $\mathcal{F}$ is unsatisfiable and WID produces a full assignment τ for $\mathcal{F}$, WID will discover that $\mathcal{F}$ is false and returns $\mathcal{F}$ unsatisfiable. It also seems plausible that checking a full assignment is unnecessary, i.e. if WID generates a full assignment for $\mathcal{F}$, then $\mathcal{F}$ is satisfiable. ■

These results are not particularly strong considering that most interesting formulas contain cycles. For instance, even in the case of graphs with a single cycle, WP, and thus WID, can fail. Also, while there is no guarantee of convergence for unsatisfiable formulas with cycles, this simple result does provide a starting point. Why are cycles particularly difficult to handle? The proof for trees uses the fact that a message depends only on messages entering the source that do not involve the destination. In other words, message $u \to v$ depends only on messages "behind" u in the same direction. But for a message in a cycle, at the very least it will depend on all messages in the same direction from every preceding iteration. In the worst case, all messages could potentially depend on all messages, hence injecting a major complication to any potential analysis. Even the simple case of a single cycle demonstrates that WID is not complete.

Example *With probability* 3/16*, WID($\mathcal{F}$) fails if $\mathcal{F}$ is a single cycle with girth 4 with randomly chosen labels. (note that $\mathcal{F}$ is always satisfiable)*

Case 1: Variable x_1 and variable x_2 appear as the same literal in both clauses. Then no clause sends a warning and the subsequent random assignment will always lead to an appropriate satisfying assignment.

Case 2: Only one of the variables (say x_1) appear as the same literal in both clauses. Again, upon convergence, no clause sends a warning because two clause messages must be zero. However, if the assignment to variable x_1 does not satisfy either clause, then $F = x_2 \wedge \neg x_2$ so the second run of WP will reject $\mathcal{F}$, thus failing. So this case has a probability 1/4 of failing.

Case 3: Neither variable appears as the same literal in both clauses. This time, the only possibility of failure is in the order of updates. In particular, failure occurs if a variable receives contradicatory messages upon convergence. Some work shows

that this case also has a probability 1/4 of failing.

Overall, Case 2 and 3 occur 3/4 of the time, so probability of failure is 3/16. □

Example *With probability* $< 3/16$*, WID($\mathcal{F}$) fails if $\mathcal{F}$ is a single cycle with girth 6 with randomly chosen labels.*

There are four cases: either each variable occurs as the same literal, exactly two variables occur as the same literal both times, exactly one variable occurs as the same literal twice or no variable occurs as the same literal twice.

In the first two cases, there is no chance of failure. Case one has no assignments that will cause the remaining formula to be false and no chance of contradictory messages. The second case also has no bad assignment. Moreover, the one variable that can have contradictory messages cannot because the other variables will "dominate" it. The third case has only one chance of failure and that is the bad assignment of the variable occuring as the same literal, with the literal set to false. This has probability 1/6 of occurring. Lastly, the final case will not always fail, and so overall, considering the random assignment of labels, the probability of failure is strictly less than 3/16. □

This tends to suggest that WID will perform better as the cycle gets larger, although it is difficult to be certain with just two examples. A potential approach for the single-cycle resembles Weiss' or Aji's proof for belief propagation on a single cycle [AHM98, Wei01]. However, even an analysis of the general case for the relatively simplistic warning propagation algorithm escapes current efforts. Evidently, the problem rests in how to analyse a message-passing algorithm when messages often depend on a host of other messages from various iterations in a complicated manner.

6.3 Belief propagation

The naïve WP algorithm is essentially a binary algorithm that does not exploit fully the warnings. Applying belief propagation aims to use probabilities to better model the received warnings. The original belief propagation algorithm used to compute marginals is the starting point for the SAT variant of belief propagation.

6.3.1 Message definitions

The original representation used for solving marginals can be rewritten in a convenient form for solving satisfiability that bears a close resemblance to survey propagation. In fact, just as belief propagation is a refined version of warning propagation, survey propagation appears to be a refined version of belief propagation. Define

$$X_i \xrightarrow{S} f_j = \prod_{f_k \in N_j^s(X_i)} (1 - (f_k \to X_i)), \quad X_i \xrightarrow{U} f_j = \prod_{f_k \in N_j^u(X_i)} (1 - (f_k \to X_i)),$$

where if l is the literal of X_i in f_j, $N_j^s(X_i)$ are the clauses containing the literal l excluding f_j, and $N_j^u(X_i)$ are the clauses containing the literal $\neg l$.

$$f_j \to X_i = \prod_{X_k \in N(f_j) \setminus \{X_i\}} X_k \to f_j, \tag{6.6}$$

$$X_i \to f_j = \frac{X_i \xrightarrow{S} f_j}{(X_i \xrightarrow{U} f_j) + (X_i \xrightarrow{S} f_j)}. \tag{6.7}$$

In SAT, the overall intention of the messages is to determine the best assignment for a variable. Thus the clause-to-variable message represents the probability that the clause f_j requires the literal of X_i in f_j to be true given an assignment to the other literals in f_j. Equivalently, this is the probability f_j is false without this literal. Consequently, each variable message must represent the probability that an assignment of X_i violates f_j. Thus the definition of $f_j \to X_i$ assumes the independence of

the neighbouring variables of f_j. Curiously, the fundamental ideas are analogous to those applied for warning propagation, but the range of values is much finer than in warning propagation.

Note that the variable message definition should be proportional to the product of the probability that clauses f_k with the same literal of X_i as f_j are true without this literal. Assuming independence, this equates to the probability that every such f_k except f_j is true excluding the literal of X_i. This does not quite correspond to the previous interpretation of the variable message, where it is the probability that X_i violates f_j. However, assume that if all clauses $f_k \in N_j^s(X_i)$ are true without the literal of X_i, then not all clauses $f_k \in N_j^u(X_i)$ are true without the literal of X_i. Then in fact, the two interpretations coincide.

Critically, these definitions make some bold independence assumptions because of the products in each message. Under certain circumstances where dependence is strong, it seems likely that belief propagation, and similarly survey propagation, is prone to to failure. Also note that the assumption matching the two conflicting variable message definitions is typically false, and is dealt with in survey propagation.

6.3.2 Algorithm

BP for SAT is nearly identical to WP. Simply change the initialisation to be a random assignment in $[0, 1]$, use the above message definitions, and instead of exact convergence, define convergence to be within some small tolerance. Upon the completion of the propagation phase, one must compute the beliefs, which is sometimes called the decimation phase because this portion of the algorithm assigns the variables of the formula [BMZ03]. To compute the beliefs B_i, define

$$S_i = \prod_{f_k \in N_+(X_i)} (1 - (f_k \to X_i)), \quad U_i = \prod_{f_k \in N_-(X_i)} (1 - (f_k \to X_i)), \quad B_i = \frac{U_i}{U_i + S_i}.$$

There is nothing complex about this definition, as it closely resembles the variable message definition, and determines the preferred assignment of variables, if they have one.

6.3.3 Correctness

It suffices to show that SAT-BP is a special case of belief propagation for marginals, in which case any property of BP follows for SAT-BP. Recall the original definitions:

$$f_j \to X_i(x_i) = \sum_{\mathbf{x_j}: X_i = x_i} f_j(\mathbf{x_j}) \prod_{X_k \in N(f_j)\setminus\{X_i\}} X_k \to f_j(x_k), \tag{6.8}$$

$$X_i \to f_j(x_i) = \prod_{f_k \in N(X_i)\setminus\{f_j\}} f_k \to X_i(x_i). \tag{6.9}$$

6.4 Survey propagation

The original description of survey propagation is rather in-depth and extensive when, in fact, the basic motivation behind the algorithm is not particularly difficult to explain [Urq03]. The following explanation follows a similar route in the interpretation of survey propagation, using its close relation to belief propagation to draw some sort of intuition.

6.4.1 Message definitions

Recall the definitions for $X_i \xrightarrow{U} f_j$ and $X_i \xrightarrow{S} f_j$, and define the following:

$$X_i \xrightarrow{0} f_j = (X_i \xrightarrow{U} f_j)(X_i \xrightarrow{S} f_j).$$

$$f_j \to X_i = \prod_{X_k \in N(f_j)\setminus\{X_i\}} X_k \to f_j, \tag{6.10}$$

$$X_i \to f_j = \frac{(X_i \xrightarrow{S} f_j) - (X_i \xrightarrow{0} f_j)}{(X_i \xrightarrow{U} f_j) + (X_i \xrightarrow{S} f_j) - (X_i \xrightarrow{0} f_j)}. \tag{6.11}$$

In fact, these messages are nearly identical to their belief propagation counterparts. The generalisation in survey propagation simply sheds the previous assumption that if all clauses $f_k \in N_j^s(X_i)$ are true without the literal of X_i, then not all clauses $f_k \in N_j^u(X_i)$ are true without the literal of X_i. It is entirely possible that all clauses do not rely on X_i at all for satisfaction. Thus subtracting this possibility yields a better expression of the two interpretations of the variable message discussed in the section on belief propagation. Specifically, the variable message is the probability that X_i violates f_j and $X_i \xrightarrow{S} f_j$ means the probability that all clauses in $f_k \in N_j^s(X_i)$ are true without the literal of X_i. The additional term $X_i \xrightarrow{0} f_j$ corresponds to the probability that all clauses in $f_k \in N(X_i)$ except f_j are true without the literal of X_i. Hence $(X_i \xrightarrow{S} f_j) - (X_i \xrightarrow{0} f_j)$ refers to precisely the probability that all clauses in $f_k \in N_j^s(X_i)$ are true without the literal of X_i *and* some clause in $f_k \in N_j^u(X_i)$ is false without the said literal.

6.4.2 Algorithm

The SP algorithm is the same as BP, except in the update. If these messages converge, first check if all of the clause messages are zero. This situation implies that survey propagation has no idea of how to proceed with the remainder of the formula, and turns over the rest of the problem to a more traditional and well-established solver. Otherwise, compute the three *biases* of a variable X_i, which globally determine the probability of X_i being true (positive bias B_i^+), false (negative bias B_i^-), or uncertain (B_i^0). Just as in all message-passing algorithms, define these biases in the same manner as the above messages except over all neighbouring factors. Note that three computations necessary because of the addition of the third possibility that belief

propagation excluded by assumption.

$$B_i^+ = \frac{U_i - U_i S_i}{U_i + S_i - U_i S_i}, \quad B_i^- = \frac{S_i - U_i S_i}{U_i + S_i - U_i S_i}, \quad B_i^0 = \frac{U_i S_i}{U_i + S_i - U_i S_i}.$$

The final decision of variable assignment is usually taken to be at least one variable with high inclination towards assignment, i.e. $|B_i^+ - B_i^-|$ is large. The algorithm then begins anew with the simplified formula.

Appendix A

Lagrange multipliers

This section deals with a fundamental technique of non-linear constrained optimisation called the Lagrange multiplier method utilised in Theorem 5.1.5. There are a plethora of references for multivariate calculus and optimisation theory that discuss the Lagrange multiplier method, and a couple of these form the basis of this overview [CZ01, MT96, Spi65]. First, a brief summary reviews the critical definitions of multivariate differentiation and extrema. Subsequently, a discussion of the Lagrange multiplier method follows in two parts. The simpler case is for problems with only equality constraints, and the harder case is for problems that also have inequality constraints.

A.1 Definitions

Let $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ be a vector-valued function, often denoted as $\boldsymbol{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$, where $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \ldots, m$. The concept of a neighbourhood is central to differentiation, and whereas the notion of a neighbourhood is clear in $\mathbb{R}^2$, its immediate generalisation to $\mathbb{R}^n$ requires a few more definitions to be understandable.

Definition For $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, the *Euclidean norm* is $\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{n} x_i^2}$. An *open set* U is a subset of $\mathbb{R}^n$ if for any point $\mathbf{x} \in U$, there exists $\epsilon > 0$ such that for any point $\mathbf{y} \in \mathbb{R}^n$ with Euclidean distance $\|\mathbf{x} - \mathbf{y}\| < \epsilon$, $\mathbf{y} \in U$. Thus intuitively, no element of an open set U is a boundary point. A *neighbourhood* of $\mathbf{x} \in \mathbb{R}^n$ is an open set U containing the point $\mathbf{x}$.

For instance, an obvious example of an open set is the open ball of radius r and centre $\mathbf{x_0}$, which is defined as the set of points $D_r(\mathbf{x_0}) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x_0}\| < r\}$. The following definition of a limit is a variant of the traditional $\epsilon - \delta$ definition.

Definition For a vector-valued function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$, and $\mathbf{a} \in \mathbb{R}^n$, the *limit of a function* $\boldsymbol{f}$ *at a point* $\mathbf{a}$ exists, or equivalently f approaches $\mathbf{b}$ at $\mathbf{a}$, denoted

$$\lim_{\mathbf{x} \to \mathbf{a}} \boldsymbol{f}(\mathbf{x}) = \mathbf{b}$$

iff for all neighbourhoods N of $\mathbf{b} \in R^m$, there exists a neighbourhood U of $\mathbf{a}$ such that if $\mathbf{x} \neq \mathbf{a}$, and $\mathbf{x}$ is in U, then $\boldsymbol{f}(\mathbf{x})$ is in N. Moreover, $\boldsymbol{f}$ is *continuous at* $\mathbf{a}$ if $\lim_{\mathbf{x} \to \mathbf{a}} \boldsymbol{f}(\mathbf{x}) = \boldsymbol{f}(\mathbf{a})$, and $\boldsymbol{f}$ is a *continuous function* if it is continuous at every point $\mathbf{a} \in \mathbb{R}^n$.

Definition For a function $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, $i \in \{1, \ldots, n\}$, and $\mathbf{e_i}$ the ith standard basis vector, the *partial derivative* of f with respect to the ith variable exists and is the real-valued function of n variables

$$\frac{\partial f}{\partial x_i}(\mathbf{x}) = \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{e_i}) - f(\mathbf{x})}{h},$$

if the limit exists.

The partial derivative is essentially the same as computing the single-variable derivative of a function, because its definition assumes that only one variable is not

fixed. As such, it only reveals part of the overall picture of the derivative of a function, and one must use all these partials to determine the differentiability of a function of many variables.

Definition For a function $f : \mathbb{R}^n \to \mathbb{R}$, the *gradient* of f at $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is $\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right)$.

Definition For a function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$, $\boldsymbol{f}$ is *differentiable* at $\mathbf{x} \in \mathbb{R}^n$ if the partial derivatives of $\boldsymbol{f}$ exist at $\mathbf{x}$ and if

$$\lim_{\mathbf{h}\to 0} \frac{\|\boldsymbol{f}(\mathbf{x}+\mathbf{h}) - \boldsymbol{f}(\mathbf{x}) - D\boldsymbol{f}(\mathbf{h})\|}{\|\mathbf{h}\|} = 0.$$

$D\boldsymbol{f}(\mathbf{x})$ is the *derivative* of $\boldsymbol{f}$ at $\mathbf{x}$, and is the $m \times n$ Jacobian matrix of partials $\frac{\partial f_i}{\partial x_j}(\mathbf{x})$. Note that a row i of the Jacobian matrix is ∇f_i, so if $m = 1$, $Df(\mathbf{x}) = \nabla f(\mathbf{x})$.

This definition is an extension of the single-variable derivative, and complete details regarding the intricacies of multivariable calculus can be found in any text on the subject [Spi65].

Definition A function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ is a C^1 *function* if all of its partials exist and they are all continuous. Moreover, all C^1 functions are differentiable [MT96, pp. 159] [Spi65, p. 31].

Definition A *path* $\boldsymbol{c}$ in $S \subseteq \mathbb{R}^n$ is a continuous map $\boldsymbol{c} : (a, b) \to S$, and a *curve* C on a surface S is the set of points $C = \{\boldsymbol{c}(t) \in S : t \in (a, b)\}$. If $\boldsymbol{c}$ is differentiable, the derivative $\boldsymbol{c}'(t)$, or velocity, is a column vector *tangent* to the path $\boldsymbol{c}(t)$, and the curve C traced out by $\boldsymbol{c}$ if $\boldsymbol{c}'(t) \neq 0$.

Definition For a C^1 function $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$, consider a surface $S = \{\mathbf{x} \in \mathbb{R}^n : \boldsymbol{h}(\mathbf{x}) = \mathbf{0}\}$. Then the *tangent space* at a point $\mathbf{x_0}$ on S is the set $T(\mathbf{x_0}) = \{\mathbf{y} \in \mathbb{R}^n : D\boldsymbol{h}(\mathbf{x_0})\mathbf{y} = \mathbf{0}\}$.

Why is this a reasonable definition for the tangent space for such a surface S? The basic intuition is that the tangent space at $\mathbf{x_0}$ on the surface S should be the set of all vectors tangent to the surface at $\mathbf{x_0}$. Not surprisingly, the tangent space at $\mathbf{x_0}$ is precisely the set of all vectors tangent to paths in S passing through $\mathbf{x_0}$, whenever the derivatives ∇h_i are linearly independent at $\mathbf{x_0}$, given by the following theorem.

Theorem A.1.1 *Let* $\mathbf{x_0}$ *be a point in* $S = \{\mathbf{x} \in \mathbb{R}^n : \boldsymbol{h}(\mathbf{x}) = \mathbf{0}\}$ *such that the derivatives* $\nabla h_i(\mathbf{x_0})$ *are linearly independent, and* $T(\mathbf{x_0}$ *be the tangent space on* S*. Then* $\mathbf{y} \in T(\mathbf{x_0})$ *iff there exists a differentiable path in* S *passing through* $\mathbf{x_0}$ *with derivative* $\mathbf{y}$ *at* $\mathbf{x_0}$*.*

Proof The if direction of the theorem is straightforward. For a path $\boldsymbol{c}(t)$ in S, where $\boldsymbol{c}(t_0) = \mathbf{x_0}$, $\boldsymbol{c}'(t_0)$ is a tangent vector to S at $\mathbf{x_0}$. Furthermore, $\frac{d}{dt}h_i(\boldsymbol{c}(t)) = 0$ for any $i = 1, \ldots, n$, and by the chain rule, the gradient of each h_i is orthogonal to the tangent space:

$$0 = \frac{d}{dt}h_i(\boldsymbol{c}(t_0)) = \nabla h_i(\mathbf{x_0}) \cdot \boldsymbol{c}'(t_0).$$

Consequently, a vector tangent to a curve in S at $\mathbf{x_0}$ is tangent to the surface S at $\mathbf{x_0}$. The converse requires the rather lengthy implicit function theorem, which uses the assumption of linear independence [MT96, pp. 226-230], [Spi65, pp. 35-39]. ■

Definition If $f : \mathbb{R}^n \to \mathbb{R}$ is a given function, then the point $\mathbf{x_0} \in \mathbb{R}^n$ is a *local minimum* of f if there is a neighbourhood V of $\mathbf{x_0}$ such that for all points $\mathbf{x} \in V$, $f(\mathbf{x_0}) \leq f(\mathbf{x})$. Similarly, $\mathbf{x_0}$ is a *local maximum* if there is a neighbourhood V of $\mathbf{x_0}$ such that for all points $\mathbf{x} \in V$, $f(\mathbf{x}) \leq f(\mathbf{x_0})$. A point is a *local extremum* if it is either a local minimum or maximum. A point $\mathbf{x_0}$ is a *critical point* of f if either f is not differentiable at $\mathbf{x_0}$, or if $\nabla f(\mathbf{x}) = 0$. A *stationary point* is a critical point where $\nabla f(\mathbf{x}) = 0$. Finally, a *saddle point* is a critical point that is not a local extremum.

For instance, consider the simple case of a differentiable function $f : \mathbb{R} \to \mathbb{R}$ of a single variable. The above definitions clearly conform with the traditional single-variable calculus definitions for extrema. Moreover, it is well-established that to find these extrema, one simply needs to find the stationary points, that is find all $x \in \mathbb{R}$ such that $f^{'}(x) = 0$, and determine which are saddle points, local minima, and maxima using higher-order derivative tests [Spi94, 185-199, 410-412].

The multivariate unconstrained case follows in an analogous fashion. To find the local extrema of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, it suffices to find all $\mathbf{x} \in \mathbb{R}^n$ such that $\nabla f(\mathbf{x}) = 0$. Finally, one uses a generalisation of the higher-order derivative tests based on the Hessian to distinguish the type of extrema [MT96, pp. 190-204].

A.2 Extrema under equality constraints

Introducing constraints into the equation complicates matters. The problem can be formulated in the following manner:

Find the local extrema of $f(\mathbf{x})$, where $f : \mathbb{R}^n \to \mathbb{R}$
subject to *equality constraints* $\boldsymbol{h}(\mathbf{x}) = \mathbf{0}$, where $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$ and $m \leq n$
and *inequality constraints* $\boldsymbol{g}(\mathbf{x}) \leq \mathbf{0}$, where $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^p$

The motivation behind this method is to consolidate all of the constraints and the main function into a single function called the Lagrangian which can be used to compute the stationary points of the original function under the given constraints. Initially, assume that there are no inequality constraints, so $p = 0$. Essentially, the following theorem shows that solving for the stationary points of the Lagrangian function will find potential extrema of the original function under constraints [CZ01, MT96].

Definition The *Lagrangian function* $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ of a C^1 function $f : \mathbb{R}^n \to \mathbb{R}$ subject to the C^1 function of constraints $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$ is defined with *Lagrangian multipliers* $\boldsymbol{\lambda} \in \mathbb{R}^m$ as

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i h_i(\mathbf{x_0}).$$

The derivative of L at $(\mathbf{x}, \boldsymbol{\lambda})$ is

$$\nabla L(\mathbf{x}, \boldsymbol{\lambda}) = \left(\nabla f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \nabla h_i(\mathbf{x}), \boldsymbol{h_i}(\mathbf{x}) \right).$$

Theorem A.2.1 *(Lagrange) Let $f : \mathbb{R}^n \to \mathbb{R}$ and $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m, m \leq n$ be C^1 functions, where f is subject to $\boldsymbol{h}(\mathbf{x}) = \mathbf{0}$, namely f is restricted to the surface $S = \{\mathbf{x} \in \mathbb{R}^n : \boldsymbol{h}(\mathbf{x}) = \mathbf{0}\}$. If $\mathbf{x_0} \in \mathbb{R}^n$ is a local minimum, or maximum, of f restricted to S and the derivatives $\nabla h_1(\mathbf{x_0}), \ldots, \nabla h_m(\mathbf{x_0})$ are linearly independent, then there exists Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^m$ such that the derivative of the Lagrangian is zero at $\mathbf{x_0}$:*

$$\nabla L(\mathbf{x_0}, \boldsymbol{\lambda}) = \mathbf{0}.$$

Proof Rewriting the summation yields $D\boldsymbol{h}(\mathbf{x_0})^T \boldsymbol{\lambda}$, so show that $\nabla f(\mathbf{x_0}) = -D\boldsymbol{h}(\mathbf{x_0})^T \boldsymbol{\lambda}$ for some $\boldsymbol{\lambda} \in \mathbb{R}^m$. Let $\mathbf{y} \in T(\mathbf{x_0})$. Observe that for any points $\mathbf{z} \in \mathbb{R}^m$ and $\mathbf{x} = D\boldsymbol{h}(\mathbf{x_0})^T \mathbf{z} \in \mathbb{R}^n$, it follows that $\mathbf{x}^T \mathbf{y} = 0$, so $\mathbf{x} \in T(\mathbf{x_0})^\perp$. Hence it suffices to show that $\nabla f(\mathbf{x_0})^T \mathbf{y} = 0$. By Theorem A.1.1, a tangent vector $\mathbf{y} \in T(\mathbf{x_0})$ is also tangent to some path $\boldsymbol{c}(t), t \in (a, b)$ in S at $\mathbf{x_0}$. Thus, there is a $t_0 \in (a, b)$ such that $\boldsymbol{c}(t_0) = \mathbf{x_0}$, and $\boldsymbol{c}'(t_0) = \mathbf{y}$. Since $\mathbf{x_0}$ is a local minimum of f restricted to S, $f(\boldsymbol{c}(t))$ has a local minimum at $t = t_0$, and by the chain rule, $0 = \frac{df}{dt}(\boldsymbol{c}(t_0)) = \nabla f(\mathbf{x_0}) \cdot \boldsymbol{c}'(t_0)$.

■

Hence expanding out the derivative ∇L, one finds the stationary points of f subject to constraints $\boldsymbol{h}$ using the following $m+n$ equations.

$$\nabla L(\mathbf{x}, \boldsymbol{\lambda}) = \left(\frac{\partial L}{\partial x_1}(\mathbf{x}, \boldsymbol{\lambda}), \ldots, \frac{\partial L}{\partial x_n}(\mathbf{x}, \boldsymbol{\lambda}), \frac{\partial L}{\partial \lambda_i}(\mathbf{x}, \boldsymbol{\lambda}), \ldots, \frac{\partial L}{\partial \lambda_m}(\mathbf{x}, \boldsymbol{\lambda}) \right) = \mathbf{0}.$$

Clearly, this method will find all of the stationary points of the function f. However as in the unconstrained case, distinguishing the type of stationary points requires higher-order derivative tests based on the Hessian, which is beyond the scope of detail required herein. Moreover, it is covered extensively in the pertinent introductory texts [CZ01, 383-387], [MT96, pp. 218-223].

A.3 Extrema under inequality constraints

When there are also inequality constraints, Lagrange multipliers require the services of Kuhn-Tucker conditions [CZ01, KT50]. As usual, some definitions are in order.

Definition An inequality constraint $g_j(\mathbf{x}) \leq 0$ is *active* at $\mathbf{x_0}$ if $g_j(\mathbf{x_0}) = 0$, and *inactive* if $g_j(\mathbf{x_0}) < 0$. Moreover, equality constraints are always active. Finally, the index set of active inequality constraints at $\mathbf{x_0}$ is $J(\mathbf{x_0}) = \{j : g_j(\mathbf{x_0}) = 0\}$.

Definition A point $\mathbf{x_0}$ is an *interior point* if all inequality constraints are inactive, and $\mathbf{x_0}$ is an *edge point* if there is an active constraint. A point $\mathbf{x_0}$ is a *local interior extremum* if it is an interior point and a local extremum such that the neighbourhood of $\mathbf{x_0}$ satisfies the constraints. Similarly, a point is an *interior stationary point* if it is an interior point and a stationary point such that the neighbourhood of $\mathbf{x_0}$ satisfies the constraints. An *edge stationary point* is an edge point and a stationary point such that the neighbourhood of $\mathbf{x_0}$ satisfies the constraints, and maintains all active inequality constraints at $\mathbf{x_0}$.

With $p \neq 0$, the definition for a Lagrangian must be generalised.

Definition Given the *Lagrange multipliers* $\boldsymbol{\lambda} \in \mathbb{R}^m, \boldsymbol{\pi} \in \mathbb{R}^p$, the *Lagrangian function* $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ of a C^1 function $f : \mathbb{R}^n \to \mathbb{R}$ subject to C^1 functions of constraints $\boldsymbol{h} : \mathbb{R}^m \to \mathbb{R}$ and $\boldsymbol{g} : \mathbb{R}^p \to \mathbb{R}$ is

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\pi}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i h_i(\mathbf{x}) + \sum_{j=1}^{p} \pi_j g_j(\mathbf{x}).$$

The derivative of L at $(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\pi})$ is

$$\nabla L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\pi}) = \left(\nabla f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \nabla h_i(\mathbf{x}) + \sum_{j=1}^{p} \pi_j \nabla g_j(\mathbf{x}), \boldsymbol{h_i}(\mathbf{x}), \boldsymbol{g_j}(\mathbf{x}) \right).$$

The Karush-Kuhn-Tucker theorem generalises the Lagrange method to handle inequality constraints.

Theorem A.3.1 *(Karush-Kuhn-Tucker) Let* $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^p$, *and* $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m, m \leq n$ *be* C^1 *functions, where* f *is subject to* $\boldsymbol{h}(\mathbf{x}) = \mathbf{0}$ *and* $\boldsymbol{g}(\mathbf{x}) \leq \mathbf{0}$, *namely* f *is restricted to the surface* $S = \{\mathbf{x} \in \mathbb{R}^n : \boldsymbol{h}(\mathbf{x}) = \mathbf{0}, \boldsymbol{g}(\mathbf{x}) \leq \mathbf{0}\}$. *If* $\mathbf{x_0} \in \mathbb{R}^n$ *is a local minimum of* f *restricted to* S *and all derivatives of active constraints (*$\nabla h_i(\mathbf{x_0})$ *for* $i = 1, \ldots, n$ *and* $\nabla g_j(\mathbf{x_0})$ *for* $j \in J(\mathbf{x_0})$*) are linearly independent at* $\mathbf{x_0}$, *then there exists Lagrange multipliers* $\boldsymbol{\lambda} \in \mathbb{R}^m, \boldsymbol{\pi} \in \mathbb{R}^p$ *such that the following conditions hold:*

1. $\boldsymbol{\pi} \geq 0$,
2. $\nabla f(\mathbf{x_0}) + \sum_{i=1}^{m} \lambda_i \nabla h_i(\mathbf{x_0}) + \sum_{j=1}^{p} \pi_j \nabla g_j(\mathbf{x_0}) = \mathbf{0}$, *and*
3. $\boldsymbol{\pi} \cdot \boldsymbol{g}(\mathbf{x_0}) = 0$.

Proof ([CZ01, pp. 399-401]) Firstly, since $\mathbf{x_0}$ is a local minimum of f restricted to S, it is also a local minimum of f restricted to the surface restricted only to the

active constraints, defined as $\{\mathbf{x} : \boldsymbol{h}(\mathbf{x}) = 0, g_j(\mathbf{x}) = 0, j \in J(\mathbf{x_0})\}$. Thus Theorem A.2.1 applies, producing Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^m, \boldsymbol{\pi} \in \mathbb{R}^p$ such that

$$\mathbf{0} = \nabla f(\mathbf{x_0}) + \sum_{i=1}^{m} \lambda_i \nabla h_i(\mathbf{x_0}) + \sum_{j=1}^{p} \pi_j \nabla g_j(\mathbf{x_0}),$$

where for all inactive constraints, $\pi_j = 0$, thus eliminating the inactive constraints, so condition 2 holds. Condition 3 is also an immediate consequences of this fact:

$$\boldsymbol{\pi} \cdot \boldsymbol{g}(\mathbf{x_0}) = \sum_{j=1}^{p} \pi_j g_j(\mathbf{x_0}) = \sum_{j \in J(\mathbf{x_0})} \pi_j g_j(\mathbf{x_0}) + \sum_{j \notin J(\mathbf{x_0})} \pi_j g_j(\mathbf{x_0}) = 0,$$

because for $j \notin J(\mathbf{x_0}), \pi_j = 0$ and for $j \in J(\mathbf{x_0}), g_j(\mathbf{x_0}) = 0$.

It remains to show that condition 1 holds, that is, for all active inequality constraints $j \in J(\mathbf{x_0})$, $\pi_j \geq 0$. Prove $\pi_j \geq 0$ by contradiction, so assume that there exists a $k \in J(\mathbf{x_0})$ such that $\pi_j < 0$. Let the remaining active constraints at $\mathbf{x_0}$ define the surface $\hat{S}$ and tangent space $\hat{T}(\mathbf{x_0})$ at $\mathbf{x_0}$:

$$\hat{S} = \{\mathbf{x} : \boldsymbol{h}(\mathbf{x}) = 0, g_j(\mathbf{x}) = 0, j \in J(\mathbf{x_0}), j \neq k\}, \text{ and}$$

$$\hat{T}(\mathbf{x_0}) = \{\mathbf{y} : D\boldsymbol{h}(\mathbf{x_0})\mathbf{y} = 0, \nabla g_j(\mathbf{x_0})\mathbf{y} = 0, j \in J(\mathbf{x_0}), j \neq k\}.$$

Then there exists $\mathbf{y} \in \hat{T}(\mathbf{x_0})$ such that $\nabla g_k(\mathbf{x_0})\mathbf{y} < 0$, because for $i = 1, \ldots, m, j \in J(\mathbf{x_0})$, the derivatives $\nabla h_i(\mathbf{x_0}), \nabla g_j(\mathbf{x_0})$ are linearly independent. Then $\nabla L\mathbf{y} = \mathbf{0}$ yields $\nabla f(\mathbf{x_0})\mathbf{y} + \pi_k \nabla g_k(\mathbf{x_0})\mathbf{y} = \mathbf{0}$. Consequently, $\nabla f(\mathbf{x_0})\mathbf{y} < 0$ because $\pi_k \nabla g_k(\mathbf{x_0})\mathbf{y} > 0$.

As stated in the previous theorem, the tangent vector $\mathbf{y} \in \hat{T}(\mathbf{x_0})$ is also tangent to some path $\boldsymbol{c}(t), t \in (a, b)$ in $\hat{S}$ at $\mathbf{x_0}$, so let $\boldsymbol{c}(t_0) = \mathbf{x_0}$ and $\boldsymbol{c}'(t_0) = \mathbf{y}$, where $t_0 \in (a, b)$. Then by the chain rule,

$$\frac{df}{dt}(\boldsymbol{c}(t_0)) = \nabla f(\mathbf{x_0})\mathbf{y} < 0, \text{ and}$$

$$\frac{dg_k}{dt}(\boldsymbol{c}(t_0)) = \nabla g_k(\mathbf{x_0})\mathbf{y} < 0.$$

Hence there exists $\delta > 0$ such that for all $t \in (t_0, t_0+\delta]$, $f(\boldsymbol{c}(t)) < f(\mathbf{x_0})$. Furthermore, there exists $\epsilon > 0$ such that for all $t \in [t_0, t_0 + \epsilon]$, $g_k(\boldsymbol{c}(t)) \leq 0$. Consequently, for all $t \in (t_0, t_0 + \min(\delta, \epsilon)]$, $f(\boldsymbol{c}(t)) < f(\mathbf{x_0})$ and $g_k(\boldsymbol{c}(t)) \leq 0$. Ergo, these points are in $\hat{S}$, contradicting the assumption that $\mathbf{x_0}$ is a local minimum of f restricted to S. ■

The same result holds for a local maximum, except that in this situation, $\boldsymbol{\pi} \leq 0$. Still, the underlying framework is identical, so one can compute the stationary points using the same process. The conditions supply the following equations and inequalities:

- $\nabla f(\mathbf{x_0}) + \sum_{i=1}^{m} \lambda_i \nabla h_i(\mathbf{x_0}) + \sum_{j=1}^{p} \pi_j \nabla g_j(\mathbf{x_0}) = \mathbf{0}$,
- $\boldsymbol{h}(\mathbf{x_0}) = \mathbf{0}$,
- $\boldsymbol{g}(\mathbf{x_0}) \leq \mathbf{0}$,
- $\boldsymbol{\pi} \geq \mathbf{0}$ (or $\boldsymbol{\pi} \leq \mathbf{0}$ for local maxima), and
- $\boldsymbol{\pi} \cdot \boldsymbol{g}(\mathbf{x_0}) = 0$.

Note that since $\boldsymbol{\pi} \geq \mathbf{0}$, or $\boldsymbol{\pi} \leq \mathbf{0}$, the last condition can also be written as $\pi_j g_j(\mathbf{x_0}) = 0$, for each $j = 1, \ldots, p$. Hence points satisfying all of these conditions are stationary points of f restricted to S. Furthermore, all stationary points of f restricted to S are all attainable using this process, i.e. if $\mathbf{x_0}$ is a stationary point of f restricted to S, then it satisfies the above conditions. This is true because $\mathbf{x_0}$ must satisfy the constraints, so the Lagrangian reduces to the original function f, and $\nabla f(\mathbf{x_0}) = 0$

A.4 Linear constraints

An interesting special case of this problem occurs when all of the constraints are linear in $\mathbf{x}$. Clearly, the previous proofs would still apply, and Lagrange multipliers will always exist. However, there is an alternate proof of this special case using only the simple lemma of Farkas, avoiding the more involved implicit function theorem used in the general case. Note that for $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} \geq 0$ means that $x_i \geq 0$ for $i = 1, \ldots, n$.

Lemma A.4.1 *(Farkas) For a matrix $A \in \mathbb{R}^{m \times n}$ and vectors $\mathbf{x} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^n, \boldsymbol{\lambda} \in \mathbb{R}^m$, $A\mathbf{x} \geq 0$ with $\mathbf{b}^T\mathbf{x} < 0$ has a solution $\mathbf{x}$ iff $A^T\boldsymbol{\lambda} = \mathbf{b}$ with $\boldsymbol{\lambda} \geq 0$ has no solution $\boldsymbol{\lambda}$.*

Proof First prove the only if direction. Then suppose that the right side has a solution, i.e. both sides have a solution. Then since $A\mathbf{x} \geq 0$ and $\boldsymbol{\lambda} \geq 0$

$$\boldsymbol{\lambda} \cdot (A\mathbf{x}) \geq 0,$$

but because $A^T\boldsymbol{\lambda} = \mathbf{b}$ and $\mathbf{b}^T\mathbf{x} < 0$,

$$\boldsymbol{\lambda} \cdot (A\mathbf{x}) = \mathbf{b}^T\mathbf{x} < 0,$$

thus a contradiction.

For the if direction, suppose that the left side has no solution, i.e. neither side has a solution. Consider the set $S = \{\mathbf{s} \in \mathbb{R}^n : \mathbf{s} = \boldsymbol{\lambda}^T A, \boldsymbol{\lambda} \geq 0\}$. By assumption, $\mathbf{b} \notin S$, and the key geometric observation is that there is a $\mathbf{x} \in \mathbb{R}^n$ such that for some $\alpha \in \mathbb{R}$,

$$\mathbf{x}^T\mathbf{b} < \alpha < \mathbf{x}^T\mathbf{s},$$

for all $\mathbf{s} \in S$. It follows that $\alpha < 0$ because $\mathbf{0} \in S$, and for all $\boldsymbol{\lambda} \geq 0$,

$$\alpha < \boldsymbol{\lambda} \cdot (A\mathbf{x}).$$

Then suppose for some i, $(A\mathbf{x})_i < 0$. But then setting $\lambda_i = (\alpha - 1)/(A\mathbf{x})_i$ and the remaining $\lambda_j = 0$ produces $\boldsymbol{\lambda} \cdot (A\mathbf{x}) = \alpha - 1$, so $A\mathbf{x} \geq 0$. Finally, $\mathbf{x}^T\mathbf{b} < \alpha < 0$, so $A\mathbf{x} \geq 0$ with $\mathbf{b}^T\mathbf{x} < 0$ has a solution, thus achieving the desired contradiction. ■

The following convenient description is obviously equivalent to Farkas' lemma: if $\mathbf{b}^T\mathbf{x} \geq 0$ for all $\mathbf{x}$ satisfying $A\mathbf{x} \geq 0$, then $A^T\boldsymbol{\lambda} = \mathbf{b}$ with $\boldsymbol{\lambda} \geq 0$ has a solution $\boldsymbol{\lambda}$.

Theorem A.4.2 *If a constrained optimisation problem has only linear constraints, then for a local minimum* $\mathbf{x_0}$ *of* f *restricted to* S*, there exist Lagrange multipliers* $\boldsymbol{\lambda} \in \mathbb{R}^m, \boldsymbol{\pi} \in \mathbb{R}^p$ *such that if there are only equality constraints, the Lagrange condition in Theorem A.2.1 hold at* $\mathbf{x_0}$ *and if there are also inequality constraints, the Karush-Kuhn-Tucker conditions in Theorem A.3.1 hold at* $\mathbf{x_0}$*.*

Proof (equality) First prove the equality constraint case. For $\mathbf{b} \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}$, and letting the ith column of A be A_i, the m equality constraints can be written as

$$\boldsymbol{h}(\mathbf{x}) = A\mathbf{x} - \mathbf{b} = 0 \text{ (so } \nabla h_i(\mathbf{x}) = A_i).$$

As in the Lagrange theorem, assume linear independence of the constraints. Since $\mathbf{x_0}$ is a local minimum, for sufficiently small $c > 0$, there exists a vector $\mathbf{y}$ such that $\mathbf{x_0} + c\mathbf{y}$ is also on f restricted to S, and by the first-order Taylor expansion [MT96, p. 183],

$$0 = h_i(\mathbf{x_0} + c\mathbf{y}) = h_i(\mathbf{x_0}) + c\nabla h_i(\mathbf{x_0}) \cdot \mathbf{y} + o(c),$$

implying that $\nabla h_i(\mathbf{x_0}) \cdot \mathbf{y} = 0$, so $A\mathbf{y} = \mathbf{0}$. Similarly,

$$f(\mathbf{x_0} + c\mathbf{y}) = f(\mathbf{x_0}) + c\nabla f(\mathbf{x_0}) \cdot \mathbf{y} + o(c),$$

and $f(\mathbf{x_0}) \leq f(\mathbf{x_0} + c\mathbf{y})$, so $\nabla f(\mathbf{x_0}) \cdot \mathbf{y} \geq 0$.

Then $A\mathbf{y} \geq \mathbf{0}$ and $-A\mathbf{y} \geq \mathbf{0}$, so applying Farkas' lemma twice results in the following equations: For nonnegative $\boldsymbol{\lambda^0}, \boldsymbol{\lambda^1} \in \mathbb{R}^m$ such that

$$\nabla f(\mathbf{x_0}) = \sum_{i=1}^{m} \lambda_i^0 \nabla h_i(\mathbf{x_0}) \text{ and } \nabla f(\mathbf{x_0}) = -\sum_{i=1}^{m} \lambda_i^1 \nabla h_i(\mathbf{x_0}).$$

Thus it follows that there exist Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^m$ such that $\nabla f(\mathbf{x_0}) = \sum_{i=1}^{m} \lambda_i \nabla h_i(\mathbf{x_0})$, and the equality constraints also hold at $\mathbf{x_0}$ by definition. ■

Proof (inequality) When there are only inequality constraints, the proof follows similarly. For $B \in \mathbb{R}^{p \times n}, \mathbf{b} \in R^p$, the p inequality constraints can be written as

$$\boldsymbol{g}(\mathbf{x}) = B\mathbf{x}) + \mathbf{b} \leq 0 \text{ (so } \nabla g_j(\mathbf{x}) = B_j), \text{ where } B_j \text{ is the } j\text{th column of } B.$$

As in the KKT theorem, assume linear independence of the active constraints, which implies that the vectors $\mathbf{n_j}$ with $j \in J(\mathbf{x_0})$ are linear independent, and assume that there is at least one active constraint.

Since $\mathbf{x_0}$ is a local minimum, for sufficiently small $c > 0$, there exists a vector $\mathbf{y}$ such that $\mathbf{x_0} + c\mathbf{y}$ is also on f restricted to S, and by the first-order Taylor expansion, where $j \in J(\mathbf{x_0})$,

$$0 \geq g_j(\mathbf{x_0} + c\mathbf{y}) = g_j(\mathbf{x_0}) + c\nabla g_j(\mathbf{x_0}) \cdot \mathbf{y} + o(c),$$

implying that $-\nabla g_j(\mathbf{x_0}) \cdot \mathbf{y} \geq 0$, so $A\mathbf{y} \geq \mathbf{0}$, where $A = -B$. Similarly,

$$f(\mathbf{x_0} + c\mathbf{y}) = f(\mathbf{x_0}) + c\nabla f(\mathbf{x_0}) \cdot \mathbf{y} + o(c),$$

and $f(\mathbf{x_0}) \leq f(\mathbf{x_0} + c\mathbf{y})$, so $\nabla f(\mathbf{x_0}) \cdot \mathbf{y} \geq 0$.

By Farkas' lemma, there exist nonnegative Lagrange multipliers $\boldsymbol{\pi} \in \mathbb{R}^p$ such that $\nabla f(\mathbf{x_0}) = -\sum_{j \in J(\mathbf{x_0})} \pi_j \nabla g_j(\mathbf{x_0})$. Hence

$$\nabla f(\mathbf{x_0}) + \sum_{j=1}^{p} \pi_j \nabla g(\mathbf{x_0}) = 0,$$

where $\pi_j = 0$ if $j \notin J(\mathbf{x_0})$. Thus this yields the conditions of the KKT theorem. The theorem is a direct consequence of these two proofs. ■

Appendix B

Performance graphs of belief propagation

The following pages are graphs plotting the convergence and error of particular types of graphs. First are the graphs for grids, secondly are the graphs for disjoint cycle graphs, and the last group are the graphs for random graphs.

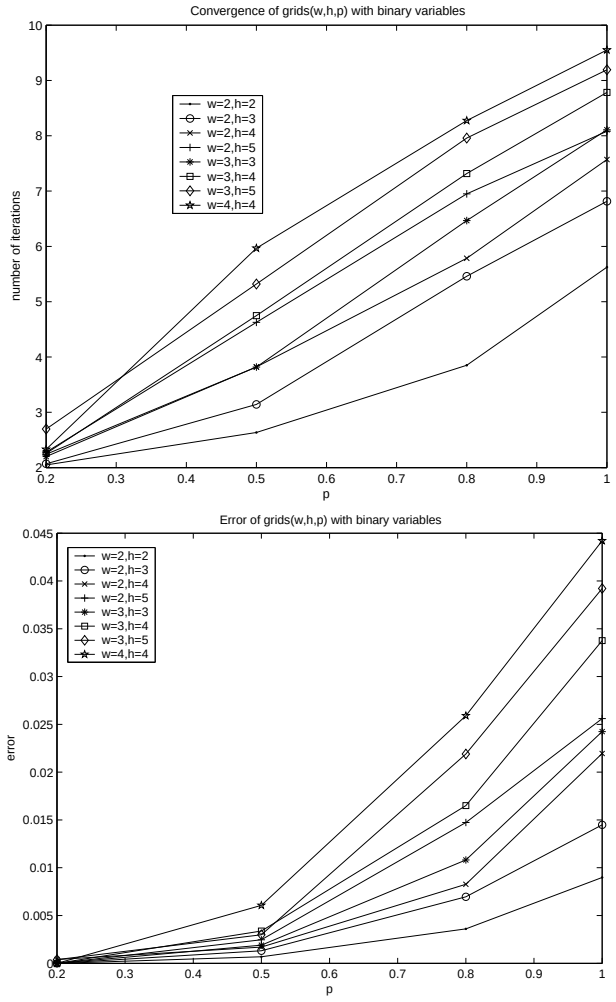

Fig. B.1: Performance of grids with binary variables.

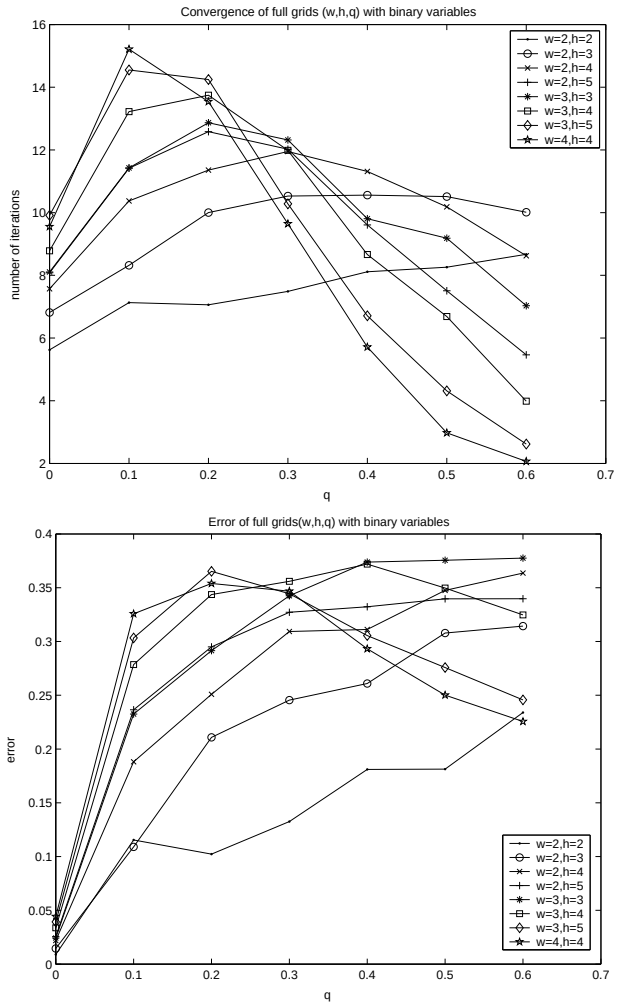

Fig. B.2: Performance of full grids with non-grid edges and binary variables.

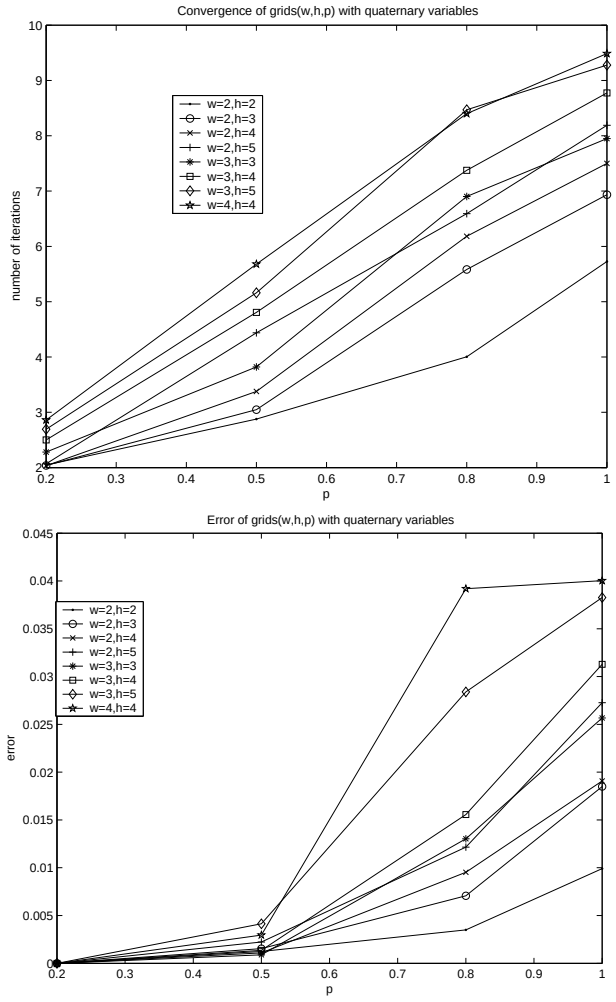

Fig. B.3: Performance of grids with quaternary variables.

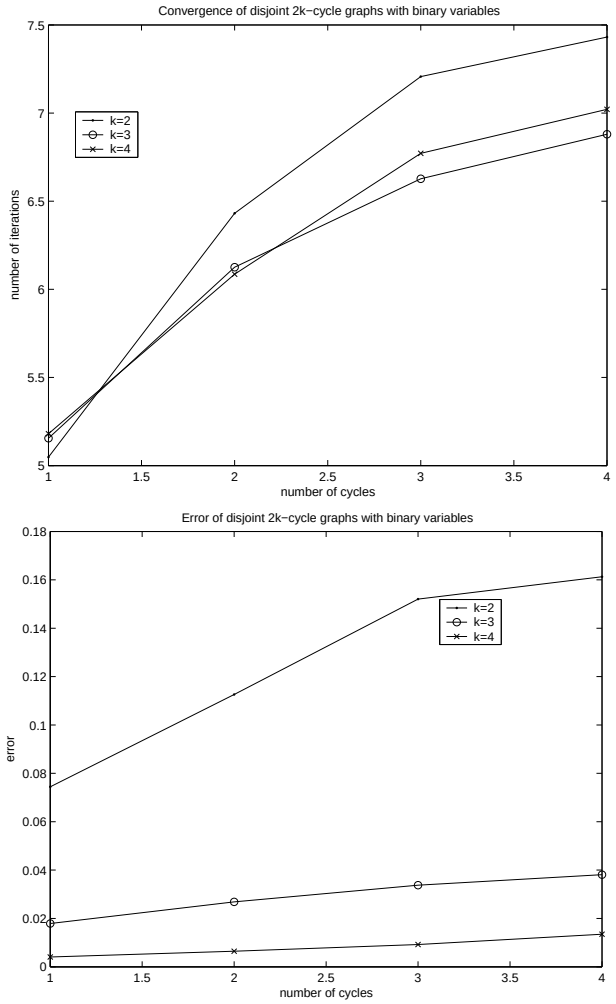

Fig. B.4: Performance of disjoint cycle graphs.

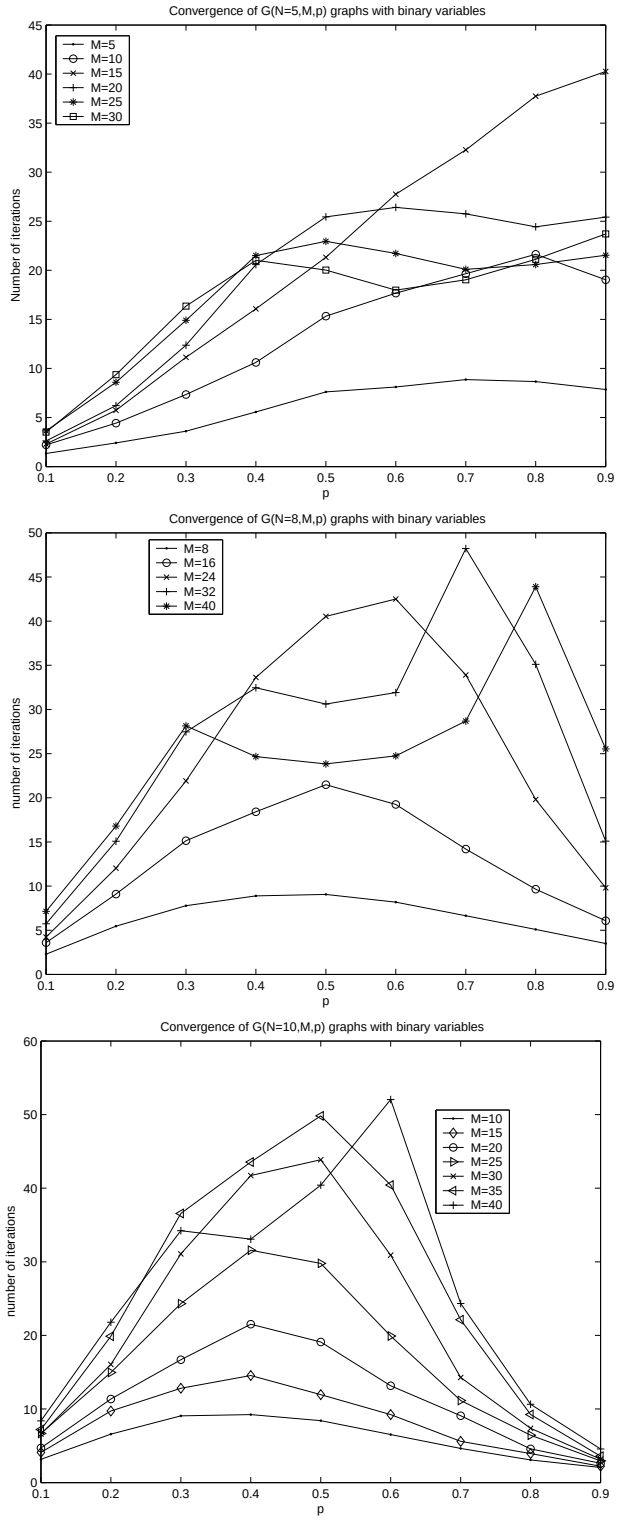

Fig. B.5: Convergence of random graphs plotted against the edge probability.

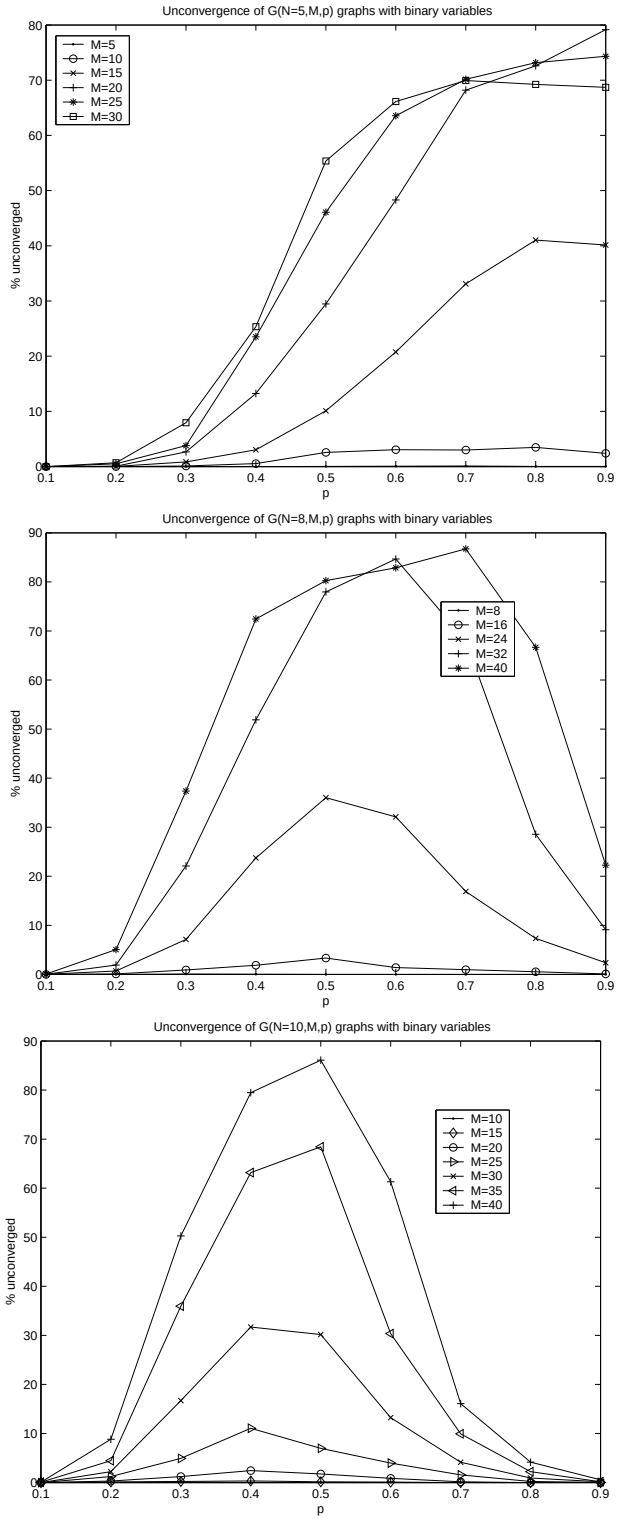

Fig. B.6: Unconvergence of random graphs plotted against the edge probability.

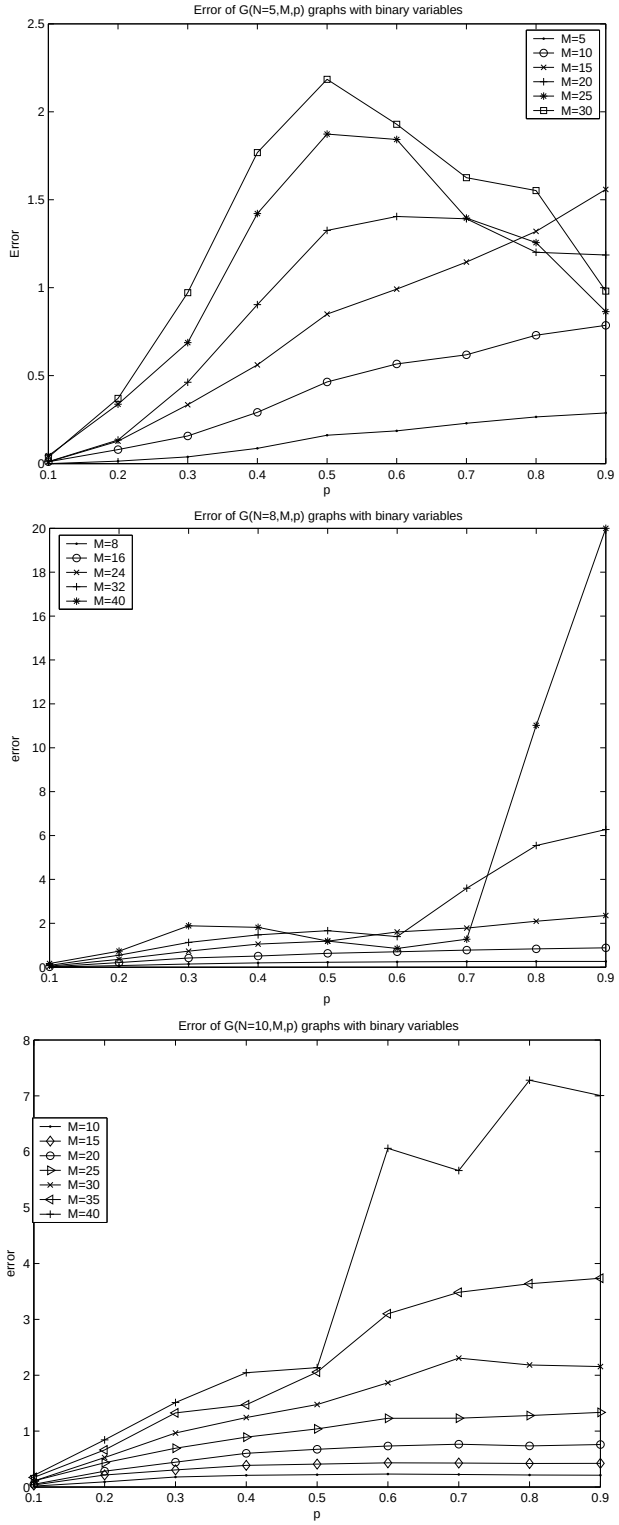

Fig. B.7: Error of random graphs plotted against the edge probability.

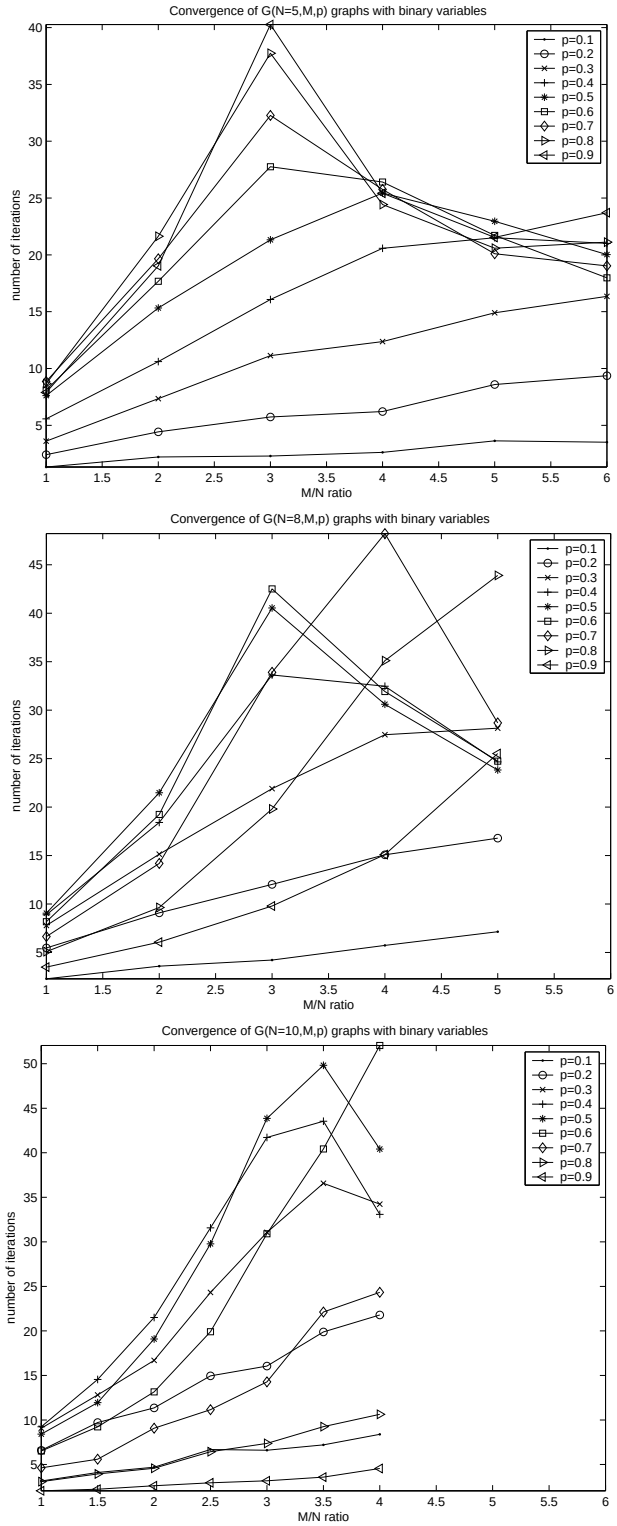

Fig. B.8: Convergence of random graphs plotted against the factor-variable ratio.

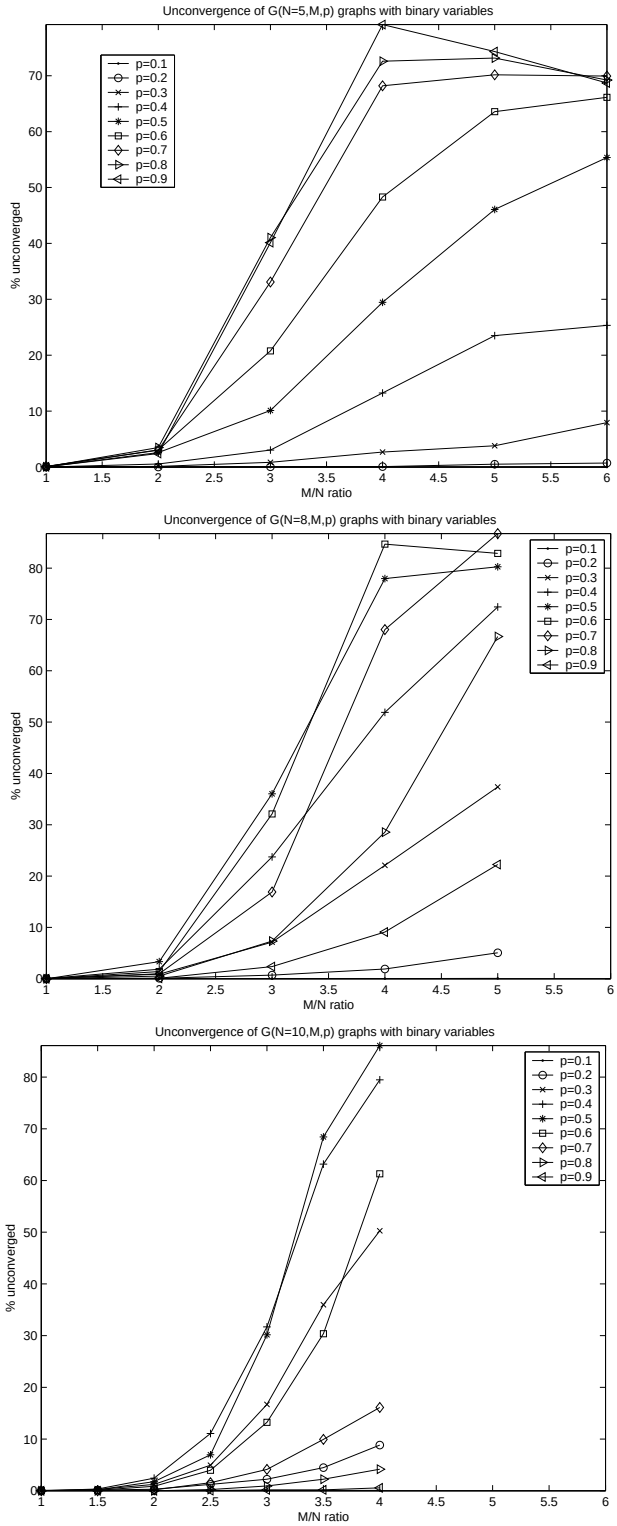

Fig. B.9: Unconvergence of random graphs plotted against the factor-variable ratio.

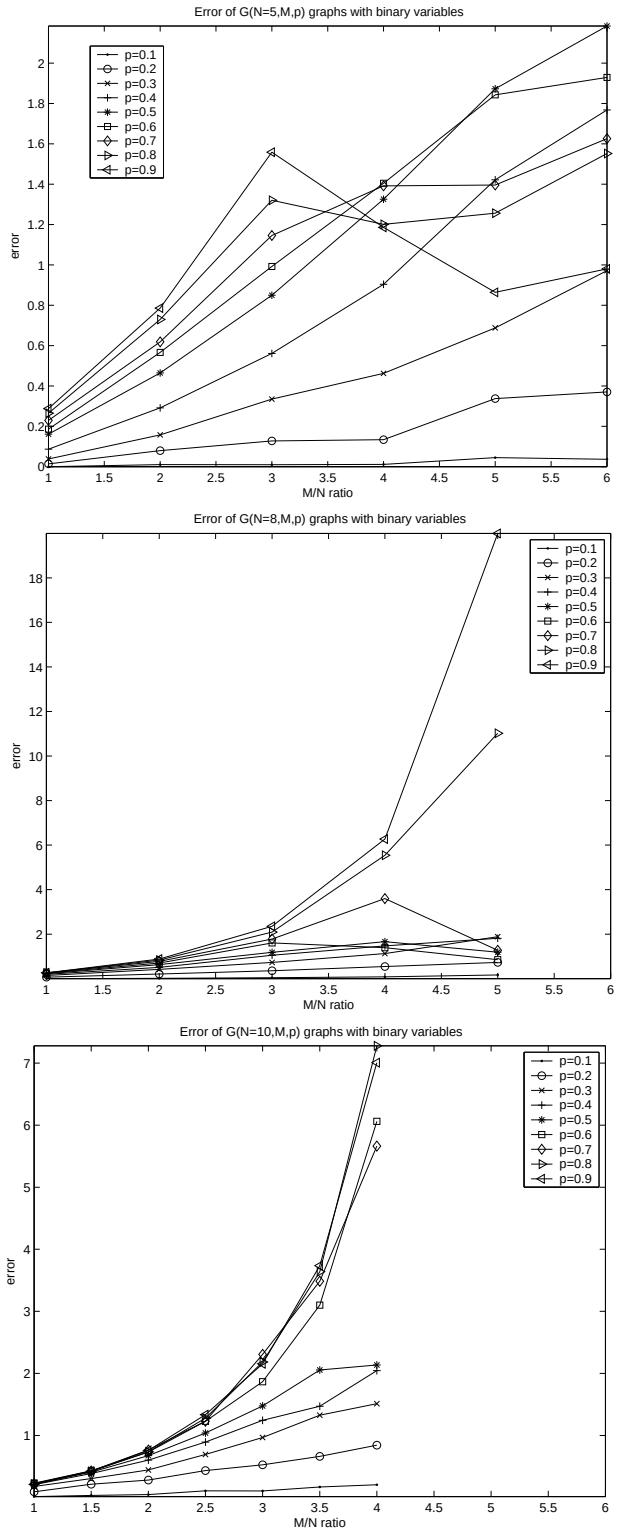

Fig. B.10: Error of random graphs plotted against the factor-variable ratio.

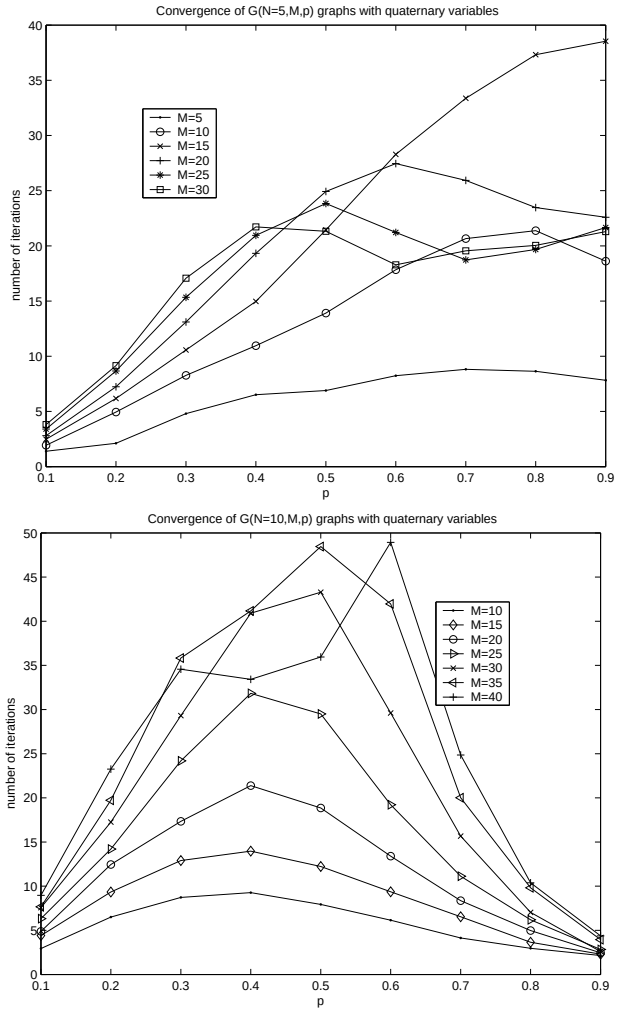

Fig. B.11: Convergence of random graphs with quaternary variables.

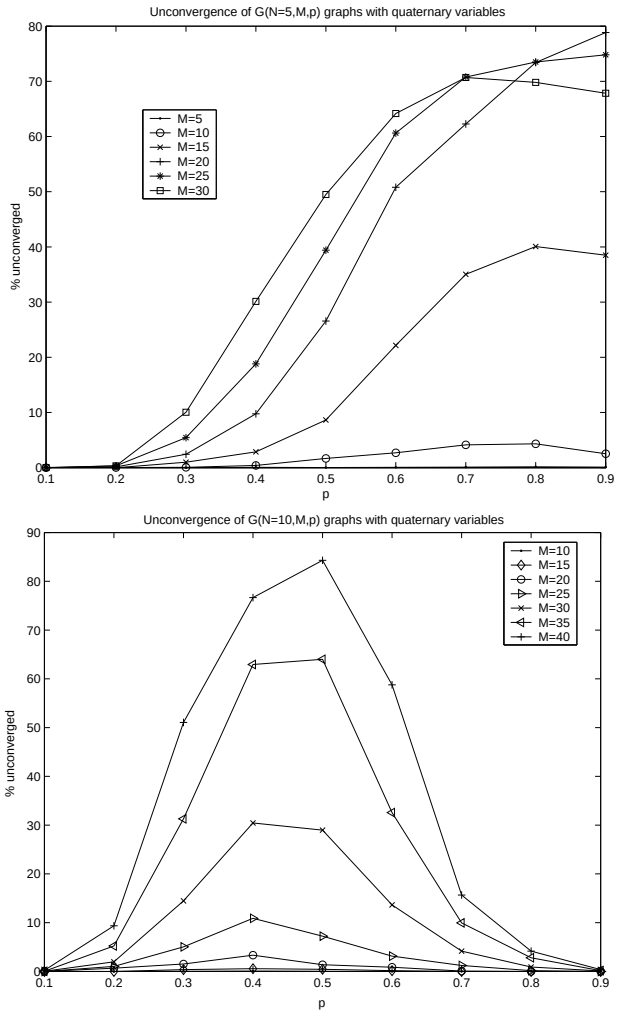

Fig. B.12: Unconvergence of random graphs with quaternary variables.

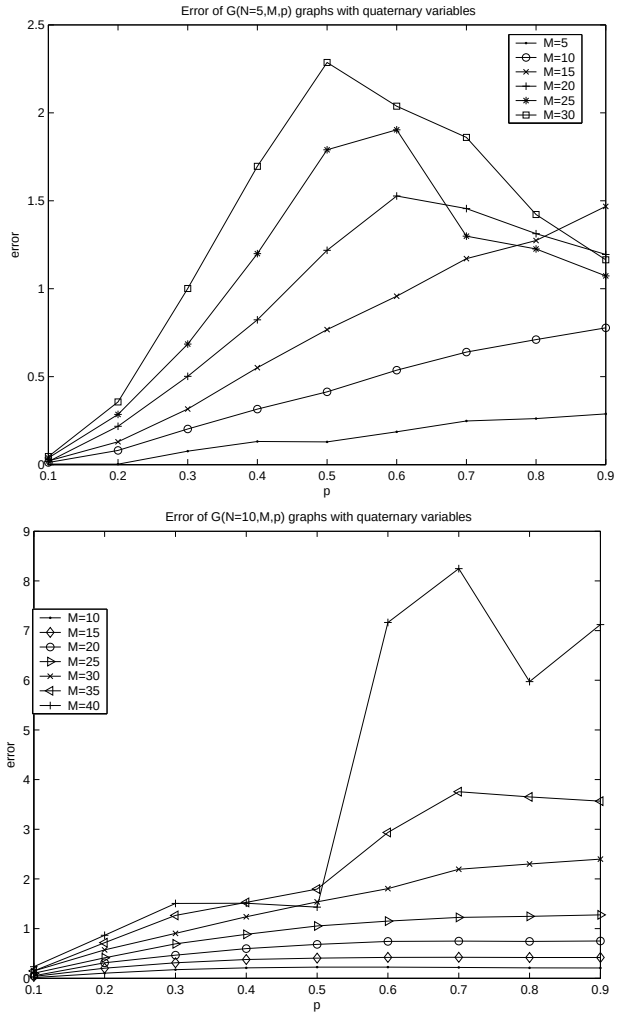

Fig. B.13: Error of random graphs with quaternary variables.

Bibliography

[AH98] A. M. Abdelbar and S. M. Hedetniemi. Approximating MAPs for belief networks is NP-hard and other theorems. *Artificial Intelligence*, 102:21–38, 1998.

[AHM98] Srinivas M. Aji, Gavin B. Horn, and Robert J. McEliece. On the convergence of iterative decoding on graphs with a single cycle. In *Proceedings of CISS*, 1998.

[AM97] Srinivas Aji and Robert McEliece. A general algorithm for distributing information in a graph. In *Proceedings of the IEEE International Symposium on Information Theory*, 1997.

[AM00] Srinivas Aji and Robert McEliece. The general distributive law. March 2000.

[AM01] Srinivas Aji and Robert McEliece. The generalized distributive law and free energy minimization. In *Proceedings of the 39th Allerton Conference on Communication, Control, and Computing*, pages 672–681, 2001.

[AP04] Dimitris Achlioptas and Yuval Peres. The threshold of random k-SAT is $2^k \log 2 - O(k)$. *Journal of the AMS*, 17:947–973, 2004.

[Bet35] Hans Bethe. Statistical theory of superlattices. *Proceedings of the Royal Society of London A*, 1935.

[BK02] Christian Borgelt and Rudolf Kruse. *Graphical Models: Methods for Data Analysis and Mining QA 76.9 .D43B67 ENG*. John Wiley and Sons, Inc, 2002.

[BMZ03] Alfredo Braunstein, Marc Mézard, and Riccardo Zecchina. Survey propagation: an algorithm for satisfiability. *http://fr.arxiv.org/abs/cs.CC/0212002*, 2003.

[CK02] Peter Clote and Evangelos Kranakis. *Boolean functions and computation models*. Springer, 2002.

[Coo90] Gregory F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[CZ01] Edwin Chong and Stanislaw Zak. *An introduction to optimization*. John Wiley and Sons, Inc, 2001.

[DL93] Paul Dagum and Michael Luby. Approximating probabilistic inference in bayesian belief networks is np-hard. *Artificial Intelligence*, 60:141–153, 1993.

[Fre98] Brendan Frey. *Graphical Models for Machine Learning and Digital Communication Q325.5 .F74 GER*. MIT Press, 1998.

[Gal63] Robert G. Gallager. *Low-density parity-check codes*. MIT Press, 1963.

[GG84] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[GMS] José Gamez, Serafin Moral, and Antonio Salmerón.

[Hes03] Tom Heskes. Stable fixed points of loopy belief propagation are minima of the bethe free energy. In *Advances in Neural Information Processing Systems*, pages 343–350. MIT Press, 2003.

[Hes04] Tom Heskes. On the uniqueness of loopy belief propagation fixed points. *Neural Computation*, 16(11):2379–2413, November 2004.

[HM97] Gavin Horn and Robert McEliece. Belief propagation in loopy Bayesian networks: experimental results. In *Proceedings of ISIT*, 1997.

[IIW05] Alexander Ihler, John Fisher III, and Alan Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6:905–936, May 2005.

[Ish81] Valerie Isham. an introduction to spatial point processes and Markov random fields. *International Statistical Review*, 49:21–43, 1981.

[Jen01] Finn Jensen. *Bayesian Networks and decision graphs QA279.5 J45 MATH.* Springer, 2001.

[JJ00] Gareth Jones and J. Mary Jones. *Information and coding theory.* Springer-Verlag, 2000.

[Jor99] Michael Jordan. *Learning in Graphical Models QA279 .L375 MATH.* MIT Press, 1999.

[KF98] Frank Kschischang and Brendan Frey. Iterative decoding of compound codes by probability propagation in graphical models. *IEEE Journal on Selected Areas in Communications*, 16(2):219–230, 1998.

[KF02] Daphne Koller and Nir Friedman. Bayesian networks and beyond. Draft, 2002.

[KFL01] Frank Kschischang, Brendan Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2), February 2001.

[KT50] H.W. Kuhn and A.W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, 1950.

[Lau96] Steffen Lauritzen. *Graphical Models QA279 L28 GER*, volume 17 of *Oxford Statistical Science Series*. Clarendon Press, Oxford, UK, 1996.

[LMP01] Michael Littman, Stephen Majercik, and Toniann Pitassi. Stochastic boolean satisfiability. *Journal of Automated Reasoning*, 27(2):251–296, 2001.

[LS88] Steffen Lauritzen and D Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B*, 50:157–224, 1988.

[Mac03] David MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[MK05] Joris Mooij and H. J. Kappen. Sufficient conditions for convergence of loopy belief propagation. Technical Report TR-2002-35, Radboud University Numegen, 2005. http://arxiv.org/abs/cs.IT/0504030.

[MMC98] Robert McEliece, David MacKay, and Jung-Fu Cheng. Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2), February 1998.

[MMW05] Elitza Maneva, Elchanan Mossel, and Martin Wainwright. A new look at survey propagation and its generalizations. In *Proceedings of SODA*, 2005.

[MN96] David MacKay and Radford Neal. Near Shannon limit performance of low density parity check codes. *Electronics Letters*, 32(18):1645–1646, August 1996. http://www.inference.phy.cam.ac.uk/mackay/abstracts/mncEL.html.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MR01] Brian Marcus and Joachim Rosenthal. *Codes, systems, and graphical models*. Springer, 2001.

[MT96] Jerrold Marsden and Anthony Tromba. *Vector calculus*. W. H. Freeman and Company, New York, fourth edition, 1996.

[MWJ99] Kevin Murphy, Yair Weiss, and Michael Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999.

[MY02] Robert McEliece and Muhammed Yildirim. Belief propagation on partially ordered sets. In D Gilliam and J Rosenthal, editors, *Mathematical Systems Theory in Biology, Communication, Computation, and Finance*, pages 275–300. Springer, 2002.

[Nea90] Richard Neapolitan. *Probabilistic Reasoning in Expert Systems: Theory and Algorithms QA76.76 E95N43 GER.* John Wiley and Sons, Inc, 1990.

[Par02] James Park. MAP complexity results and approximation methods. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 388–396, 2002.

[PD04] James Park and Adnan Darwiche. Complexity results and approximation strategies for map explanations. *Journal of Artificial Intelligence Research*, 21:101–133, 2004.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference Q335 P383 GER.* Morgan Kaufmann Publishers, Inc., San Francisco, USA, second edition, 1988.

[Rot96] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82:273–302, 1996.

[RU01] Thomas Richardson and Rüdiger Urbanke. The capacity of low-density parity check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2), February 2001.

[Sha48] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[Shi94] Solomon Eyal Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68:399–410, 1994.

[Spi65] Michael Spivak. *Calculus on Manifolds*. W A Benjamin, Inc., New York, 1965.

[Spi94] Michael Spivak. *Calculus*. Publish or Perish, Inc., Houston, USA, third edition, 1994.

[TJ01] Sekhar Tatikonda and Michael I. Jordan. Loopy belief propagation and gibbs measures. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 493–500, 2001.

[Urq03] Alasdair Urquhart. Notes on survey propagation. unpublished notes, 2003.

[Wei01] Yair Weiss. Correctness of local probability propagation in graphical models. In Michael Jordan and Terrence Sejnowski, editors, *Graphical Models: Foundations of Neural Computation QA76.87 .G72 ENG*. MIT Press, 2001.

[WF01] Yair Weiss and William Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2), February 2001.

[WJW03] Martin Wainwright, Tommi Jaakkola, and Alan Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 49(5), May 2003.

[Xia02] Yang Xiang. *Probabilistic Reasoning in multiagent systems: a graphical models approach*. Cambridge University Press, 2002.

[YFW01] Jonathan Yedidia, William Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. Technical Report TR-2001-22, Mitsubishi Electric Research Laboratories, 2001.

[YFW02] Jonathan Yedidia, William Freeman, and Yair Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report TR-2002-35, Mitsubishi Electric Research Laboratories, 2002.