Universität des Saarlandes
Max-Planck-Institut für Informatik

Message Passing Algorithms

Masterarbeit im Fach Informatik
Masters Thesis in Computer Science
von / by

Megha Khosla

angefertigt unter der Leitung von / supervised by

Prof. Dr. Kurt Melhorn

betreut von / advised by

Dr. Konstantinos Panagiotou

begutachtet von / reviewers

Prof. Dr. Kurt Melhorn
Dr. Konstantinos Panagiotou

December 2009

Declaration of Authorship

I, Megha Khosla, declare that this thesis titled, 'Message Passing Algorithms' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Masters degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ________________________________

Date: ________________________________

Abstract

Constraint Satisfaction Problems (CSPs) are defined over a set of variables whose state must satisfy a number of constraints. We study a class of algorithms called Message Passing Algorithms, which aim at finding the probability distribution of the variables over the space of satisfying assignments. These algorithms involve passing local messages (according to some message update rules) over the edges of a factor graph constructed corresponding to the CSP. We focus on the Belief Propagation (BP) algorithm, which finds exact solution marginals for tree-like factor graphs. However, convergence and exactness cannot be guaranteed for a general factor graph. We propose a method for improving BP to account for cycles in the factor graph. We also study another message passing algorithm known as Survey Propagation (SP), which is empirically quite effective in solving random $K-SAT$ instances, even when the density is close to the satisfiability threshold. We contribute to the theoretical understanding of SP by deriving the SP equations from the BP message update rules.

Acknowledgements

I would like to thank Prof. Kurt Melhorn for giving me the opportunity to pursue this thesis under his supervision and his timely and valuable inputs. I am grateful to my advisor, Konstantinos Panagiotou for bringing an interesting and challenging research topic to my attention. I also thank him for his persistent support and patience through many discussions we had through the course of the thesis. He has been an excellent advisor. I thank my parents for being a source of continued emotional support and teaching me to aim high. I am thankful to all my friends for their support and encouragement.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

List of Figures vi

1 Introduction 1
- 1.1 Summary of Contributions 4
- 1.2 Organization 4

2 Preliminaries 6
- 2.1 Constraint Satisfaction Problems 6
- 2.2 Graphical Models 7
  - 2.2.1 Conditional Independence and Markovian Properties 9
  - 2.2.2 Factor Graphs 12
- 2.3 Exploiting Structure for Efficient Computations 13

3 Belief Propagation 15
- 3.1 Introduction 15
- 3.2 Beliefs and Message Equations 15
- 3.3 Belief Propagation for Trees 18
- 3.4 Loopy Belief Propagation for General Graphs 22
  - 3.4.1 Conditioning/Clamping for Exact Marginals 23
  - 3.4.2 Exact BP Algorithm for General Graphs 26

4 Properties of Loopy Belief Propagation 34
- 4.1 Introduction 34
- 4.2 Fixed Points of BP 35
- 4.3 Generalized Belief Propagation 38
- 4.4 Correctness of Loopy BP for some Graphical Models 40

5 Survey Propagation 44
- 5.1 Introduction 44

5.2 Solution Space of HARD-SAT Problems 45
5.3 Solutions Clusters and Covers 47
5.4 The SP Algorithm 48
5.4.1 Deriving SP Equations for Trees 49
5.5 Discussion 57

Bibliography **58**

List of Figures

2.1 Constraint graph corresponding to the formula $F = (x \vee y) \wedge (\neg x \vee z)$. . . . 7
2.2 A factor graph for a general CSP. 13

3.1 Diagrammatic representation for the belief of a variable node i. 16
3.2 Diagrammatic representation for belief of factor node A. 16
3.3 A rooted tree factor graph. 20
3.4 BP messages in a tree. 21
3.5 Factor graph representing the SAT formula $(p \vee q \vee r) \wedge (\neg p \vee q \vee s)$. . . . 24
3.6 Diagrammatic representation of the clamping procedure. 24
3.7 Factor graph of $2N$ variables and pairwise constraints. 26
3.8 The cutset S. 26
3.9 The decomposition of 2×5 grid. 28
3.10 Factor graph. 29
3.11 Diagrammatic representation of the output of decompose(G). 30

4.1 A Markov network. 40
4.2 Factor Graph corresponding to a pairwise Markov network. 41

5.1 Diagrammatic representation of the solution space of random 3-SAT with respect to α. Satisfying clusters are represented by gray and the unsatisfying by white. 46
5.2 A part of a factor graph showing variable node i and its neighboring factor nodes. 48
5.3 A part of tree-factor graph for a SAT formula. 50
5.4 Message update equations for enumerating covers of a formula. 54

Chapter 1

Introduction

Constraint Satisfaction Problems (CSPs) appear in a large spectrum of scientific disciplines. In any constraint satisfaction problem, there is a collection of variables all of which have to be assigned values, subject to specified constraints. Computational problems like scheduling a collection of tasks, or interpreting a visual image, can all be seen as CSPs. We are interested in finding a satisfying assignment for the CSP, which means that we need to assign values to each of the variables from their respective domain spaces, such that all the constraints are satisfied.

A satisfying assignment to a CSP can be found by setting variables iteratively as follows. Suppose that we can find a variable that assumes a specific value in most of the solutions. We refer to such a variable as the most *biased* variable. We set this variable to its preferred value and simplify the problem by removing the variable and removing the satisfied constraints. We repeat the process until all the variables have been assigned or we reach a contradiction. Such a strategy of assigning variables one by one is known as *decimation*. Note that by setting the variable to its most preferred value, we assure that the reduced problem inherits most of the solutions from the parent problem. Now the basic task of finding the most biased variable can be reduced to computing the *marginal* probability distribution of the involved variables over the probability space of satisfying assignments.

To understand the notion of the marginal probability, consider some variable X in a CSP $\mathcal{P}$ and a set of satisfying assignments $S(\mathcal{P})$. Now if we draw uniformly at random a satisfying assignment from $S(\mathcal{P})$, the probability that a variable X will assume some value x is the marginal probability for the occurrence of the event "$X = x$" in the probability space of satisfying assignments. With the above description, the computation of marginals requires us to enumerate all the satisfying assignments, which is harder than finding a single satisfying assignment. Such a naive counting procedure will require exponential time in the number of variables.

Message Passing Algorithms address the above problem of calculating the marginal probability distribution in computationally tractable time. The key observation behind the design of such algorithms is the fact that the underlying joint probability density over solutions in a CSP can be factorized in terms of local functions (factors), each dependent on a subset of variables. Such a distribution is represented by a *factor graph*: a bipartite graph with nodes for the variables and for the factors. Each factor node is connected to every variable node over which the factor is defined. Message Passing Algorithms operate on such factor graphs and allow nodes to communicate by sending messages over the edges. They can be seen as dynamic programming solutions, where a node collects results from a sub-part of the graph and communicates it to the next neighbor via a message. The receiving node further repeats the process. Once a node receives messages from all of its neighbors and messages are no longer changing, the global solution (marginal in our case) for that particular node can be computed or approximated.

The Belief Propagation (BP) Algorithm belongs to this class of algorithms and calculates exact solution marginals when the factor graph is a tree. The algorithm proceeds in iterations as follows. In every iteration, messages are sent along both directions of every edge. The outgoing message from a particular node is computed based on the incoming messages to this node in the previous iteration from all other neighbors. When the messages converge to a fixed point or a prescribed number of iterations has passed, the *beliefs* for each of the variables are estimated based on the fixed incoming messages into the variable node. The message update rules are such that if the graph is acyclic, then the beliefs correspond to exact solution marginals. For general graphs with cycles, BP may not converge or give inaccurate marginals. In this thesis, we focus on this issue and propose an exact algorithm based on decomposing the factor graph. However, the running time of this algorithm may be superpolynomial depending on the underlying structure of the factor graph. More details can be found in Chapter 3.

Constraint Satisfaction Problems have received considerable attention from the statistical physics community, where they represent particular examples of spin systems. Belief Propagation fails to converge for dense CSPs, i.e., when the ratio of number of constraints to the number of variables is high. The reason has been attributed to the solution space of these CSPs. It has been explained that as the constraint density increases, there occurs a phase transition from the satisfiable regime to the unsatisfiable regime at some critical density α_s. Also, there exists another threshold α_{clust}, which divides the satisfiable regime into two separate regions, called EASY-SAT and HARD-SAT. In the EASY-SAT region, the solution space is connected and it is possible to move from one solution to any other with a number of single bit flips. In this region, BP and other simple heuristics can easily find a solution. In the HARD-SAT region, the solution space breaks up into *clusters*, so that moving from a satisfying assignment within one cluster to some other assignment in another cluster requires flipping some constant fraction of the variables simultaneously. Such a phenomena suggests that different parts of the problem may obtain locally optimal configurations corresponding to different clusters, that cannot be merged to find a global configuration. This serves as an intuitive explanation for the failure of local search algorithms like BP in the clustered region. We will elaborate more on this issue in Chapter 5.

To understand a cluster, one can think of the solution graph for a CSP in which nodes correspond to solutions and are neighbors if they differ on the assignment of exactly one variable. A cluster is then a connected component of the solution graph. The variables which take the same state in all the satisfying assignments within a cluster are termed as *frozen* variables whereas others are *free* variables that do change their values within the cluster. To account for free variables, the domain space is extended to include an undecided state or a *joker* state "$*$". For Boolean CSPs, the above description attributes to each cluster an assignment, which is a single string in $\{0, 1, *\}^n$. Here, the free variables assume the "$*$" state in the respective cluster.

A relatively new and exciting technique known as Survey Propagation (SP) turns out to be able to deal with the clustering phenomena for hard instances of a random CSP like $K - SAT$. SP has its origins in sophisticated arguments in statistical physics, and can be derived from an approach known as the *cavity* method. Algorithmically, it is a message passing algorithm and computes/approximates marginals over clusters of a CNF formula, i.e., it computes the fraction of clusters in which a variable assumes one of the values from $\{0, 1, *\}$. Survey Propagation has now been discovered to be a special form of Belief Propagation. We contribute to the theoretical understanding of SP by giving a simpler derivation of SP equations from the BP message update rules. The BP message rules are interpreted as combinatorial objects counting the number of satisfying assignments of a sub formula of the original formula in which some variable assumes a particular value. These rules are modified to count the number of cluster assignments under the same conditions. The new beliefs corresponding to the modified message update rules calculate the marginal distribution of variables over clusters in a formula. Such marginals are provably exact only for tree-factor graphs. The experimental results suggest that SP computes a good approximation of marginals for general graphs too.

We now summarize the main contributions of this thesis.

1.1 Summary of Contributions

In this thesis, we contribute to the field by the following.

Exact BP algorithm. We propose an exact BP algorithm for general graphs with cycles. Our method can be understood as an improvement over the Pearl's method of conditioning in which an acyclic structure is derived from the original graph by fixing some variables to one of their values. BP can then be used to compute exact marginals over such a structure. We make use of the observation that a factor graph is inherently a Markov Random field and can be decomposed into smaller parts, which can then be solved independently. We fix the variables so as to divide the original graph into conditionally independent parts. We carry on such a decomposition recursively till we obtain acyclic components. The exact marginals obtained by running standard BP over these acyclic structures can then be combined to compute the corresponding marginals in the parent graph. We show by example that for certain graphs over which standard BP may not converge and the exact solution by Pearl's conditioning takes exponential time, our algorithm returns exact marginals in polynomial time. We prove that in general, the running time for the algorithm is $O(2^M N^2)$ where M is the maximum number of variables fixed along any path (of decomposition) from the original graph to some acyclic component. Moreover, we use this result to prove the following corollary for minor-free graphs.

Corollary 1.1. *Let G be a factor graph with N variable nodes that does not contain K_h as a minor. Then the marginal probabilities for G can be computed in time $O(2^{h^{3/2}\sqrt{N}} N^2)$.*

Understanding SP. We establish a link between SP and BP by deriving SP equations from BP message update rules. We consider the CNF formulas for which the corresponding factor graph is a tree. We show that the BP messages over these factor graphs count the number of solutions corresponding to a sub-formula in which some variable assumes a particular value. We modify these messages to enumerate clusters instead of solutions under the same conditions. We prove that these new messages are indeed equivalent to the SP update equations.

1.2 Organization

The organization of the thesis is summarized below.

Chapter 2: **Preliminaries.** This chapter presents a basic introduction to CSPs and graphical models. We emphasize on the factorization properties of the joint probability distribution and explain how such a structure can be exploited by Message Passing Algorithms.

Chapter 3: **Belief Propagation.** In this chapter, we describe the Belief Propagation algorithm in detail. We prove the exactness of BP for tree factor graphs and discuss the conditioning method for general graphs. We also present our proposed exact algorithm for general graphs, and prove run time bounds for minor-free factor graphs.

Chapter 4: **Properties of Loopy Belief Propagation.** In this chapter, we discuss the convergence and exactness properties of standard BP over graphs with cycles, also known as loopy BP. We present some known results for the equivalence of BP with a free energy approximation known as Bethe approximation. We also discuss the correctness of BP over Gaussian models.

Chapter 5: **Survey Propagation.** This chapter presents another message passing algorithm known as Survey Propagation. We explain the solution space geometry of random $K - SAT$ problem as given by the statistical physics community. We present a derivation of SP equations from the BP message update rules. We conclude by discussing the known experimental results of SP for random $K - SAT$ problems.

Chapter 2

Preliminaries

2.1 Constraint Satisfaction Problems

A Constraint Satisfaction Problem (CSP) is defined by a set $X = \{X_1, X_2, \ldots, X_n\}$ of variables and a set $C = \{C_1, C_2, \ldots C_m\}$ of constraints. Each variable X_i has a non-empty domain $\mathcal{X}_i$ of possible values. Each constraint C_i is defined on a subset of variables in X and specifies allowable combinations of the values for that subset. We denote by $V(C_i)$ the set of variables constrained (participating) in C_i. A solution to a CSP will be an assignment to all variables, $x = (x_1, x_2, \ldots, x_n), x_i \in \mathcal{X}_i$ such that all constraints in C are satisfied. The problem is satisfiable if there exists at least one satisfying assignment for the variables in X. As a simple example of a CSP, consider the following formula over boolean variables.

$$F = (x \vee y) \wedge (\neg x \vee z) \tag{2.1}$$

Here $X = \{x, y, z\}$ where each variable can take values from $\{0, 1\}$. Each clause defines a constraint, so the constraint set is given by $C = \{x \vee y, \neg x \vee z\}$. For F to be satisfiable, all the clauses have to be satisfied. We define the space of satisfying assignments for the formula F as

$$S(F) = \{\sigma \in \{0,1\}^{|X|} \mid \sigma(c) = 1 \ \forall c \in C\}.$$

Here, $\sigma(c) = 1$ if and only if c is satisfied (evaluates to true) when each $i \in V(c)$ assumes the values according to the configuration σ.

Now if we draw uniformly at random a satisfying assignment from $S(F)$, there are more chances to find y set to 1 than y set to 0. In other words, we find that the $Pr(y = 1 \mid \textit{all clauses in F are satisfied})$ is more than that of the event "$y = 0$". Once such a distribution is known, we know which variables are most likely to assume a particular value, and setting those variables simplifies the problem. We call such a probability distribution as *marginal* probability distribution defined over the space of satisfying assignments. Formally, we have the following definition.

Definition 2.1 (Marginal Probability Distribution)**.** Given a set $X = \{X_1, X_2, \dots, X_n\}$ of random variables whose joint distribution is known, the marginal distribution of X_i is the probability distribution of X_i averaging over all information about $X \backslash X_i$. This is calculated by summing or integrating the joint probability distribution over $X \backslash X_i$. Particularly, if $i = 1$, the marginal probability is given by

$$Pr(X_1 = x_1) = \sum_{\{x_2, x_3, \dots, x_n\} \in \mathcal{X}_2 \times \mathcal{X}_3 \times \dots \times \mathcal{X}_n} Pr(X_1 = x_1, X_2 = x_2, \dots X_n = x_n).$$

Note that the above sum contains exponentially many terms. The naive calculation of the marginal probabilities in the case of CSP described in the previous section will require to enumerate all the satisfying assignments which counters our idea of finding a satisfying assignment using marginals. Before we provide algorithms for solving such problems in the next chapter, we introduce some basic tools.

2.2 Graphical Models

A graph $G = (V, E)$ is formed by a set of vertices $V = \{1, 2, \dots, n\}$ and a set of edges $E \subseteq \binom{V}{2}$. To define a graphical model, we associate with each vertex $s \in V$ a random variable X_s taking values in some space $\mathcal{X}_s$. We will deal with discrete and finite space $\mathcal{X}_s$ (e.g $\mathcal{X}_s = \{1, 2, \dots, r\}$). We use lower-case letters (e.g. $x_s \in \mathcal{X}_s$) to represent particular elements of $\mathcal{X}_s$ and the notation $\{X_s = x_s\}$ corresponds to the event that the random variable X_s takes the value $x_s \in \mathcal{X}_s$. For any $S \subseteq V$, we have an associated subset of random variables $X_S = \{X_s, s \in S\}$ with the notation $X_S = \{x_s, s \in S\}$ corresponding to the event that the random vector X_S takes the configuration $\{x_s\}$. Graphical Models provide an intuitive interface for representing interacting sets of variables. It is helpful

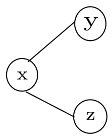

Figure 2.1: Constraint graph corresponding to the formula $F = (x \vee y) \wedge (\neg x \vee z)$.

to visualize the formula in Equation (2.1) as a constraint graph in Figure 2.1 where the nodes correspond to the variables and the edges represent the constraints. Here, constraints are the clauses that allow combinations of the member variables such that the corresponding clause evaluates to 1. We write the set of constraint functions as $C = \{f_{xy}(x, y), f_{xz}(x, z)\}$ where

$$f_{xy}(x_i, y_i) = \begin{cases} 1, & \text{if clause corresponding to edge } (x, y) \\ & \text{evaluates to 1 with } x = x_i \text{ and } y = y_i \\ 0, & \text{otherwise} \end{cases}.$$

We now define a global function G on a configuration of variables in F which evaluates to 1 if and only if all constraints in the problem are satisfied.

$$G(x_i, y_i, z_i) = \begin{cases} 1, & \text{if } f_{xy}(x_i, y_i) \cdot f_{xz}(x_i, z_i) = 1 \\ 0, & \text{otherwise} \end{cases}.$$

The joint probability distribution over the space of satisfying assignments, $S(F)$ can then be written as

$$Pr(x = x_i, y = y_i, z = z_i) = \begin{cases} \frac{1}{Z}, & \text{if } G(x_i, y_i, z_i) = 1 \\ 0, & \text{otherwise} \end{cases},$$

where $Z = |S(F)|$, i.e. the total number of satisfying assignments of F. So,

$$\begin{aligned} Pr(x = x_i, y = y_i, z = z_i) &= \frac{1}{Z} G(x_i, y_i, z_i) \\ &= \frac{1}{Z} f_{xy}(x_i, y_i) \cdot f_{xz}(x_i, z_i) \end{aligned}. \tag{2.2}$$

By summing both sides of (2.2) over all possible configurations of the variables, we can count the total number of satisfying assignments as

$$Z = \sum_{\{x_i, y_i, z_i\}} f_{xy}(x_i, y_i) \cdot f_{xz}(x_i, z_i). \tag{2.3}$$

Particularly, for the formula F

$$\begin{aligned} Z = &\sum_{\{x_i, y_i, z_i\}} f_{xy}(x_i, y_i) \cdot f_{xz}(x_i, z_i) \\ =& f_{xy}(0,0) \cdot f_{xz}(0,0) + f_{xy}(0,0) \cdot f_{xz}(0,1) + f_{xy}(0,1) \cdot f_{xz}(0,0) \\ &+ f_{xy}(0,1) \cdot f_{xz}(0,1) + f_{xy}(1,0) \cdot f_{xz}(1,0) + f_{xy}(1,0) \cdot f_{xz}(1,1) \\ &+ f_{xy}(1,1) \cdot f_{xz}(1,0) + f_{xy}(1,1) \cdot f_{xz}(1,1) \\ =& 4. \end{aligned} \tag{2.4}$$

The marginal distribution for the variable y is given by

$$\begin{aligned}
Pr(y=1) =& \frac{1}{Z}\sum_{\{x_i,z_i\}} Pr(x=x_i, y=1, z=z_i) \\
=& \frac{1}{4}\sum_{\{x_i,z_i\}} f_{xy}(x_i,1)\cdot f_{xz}(x_i,z_i) \\
=& \frac{1}{4}\Big\{ f_{xy}(0,1)\cdot f_{xz}(1,0) + f_{xy}(0,1)\cdot f_{xz}(1,1) \\
& + f_{xy}(1,1)\cdot f_{xz}(1,0) + f_{xy}(1,1)\cdot f_{xz}(1,1)\Big\} \\
=& \ \frac{3}{4}.
\end{aligned} \tag{2.5}$$

Intuitively, marginals represent the fraction of satisfying assignments in which the variable is assigned a particular value. We also note that that the joint distribution over satisfying assignments in a CSP can be factorized and the marginals in turn can be expressed in a sum-product form. Such factorizable distributions are known to obey certain properties called Markovian properties and the corresponding graphical model form a Markov Random field.

In the next section we will explain what a Markov Random field is. We will then introduce a more general graphical model to represent such factorizable distributions or Markov Random fields.

2.2.1 Conditional Independence and Markovian Properties

Consider again the formula F in Equation (2.1). If we fix the value of x in a configuration, y and z can take values independent of each other in a satisfying assignment (if one exists) i.e. they are no longer correlated. In other words, knowing the value of y does not affect the value z takes in the satisfying assignment and vice versa as long as x is fixed. Such a type of independence between y and z given x is known as conditional independence.

Informally, two random variables are independent if observing one does not give any information about the other. They are conditionally independent, if they become independent once a set of another random variables has been observed. We now formally define conditional independence.

Definition 2.2 (Conditional Independence)**.** Two random vectors X and Y are conditionally independent given random vector Z iff

$$\forall x \in Dom(X), y \in Dom(Y), z \in Dom(Z)$$

$$P(X=x, Y=y \mid Z=z) = P(X=x \mid Z=z)\cdot P(Y=y \mid Z=z).$$

Intuitively, learning Y does not influence distribution of X if you already know Z. As in literature, we use the following short notation for conditional independence between X and Y given Z.

$$X \perp\!\!\!\perp Y \mid Z. \tag{2.6}$$

Again, the graph in Figure 2.1 aids in visualizing the conditional independence between y and z by means of a missing edge. Also, note that the nodes y and z become disconnected in absence of node x. In terms of graphical models, conditional independence can be defined as follows.

Definition 2.3. Two (sets of) nodes A and B are said to be conditionally independent given a third set, S, if all paths between the nodes in A and B are separated by a node in S.

Such an undirected graphical model where the edge structure is used to represent the conditional independence among the nodes is termed as a *Markov Random Field*. Interestingly, the concept of a Markov Random Field (MRF) originally came from the attempts to put into a general probabilistic setting a very specific model in statistical physics known as *Ising Model* [1]. Without going into the foundations of the theory of MRFs, we define Markov Random Field with respect to undirected graphical models.

Definition 2.4 (Markov Random Field)**.** Given an undirected graph $G = (V, E)$, a set of random variables $X = (X_v)_{v \in V}$ form a *Markov random field* with respect to G if they satisfy the following equivalent Markov properties [2].

1. The *pairwise Markov property*, relative to G, if for any pair (u, v) of non-adjacent vertices

$$X_u \perp\!\!\!\perp X_v \mid X_{V \setminus \{u,v\}}.$$

2. The *local Markov property*, relative to G, if for any vertex $v \in V$

$$X_v \perp\!\!\!\perp X_{V \setminus N(v)} \mid X_{N^+(v)},$$

   where $N(v)$ is the set of neighbors of v, and $N^+(v) = v \cup N(v)$ is the closed neighborhood of v.

3. The *global Markov property*, relative to G, if for any triple (A, B, S) of disjoint subsets of V, such that S separates A from B in G

$$X_A \perp\!\!\!\perp X_B \mid X_S.$$

In other words, if an undirected graphical model represents a Markov Random field, we can safely assume the presence of a cut set (a set of variable nodes) which if removed, decomposes the graph into conditionally independent parts. We will use such properties of Markov Random fields to develop a recursive algorithm which can work independently on smaller parts of the whole graph.

Also Conditional Independence or the Markovian properties establish themselves in the probability model via the factorization structure. For example if X_a is conditionally independent of X_c given X_b then by chain rule and conditional independence we have

$$\begin{aligned} Pr(x_a, x_b, x_c) =& Pr(x_a) \cdot Pr(x_b|x_a) \cdot Pr(x_c|x_a, x_b) \\ =& Pr(x_a) \cdot Pr(x_b|x_a) \cdot Pr(x_c|x_b), \end{aligned} \tag{2.7}$$

i.e., the joint distribution can be expressed as a product of factors which are local in the sense that they depend only on a subset of variables. Such factorizable distribution, if positive, can be expressed as an undirected graphical model known as *Gibbs Random field* which we define below.

Definition 2.5 (Gibbs Random field)**.** A probability measure $Pr(x_V)$ on an undirected graphical model $G = (V, E)$ is called a Gibbs distribution if it can be factorized into positive functions defined on all maximal cliques in the graph, i.e.,

$$Pr(x_V) = \frac{1}{Z} \prod_{c \in C_G} \psi(x_c),$$

where C_G is the set of all maximal cliques in the graph G and $Z = \sum_{x_V} \prod_{c \in C_G} \psi(x_c)$ is the normalization constant. A random field with Gibbs distribution is known as Gibbs Random field.

We now state the *Hammersley Clifford theorem* [3] which establishes the relation between Markov and Gibbs random fields.

Theorem 2.6 (Hammersley and Clifford)**.** *A probability distribution P with positive and continuous density f with respect to a product measure μ satisfies the pairwise Markov property with respect to an undirected graph G if and only if it factorizes according to G (i.e. it is a Gibbs Random field).*

The importance of the theorem lies in the fact that it provides a simple way of expressing the joint distribution. However, the positivity condition on the density function restricts us to apply the above theorem to CSPs. But we have already seen in the previous section that the joint distribution over satisfying assignments in a CSP can be factorized. It is easy to show that the factorizable joint distributions follow the Markovian properties (we refer [2] for the detailed proof). The Factor Graphs described in the next section are then used to represent such factorizable distributions graphically. In general, any Markov Random Field can be alternatively represented by a factor graph.

2.2.2 Factor Graphs

It is quite convenient to visualize a CSP as a graphical model such as a constraint graph. But, in case the constraints are not binary, such a graph turns out to be a hyper-graph and the factorization properties of the corresponding distribution may not be as evident. To overcome such problems, we describe a yet another graphical model known as Factor Graph.

Consider again the formula F in Equation (2.1). We have shown that the probability that a given configuration is one of the satisfying assignments is proportional to the global function defined on all variables of the formula. Also, this global function can be expressed as a product of the local functions defined for each constraint. A factor graph is useful to express such a splitting of a global function on a large number of variables into local functions involving fewer variables.

Formally, a factor graph, $G = (V, E)$ is a bipartite graph with two types of nodes.

1. **Variable nodes.** These correspond to the vertices in the original graphical model and are represented by circle nodes in the factor graph. Let $I = \{1, 2, \dots, n\}$ be the set of n variable nodes. Associated with each node i is a random variable X_i. Let x_i represent one of the possible realizations of the random variable X_i.

2. **Factor Nodes.** These correspond to the set of factors defining the joint probability distribution and are represented by square nodes in the factor graph. Let $A = \{a, b \dots m\}$ be the set of m factor nodes.

We denote by $A(i)$ the set of factors in which i appears and equivalently by $I(a)$ the set of variables participation in factor a. There is an edge $(i, a) \in E$ if and only if the x_i participates in the factor represented by $a \in A$. Formally, we define $V = I \cup A$ and $E = \{(i, a) \mid i \in I, a \in A(i)\}$. We will denote by $N(i)$, the neighbors of node i in factor graph G. The joint probability mass function is given by

$$Pr(X_1 = x_1, \dots, X_n = x_n) = \frac{1}{Z} \prod_a f_a(X_a). \tag{2.8}$$

Here a is an index labelling m functions $f_a, f_b, \dots, f_m$ and $X_a = \{x_i \mid i \in N(a)\}$ represents the configuration of variables participating in a. Z is a normalization constant. Throughout this thesis, we will use the above described notations with respect to factor graphs.

In general, a CSP can be represented by a factor graph as follows. We define a function $f_A(X_A)$ for each constraint A which evaluates to 1 if the constraint A is satisfied and 0 otherwise. We then represent these constraints by the factor nodes in the corresponding factor graph. An edge is drawn between each factor and the variables over which it is defined. Note that the constraints themselves can be arbitrary functions restricting the values taken by the participating variables. In Figure 2.2, we show a factor graph for a general CSP with four variables and four constraints defined on these variables. Variable nodes are represented by circles and factor nodes by squares.

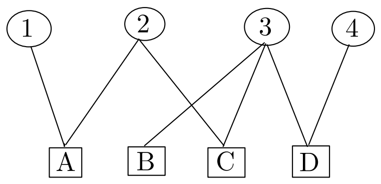

FIGURE 2.2: A factor graph for a general CSP.

Recall that we aim to find a satisfying assignment to a CSP by setting variables to their most preferred values in steps. Also, we obtain these preferred values by finding marginals for the involved variables. We have already discussed that the naive computation of marginals becomes computationally intractable as the number of variables increases. However, the joint distribution of our interest has a special form. It can be expressed as a product of local functions (factors) over smaller subset of variables. In the next section, we will show how we can exploit such a structure of the underlying model (joint distribution) to have tractable computations.

2.3 Exploiting Structure for Efficient Computations

In this section, we will elaborate on how the factorization structure of the distribution allows for the efficient computation of the marginals.

Consider the joint distribution corresponding to factor graph in Figure 2.2.

$$Pr(X_1 = x_1, \dots, X_4 = x_4) = \frac{1}{Z} f_A(x_1, x_2) f_B(x_3) f_C(x_2, x_3) f_D(x_3, x_4). \tag{2.9}$$

To compute the marginal density for X_3 we can compute

$$Pr(X_3 = x_3) = \frac{1}{Z} \sum_{\{x_1,x_2,x_4\}} f_A(x_1,x_2) f_B(x_3) f_C(x_2,x_3) f_D(x_3,x_4). \quad (2.10)$$

If each of the variables can take k values, then the total computation time is clearly $O(k^3)$ time, as the sum must be performed over k^3 admissible configurations. On the other hand, by applying the distributive law, we can move the summations over the factors that do not involve the variables being summed over.

$$Pr(X_3 = x_3) = \frac{1}{Z} f_B(x_3) \sum_{x_2} \Bigg(\sum_{x_1} f_A(x_1,x_2) f_C(x_2,x_3) \Bigg(\sum_{x_4} f_D(x_3,x_4) \Bigg) \Bigg). \quad (2.11)$$

In order to evaluate this sum, we first consider the inner sum over the variable X_4. This requires time O(k). We then compute the outer sum, the total time being $O(k^2)$ time. In this example, the time savings are modest but as the number of variables grows, factorization structure can reduce the complexity exponentially.

Message Passing algorithms exploit such a structure of the problem, where a global function can be expressed as a product of local functions depending on a subset of variables. Typically, in a CSP only few of the variables are involved in any constraint, which implies that the corresponding factor functions involve a small number of variables. In general, Message Passing algorithms take the factor graph representation of the problem as input and return marginals over the variable nodes. They allow the nodes to communicate their local state by sending messages over the edges. The marginal computed for a node can be seen as a function of the messages received by that node. Just as in our example above, these algorithms are designed to use the distributive property of the sum and product operations to minimize the time complexity.

In the next chapter, we will describe in detail one such message passing algorithm known as Belief Propagation. This algorithm is proven to compute exact solution marginals for tree factor graphs in *linear* time.

Chapter 3

Belief Propagation

3.1 Introduction

Belief propagation (BP) [4] is a message passing algorithm proposed by Judea Pearl in 1982, for performing inference on graphical models, such as Bayesian Networks and Markov Random fields. It is inherently a Bayesian procedure, which calculates the marginal distribution for each unobserved node, conditioned on the observed nodes. BP was supposed to work only for tree-like graphs but it has demonstrated empirical success in networks with cycles such as error-correcting codes. It has also been shown to converge to a stationary point of an approximate free energy, known as the Bethe free energy [5] in statistical physics.

Belief propagation works by sending messages along the edges of the factor graph. Recall that a variable node i in a factor graph is associated with a random variable X_i, which can take a value from its state space $\mathcal{X}_i$. A belief of a variable node i, denoted by $b_i(x_i)$ represents the likeness of random variable X_i to take value $x_i \in \mathcal{X}_i$. In BP, the beliefs are calculated as a function of messages received by the nodes. It turns out that the beliefs are equal to the marginal probabilities for tree-like graphs.

In the next section we explain the basic intuition behind the beliefs and derive the message passing rules for the Belief Propagation algorithm.

3.2 Beliefs and Message Equations

The belief $b_i(x_i)$ of a node i in its value x_i can be understood as how likely the node i thinks it should be in state x_i. A node bases its decision on messages from its neighbors. A message $m_{a\to i}(x_i)$ from node a to i can be interpreted as how likely node a thinks that node i will be in the corresponding state. The belief for a variable node, is thus, proportional to the product of messages from the neighboring factor nodes.

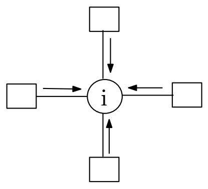

FIGURE 3.1: Diagrammatic representation for the belief of a variable node i.

$$b_i(x_i) \propto \prod_{a \in N(i)} m_{a \to i}(x_i) \tag{3.1}$$

where the symbol "$\propto$"stands for the proportionality between the L.H.S and R.H.S. such that $\sum_{x_i} b_i(x_i) = 1$.

We can write (3.1) as

$$b_i(x_i) = \frac{1}{Z} \prod_{a \in N(i)} m_{a \to i}(x_i) \tag{3.2}$$

where Z is the normalization constant.

Now consider the setting as in Figure 3.2. The belief of factor node A can be considered as the joint belief of its neighboring variable nodes. To compute the same, we consider a new node which contains the factor node A and all its neighboring variable nodes (labelled as L). Now $b_A(X_A) = b_L(x_L)$. Here, $X_A = \{x_j : j \in N(A)\}$ is the configuration of the variables participating in A. If $\mathcal{X}_l$ is the domain space associated with node L,

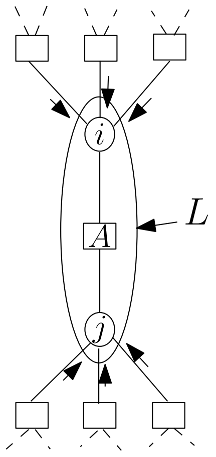

FIGURE 3.2: Diagrammatic representation for belief of factor node A.

then

$$\mathcal{X}_L = \{(x_i, x_j) \mid f_A(x_i, x_j) = 1, x_i \in \mathcal{X}_i, x_j \in \mathcal{X}_j\}.$$

By the same argument as above, the belief of node L should be proportional to the product of messages coming into it, i.e.,

$$\begin{aligned} b_L(x_L) =& \frac{1}{Z} \prod_{a \in N(L)} m_{a \to L}(x_L) \\ =& \frac{1}{Z} f_A(x_i, x_j) \prod_{a \in N(i)} m_{a \to i}(x_i) \prod_{a \in N(j)} m_{a \to i}(x_j) \\ =& \frac{1}{Z} f_A(X_A) \prod_{k \in N(A)} \prod_{b \in N(k) \setminus A} m_{b \to k}(x_k). \end{aligned} \tag{3.3}$$

To extract the belief corresponding to a single node, say i, from the joint belief we impose the following constraint on the beliefs. We will refer to this as the *marginalization* condition.

$$b_i(x_i) = \sum_{X_A \setminus x_i} b_A(X_A). \tag{3.4}$$

The notation $\sum\limits_{X_A \setminus x_i}$ means that the sum is taken over all configurations X_A, in which variable i is fixed to x_i. We will use this notation throughout the thesis.

Using Equations (3.2), (3.3) and (3.4) we obtain

$$\prod_{a \in N(i)} m_{a \to i}(x_i) = \sum_{X_A \setminus x_i} f_A(X_A) \prod_{k \in N(A)} \prod_{b \in N(k) \setminus A} m_{b \to k}(x_k). \tag{3.5}$$

Dividing both sides by $\prod\limits_{a \in N(i) \setminus A} m_{a \to i}(x_i)$ we obtain

$$m_{A \to i}(x_i) = \sum_{X_A \setminus x_i} f_A(X_A) \prod_{k \in N(A) \setminus i} \prod_{b \in N(k) \setminus A} m_{b \to k}(x_k). \tag{3.6}$$

Also (3.6) can be written as

$$m_{A \to i}(x_i) = \sum_{X_A \setminus x_i} f_A(X_A) \prod_{k \in N(A) \setminus i} m_{k \to A}(x_k) \tag{3.7}$$

where

$$m_{k \to A}(x_k) = \prod_{b \in N(k) \setminus a} m_{b \to k}(x_k), \tag{3.8}$$

which is infact a closed form expression for a message from a variable node to the factor node. Intuitively, a message $m_{i \to a}(x_i)$ from a variable node i to factor node a tells us

how likely i will take state x_i in a factor graph without factor node a.
In the next section, we will use these message update rules construct the BP algorithm for tree factor graphs. We will also prove that it computes the exact marginal probabilities.

3.3 Belief Propagation for Trees

We use the message rules in Equations (3.7) and (3.8) to derive the BP algorithm to compute exact marginals for tree factor graphs. We introduce the notation $M_{i\to j}(x)$, which corresponds to the message from node i to node j and define it as follows

$$M_{i\to j}(x) = \begin{cases} m_{i\to j}(x), & \text{if } i \text{ is a variable node and } x \in \mathcal{X}_i, \\ m_{i\to j}(x), & \text{if } i \text{ is a factor node and } x \in \mathcal{X}_j, \\ \phi & \text{otherwise} \end{cases} . \tag{3.9}$$

We now present the *BPTree* algorithm, which takes as input a tree factor graph $T = (V, E)$ with $V = I \cup A$, where I is the set of N variable nodes and A is the set of M factor nodes. Also, T is rooted at a node $r \in I$. There are $|E| = N + M - 1$ edges in T. The output of the algorithm is the set of beliefs for the variable nodes that, as we will prove, coincide with the respective marginal probabilities.

We start by interpreting the belief equations for the tree factor graph and derive an alternative form of the belief as required in the proof. Consider the following message from a factor node a to a variable node i in *BPTree*. Recall that $N(a)$ denotes the neighbors of node a and $f_a(X_a)$ is the function corresponding to the factor node a.

$$m_{a\to i}(x_i) = \sum_{X_a \setminus x_i} f_a(X_a) \prod_{j \in N(a)\setminus i} m_{j\to a}(x_j) \tag{3.10}$$

where

$$m_{j\to a}(x_j) = \prod_{b \in N(j)\setminus a} m_{b\to j}(x_j). \tag{3.11}$$

Also

$$b_j^a(x_j) \propto \prod_{b \in N(j)\setminus a} m_{b\to j}(x_j) = m_{j\to a}(x_j), \tag{3.12}$$

Algorithm 1 $BPTree(T)$

```
Input: Rooted tree factor graph, $T = (V, E)$
Output: Set of beliefs for the variable nodes
 1: $H \leftarrow height(T)$
 2: Let $I_\ell$ denotes the set of nodes at depth $\ell$
 3: for $\ell = H$ to 0 do
 4:     for all $i \in I_\ell$ and $j \in N(i) \cap I_{\ell-1}$ do
 5:         Compute messages $M_{i \to j}(x)$
 6:     end for
 7: end for
 8: for $\ell = 0$ to $H$ do
 9:     for all $i \in I_\ell$ and $j \in N(i) \cap I_{\ell+1}$ do
10:         Compute messages $M_{i \to j}(x)$
11:     end for
12: end for
13: for each $i \in I$ and for all $x_i \in \mathcal{X}_i$ do
14:     $S_i(x_i) \leftarrow \prod_{a \in N(i)} M_{a \to i}(x_i)$
15: end for
16: for each $i \in I$ do
17:     $Z_i \leftarrow \sum_{x_i} S_i(x_i)$
18:     for each $x_i \in \mathcal{X}_i$ do
19:         $b_i(x_i) \leftarrow \frac{S_i(x_i)}{Z_i}$
20:     end for
21: end for
22: return Set of beliefs, $\{b_i(x_i), \forall i \in I\}$
```

where $b_j^a(x_j)$ is the belief of variable j in a factor graph in which the factor a is removed. Now the belief of node i can be written as

$$\begin{aligned} b_i(x_i) &\propto \prod_{a \in N(i)} m_{a \to i}(x_i) \\ &\propto \prod_{a \in N(i)} \sum_{X_a \setminus x_i} f_a(X_a) \prod_{j \in N(a) \setminus i} b_j^a(x_j). \end{aligned} \tag{3.13}$$

The following theorem establishes the convergence and exactness of $BPTree$.

Theorem 3.1. *Let T be a tree factor graph rooted at a variable node r. The set of beliefs returned by $BPTree(T)$ are the exact marginal probabilities of the variable nodes of T. Also the computation time is linear in the number of variables.*

Proof. We divide the proof in two parts. In Part 1, we show by induction on height of the tree and considering messages only from leaves to root, that the belief at the root is exactly the marginal at the root. In Part 2, we prove the correctness of marginals at all other nodes by induction again on height and considering the messages now in the downwards direction, i.e. from root to leaves. Let H be the height of the tree.

Part 1. For $H = 1$, the marginal of the root is trivially equal to the product of its neighboring factors. For a variable node i at height h, let $T_{i,h}$ denote the tree rooted at node i as shown in Figure 3.3. We denote the set of variable nodes in $T_{i,h}$ by $I(T_{i,h})$ and the set of factor nodes by $A(T_{i,h})$. Let X_{T_i} denote a configuration of variables in $I(T_{i,h})$. Assume that for any variable node i at height $h < H$, the belief of i computed as a product of messages received from its neighbors in $T_{i,h}$ is equal to its true marginal in $T_{i,h}$. Now the belief of the root node r by Equation (3.13) is given by

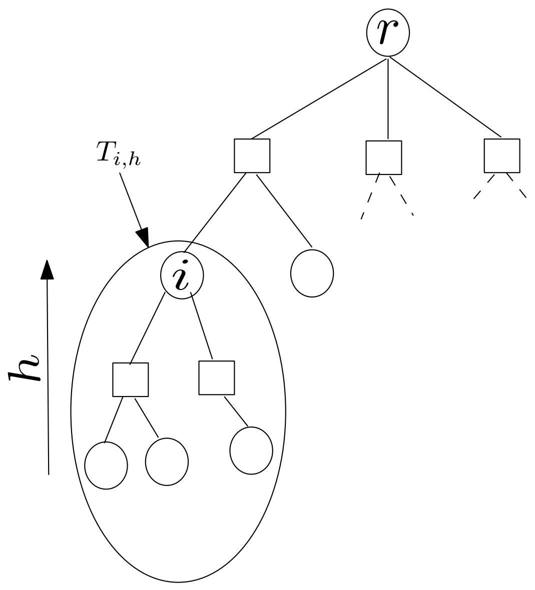

FIGURE 3.3: A rooted tree factor graph.

$$b_r(x_r) \propto \prod_{a\in N(r)} \sum_{X_a \backslash x_r} f_a(X_a) \prod_{i\in N(a)\backslash r} b_i^a(x_i) \tag{3.14}$$

where $b_i^a(x_i)$ is the belief of node i in $T_{i,H-2}$. Let $P_i^a(x_i)$ is the marginal for node i in $T_{i,H-2}$. By applying the inductive hypothesis, we have

$$\begin{aligned} b_r(x_r) &\propto \prod_{a\in N(r)} \sum_{X_a \backslash x_r} f_a(X_a) \prod_{i\in N(a)\backslash r} P_i^a(x_i) \\ &\propto \prod_{a\in N(r)} \sum_{X_a \backslash x_r} f_a(X_a) \prod_{i\in N(a)\backslash r} \sum_{X_{T_i}\backslash x_i} \prod_{b\in A(T_{i,H-2})} f_b(X_b) \\ &\propto \prod_{a\in N(r)} \sum_{X_a \backslash x_r} f_a(X_a) \sum_{\substack{X_{T_i}\backslash x_i, \\ i\in N(a)\backslash r}} \prod_{i} \prod_{b\in A(T_{i,H-2})} f_b(X_b) \\ &\propto \prod_{a\in N(r)} \sum_{X_a \backslash x_r} \sum_{\substack{X_{T_i}\backslash x_i, \\ i\in N(a)\backslash r}} f_a(X_a) \prod_{i} \prod_{b\in A(T_{i,H-2})} f_b(X_b) \end{aligned} \tag{3.15}$$

$$
\begin{aligned}
&\propto \sum_{X \setminus x_r} \prod_{a \in N(r)} f_a(X_a) \prod_{i \in N(a) \setminus r} \prod_{b \in A(T_{i,H-2})} f_b(X_b) \\
&\propto \sum_{X \setminus x_r} \prod_{a \in A(T)} f_a(X_a)
\end{aligned}
\tag{3.16}
$$

which is by definition the exact marginal.

Part 2. Now the messages stabilize and do not change for the upward direction. When the root has received correct messages from all of its neighbors, it can compute the outgoing messages. Assume that with the given set of upward and downward messages, the belief of any node j at distance $d <= H - 1$ from the root is the exact marginal in original graph T. We now prove that, the belief calculated for any leaf is equal to its true marginal in G. Indeed, the belief for leaf node l is given by

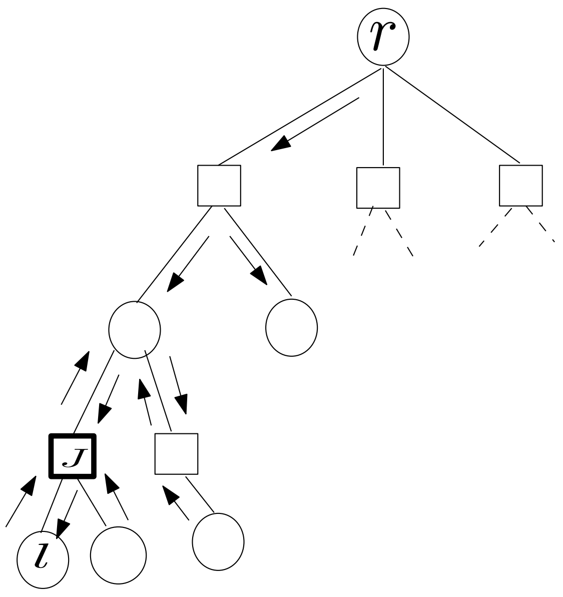

FIGURE 3.4: BP messages in a tree.

$$
\begin{aligned}
b_l(x_l) &\propto \prod_{a \in N(l)} m_{a \to l}(x_l) \\
&\propto m_{J \to l}(x_l) \\
&\propto \sum_{X_J \setminus x_l} f_J(X_J) \sum_{X \setminus X_J} \prod_{b \in A(T) \setminus J} f_b(X_b) \\
&\propto \sum_{X \setminus x_l} \prod_{b \in A(T)} f_b(X_b)
\end{aligned}
\tag{3.17}
$$

which is the exact marginal by definition. We will now show that the running time of *BPTree* is linear in the number of variables.

Running time. We observe that in Algorithm $BPTree(T)$, two messages are passed along each of the edge of T. Note that the number of edges in T is $N + M - 1$. By assuming that $M = \alpha N$, where α is some constant, we have $O(N)$ running time complexity for the computation of the messages. Also, the calculation of the marginals from the messages require $O(N)$ time. The rest of the steps take constant time. Thus the total running time for $BPTree$ is $O(N) + O(N) + O(1) = O(N)$. □

We have seen that in a tree, messages can be computed from leaves to root and once the root has received messages from all of its neighbors, it can send the message to each of its child nodes. This procedure continues until the leaves. Additionally we showed that, once a node has received messages from its neighbors, the computed belief is the exact marginal for that node. But such a sequential procedure does not hold for a general graph with cycles. We will now describe a more general version of the standard BP algorithm also called *loopy BP* which can run on general graphs.

3.4 Loopy Belief Propagation for General Graphs

In the previous section we have seen that BP gives exact marginals in case the input factor graph is a tree. The reason behind this exactness is attributed to the fact that in a tree, messages received by a node from its neighbors are independent. This is in general not true, because of the presence of cycles, which makes the neighbors of a node correlated and hence, the messages are no longer independent. Nevertheless, BP has been applied successfully in many applications such as computer vision and error correcting codes where the associated factor graphs have many cycles.

We now present BP algorithm for general factor graphs which has been designed assuming that the messages into a node are independent or weakly correlated. We first normalize the variable to factor messages so that all messages represent probability distributions. Let $\mu_{i\to a}(x_i)$ be the normalized message from variable node i to a.

$$\mu_{i\to a}(x_i) = Z_{i\to a} \prod_{b\in N(i)\setminus a} \mu_{b\to i}(x_i), \tag{3.18}$$

where $Z_{i\to a}$ is the normalization constant such that $\sum_{x_i} \mu_{i\to a}(x_i) = 1$. The clause to variable message is given by

$$\mu_{a\to i}(x_i) = \sum_{X_a\setminus x_i} f_a(X_a) \prod_{j\in N(a)\setminus i} \mu_{j\to a}(x_j). \tag{3.19}$$

The input to the algorithm, *BPGraph* is a factor graph $G = (V, E)$; a maximum number of iterations t_{max}; a requested precision ϵ. If *BPGraph* has converged after t_{max} iterations, it returns the set of beliefs for the variable nodes.

Algorithm 2 *BPGraph*(G)

```
1: For t=0, initialize messages $\mu_{a\to i} \in [0,1]$ along the edges of the factor graph
2: for $t = 1$ to $t = t_{max}$ do
3:     Sweep the set of edges in some order and update sequentially the messages on all edges of the graph, generating new set of messages $\mu^t_{a\to i}(x_i)$, using message update rules in Equations (3.18) and (3.19)
4:     if $|\mu^t_{a\to i}(x_i) - \mu^{t-1}_{a\to i}(x_i)| < \epsilon$ for each edge $\{a, i\}$ then
5:         break;
6:     end if
7: end for
8: if $t = t_{max}$ then
9:     return UNCONVERGED
10: else
11:     for $i \in I$ do
12:         Compute $b_i(x_i) \leftarrow \prod_{a\in N(i)} \mu^t_{a\to i}(x_i)$
13:     end for
14: end if
```

The correctness and convergence of loopy BP has been established for trees [5] and for Gaussian graphical models. We will discuss the known results about the same in the next chapter. For other cases, theoretical guarantees for the quality of the beliefs obtained is still lacking.

In the next section, we discuss the conditioning method proposed by Pearl to compute the exact marginals using standard Belief Propagation.

3.4.1 Conditioning/Clamping for Exact Marginals

Pearl [4] suggested a technique called cutset conditioning for exact inference in Bayesian Networks. The idea behind this method is to instantiate a minimum number of variables so that the resulting network is a tree and can be accurately solved. Also [6] proposes algorithms based on conditioning for exact and approximate inference in casual networks. We first formally define the conditioning/clamping procedure and show how the method of conditioning can be adopted in Belief Propagation.

We say that a factor graph $G = (V, E)$ is clamped if there exists at least one variable node $i \in I(G)$ which has been fixed or instantiated to one of its values $x_i \in \mathcal{X}_i$. In general, let $V_c(G)$ be the set of instantiated variables in G. For a configuration, $\sigma \in \{x_i : i \in V_c(G), x_i \in \mathcal{X}_i\}$ we create a clamped factor graph $\hat{G} = (\hat{V}, \hat{E})$ as follows

1. Remove all variable nodes $V_c(G)$, such that $\hat{V} = V - V_c(G)$.
2. Obtain the new edge set $\hat{E}$ by removing all the edges incident on the nodes in $V_c(G)$.
3. For each factor node $a \in A(G)$ such that $N(a) \cap V_c(G) \neq \phi$, create a node $\bar{a}$ such that $f_{\bar{a}}(X_{\bar{a}})$ is obtained from $f_a(X_a)$ by replacing each variable $i \in N(a) \cap V_c(G)$ by the constant x_i.

We illustrate the above procedure by taking the example of a SAT formula $F = (p \vee q \vee r) \wedge (\neg p \vee q \vee s)$. The corresponding factor graph G involves a cycle, as shown in Figure 3.5. The functions with respect to the factor nodes A and B are $f_A(p, q, r) = (p \vee q \vee r)$ and $f_B(p, q, s) = (\neg p \vee q \vee s)$ respectively. We now fix the variable p to take value 0. The corresponding clamped factor graph is shown in Figure 3.6. The factor functions corresponding to new factor nodes $\bar{A}$ and $\bar{B}$ are now $f_{\bar{A}}(0, q, r) = (0 \vee q \vee r) = (q \vee r)$ and $f_{\bar{B}}(0, q, s) = (1 \vee q \vee s) = 1$. In the same way, we can obtain the clamped factor graph corresponding to $p = 1$. For each of these clamped factor graphs, we can now

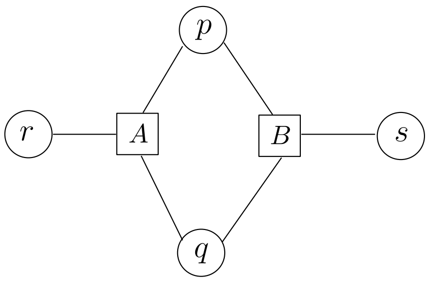

FIGURE 3.5: Factor graph representing the SAT formula $(p \vee q \vee r) \wedge (\neg p \vee q \vee s)$.

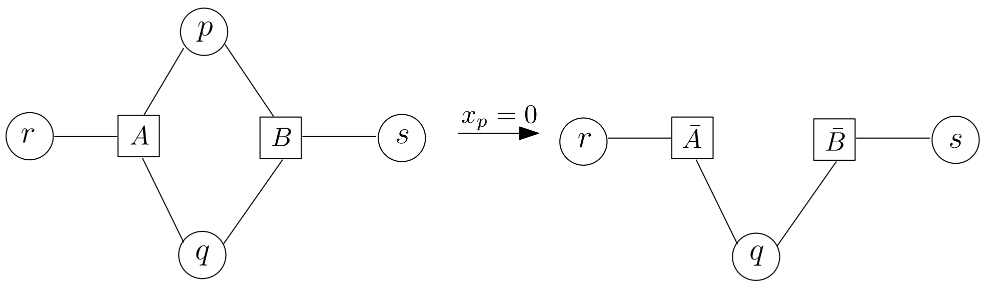

FIGURE 3.6: Diagrammatic representation of the clamping procedure.

run *BPTree* as they are now cycle free. But the marginals obtained will be conditioned on p assuming a value $x_p \in \mathcal{X}_p$. The true marginals can then be computed by summing conditioned marginals corresponding to each of the clamped factor graphs. This is best illustrated by the identity

$$Pr(X_i = x_i) = \sum_{\{x_p\}} Pr(X_i = x_i, X_p = x_p). \tag{3.20}$$

We now introduce some notations to formally present the conditioning algorithm. We denote by $V_c(G)$ the set of variables to be instantiated in factor graph G so that the corresponding clamped graph is a tree. We assume that this set is computed in prior using some efficient method. For each configuration σ of the variables in $V_c(G)$, we denote the corresponding clamped graph by G_σ. We denote by $b_i^\sigma(x_i)$, the belief of node i in G_σ. Recall that $I(G)$ represents the set of variable nodes in G.

Algorithm 3 *CondBP*(G, V_c)

```
Input: A factor graph, $G = (V, E), V_c(G)$
Output: Marginal for each of the variables
 1: if $G$ is a tree then
 2:   Call $BPTree(G)$
 3: else
 4:   for each $\sigma \in \{x_i : i \in V_c(G),\ x_i \in \mathcal{X}_i\}$ do
 5:     Call $BPTree(G_\sigma)$
 6:   end for
 7:   for each variable $i \in I(G)$ do
 8:     if $i \in V_c(G)$ then
 9:       $S_i(x_i) \leftarrow \{\sigma : i$ assumes value $x_i$ under $\sigma\}$
10:       $b_i(x_i) \leftarrow \sum_{\sigma \in S_i(x_i)} b_i^\sigma(x_i)$
11:     else
12:       $b_i(x_i) \leftarrow \sum_{\sigma} b_i^\sigma(x_i)$
13:     end if
14:   end for
15: end if
```

We note that the number of clamped graphs is exponential in the number of clamped variables. This implies that the number of terms under the summation in steps 10 and 12 in *CondBP* grows exponentially with the size of $V_c(G)$. But, we can exploit certain properties of the underlying graph structure to reduce the time complexity. We recall from Chapter 2 that a factor graph is a Markov Random field and it can be decomposed into conditionally independent parts. In the next section we use such properties of the factor graph to develop an exact algorithm for computing marginals.

3.4.2 Exact BP Algorithm for General Graphs

The conditioning method as given by Pearl requires that the nodes should be instantiated such that the connectivity of the original graph is preserved. This can be easily satisfied by clamping exactly one node from each cycle. This, in general will require exponential time to compute the marginals. For example consider the following factor graph, $G = (V, E)$ which is $2 \times N$ grid with pairwise constraints on $2N$ variables. The minimum size

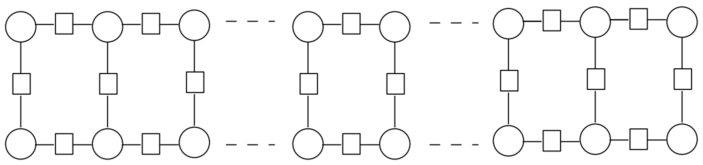

FIGURE 3.7: Factor graph of $2N$ variables and pairwise constraints.

of instantiated variable set $V_c(G)$ will be $N-2$. The running time for $CondBP(G, V_c(G))$ will be $\Omega(2^N)$.

We know that a factor graph follows the global Markov property. This guarantees that for the factor graph $G = (V, E)$, if there exists a triple (A, B, S) of disjoint subsets of V, such that S separates A from B in G then

$$X_A \perp\!\!\!\perp X_B | X_S$$

A triple (A, B, S) for the factor graph in Figure 3.7 is shown in Figure 3.8. Now if we

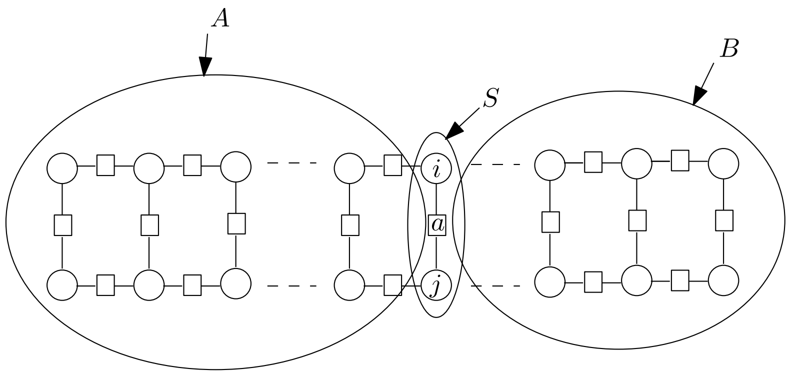

FIGURE 3.8: The cutset S.

instantiate variables i and j, one can obtain two independent clamped graphs as follows. We first remove the nodes in S to split the graph into two parts A and B. We will refer to such a set S as the *cut-set*. We call an instantiation $\{x_i, x_j\}$ of variables i and j *valid* if $f_a(x_i, x_j) = 1$. For all valid instantiations of variables i and j, we replace the variables i and j in the factor functions of A and B with the respective constants. We call the two clamped graphs obtained, as G_A and G_B. Assume that we can compute beliefs for variables in G_A and G_B, which we suppose are exact marginals. Let l be any node in the clamped graph G_A. We denote by $b_l^{G_A}(x_l)$ the belief of node l in G_A. Note that such a belief will be conditioned on the clamped variables. Let $Pr^{G_A}(x_k)$ denotes the marginal for the event "$X_k = x_k$" in G_A. The joint marginal for the events "$X_i = x_i$" and "$X_j = x_j$" in G_A can then be computed as

$$\begin{aligned} Pr^{G_A}(x_i, x_j) &= \sum_{x_l} Pr^{G_A}(x_l, x_i, x_j) \\ &= \sum_{x_l} b_l^{G_A}(x_l). \end{aligned} \tag{3.21}$$

Also

$$\begin{aligned} Pr^{G}(x_l, x_i, x_j) =& Pr^{G_A}(x_l, x_i, x_j) \cdot Pr^{G_B}(x_i, x_j) \\ =& Pr^{G_A}(x_l, x_i, x_j) \cdot Pr^{G_B}(x_i, x_j) \\ =& b_l^{G_A}(x_l) \cdot \sum_{x_k} b_k^{G_B}(x_k). \end{aligned} \tag{3.22}$$

where k is some variable node in G_B. We can now compute exact marginal for the event "$X_l = x_l$" by summing (3.22) over all valid configurations of i and j.

$$\begin{aligned} Pr^{G}(x_l) &= \sum_{\{x_i, x_j\}} Pr^{G}(x_l, x_i, x_j) \\ &= \sum_{\{x_i, x_j\}} \left(b_l^{G_A}(x_l) \cdot \sum_{x_k} b_k^{G_B}(x_k) \right). \end{aligned} \tag{3.23}$$

The marginals corresponding to clamped variables can be obtained as

$$\begin{aligned} Pr^{G}(x_i) &= \sum_{x_j} Pr^{G}(x_i, x_j) \\ &= \sum_{x_j} Pr^{G_A}(x_i, x_j) \cdot Pr^{G_B}(x_i, x_j) \\ &= \sum_{x_j} \sum_{x_l} b_l^{G_A}(x_l) \cdot \sum_{x_k} b_k^{G_B}(x_k) \end{aligned} \tag{3.24}$$

We showed that if exact marginals in two clamped subgraphs are known, we can combine the results to find marginals corresponding to the parent graph. Note that the two subparts can be solved independently. We can generalize such an approach by decomposing the graph recursively until we obtain tree structured components. We show the result of such a decomposition in case of a 2×5 grid in Figure 3.9, in the form of a tree. In general, if G is decomposed into k components $G_1, G_2, \ldots G_k$ by clamping variables in C, the exact marginals for each node in G are given by the following equations. We will denote by α_c the configuration of variables in C. For some $l \in I(G_i)$

$$Pr^G(x_l) = \sum_{\alpha_c} b_l^{G_i}(x_l) \cdot \prod_{j \in [K] \setminus i} \sum_{x_m} b_m^{G_j}(x_m) \tag{3.25}$$

Also, for any $p \in C$ we have

$$Pr^G(x_p) = \sum_{\alpha_c} \prod_{j \in [K]} \sum_{x_m} b_m^{G_j}(x_m) \tag{3.26}$$

Here $b_m^{G_j}(x_m)$ corresponds to the belief of some variable $m \in I(G_j)$.

We observe that corresponding to each node in this tree, we obtain clamped graphs

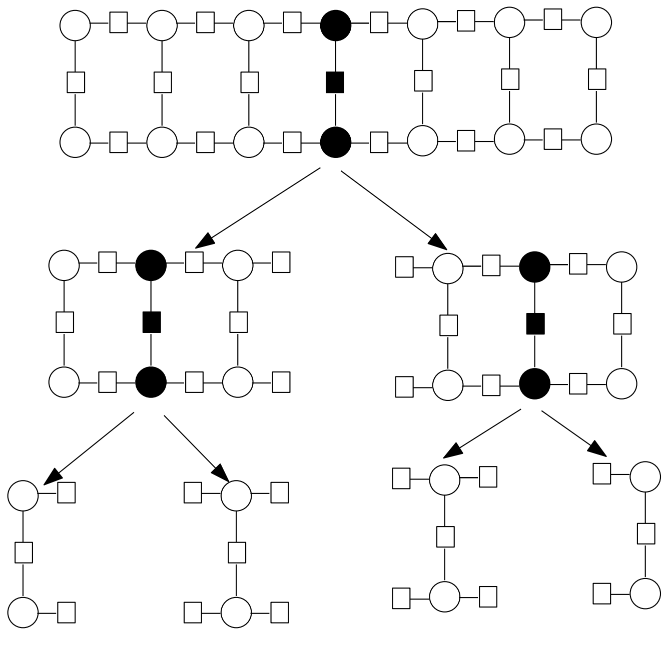

FIGURE 3.9: The decomposition of 2×5 grid.

equal to the number of *valid* instantiations of variables clamped in the parent node. We can then run $BPTree$ on each of these clamped subgraphs and obtain the marginals in the parent node (graph) by (3.23) and (3.24).

For a general $2 \times N$ grid, we note that the maximum height of the corresponding decomposition tree obtained is at most $\log N$ (if we always break the graph in the middle). As we choose two variables to clamp in each step, the total number of variables to be clamped till we reach the leaf is at most $2 \log N$. Now the leaves are tree structured and can be solved in linear time in the number of variable nodes. The cost for combining the marginals at the leaves (L_1 and L_2) into conditioned marginals at the parent node P will be $O(2^2(I(L_1) + I(L_2))) \leq O(2^2 \cdot N)$. The cost of extracting the true marginals by summing over all configurations of clamped variables equals $O(2^2 \cdot I(P) \cdot 2) \leq O(2^2 \cdot N)$. The total computations then required to compute marginals at the root by recursively computing computing marginals at each level will be $O(2^{2 \log N} N)$. Therefore, the total cost for computing exact marginals in $2 \times N$ grid is $O(N^3)$ in contrast to $\Omega(2^N)$ required in case of Pearl's conditioning method.

We will now formally describe the procedure of obtaining exact marginals in general graphs. First, we describe the *decompose*(G) algorithm, which takes the factor graph G as input and returns the corresponding decomposition tree. We assume that the cut-sets required for the decomposition have been pre-computed by some efficient procedure.

Algorithm 4 *decompose*(G)

```
if G is a tree then
  return G
else
  let $C \subseteq I(G)$ be some cutset
  let $G_1, G_2, \ldots, G_k$ be the components obtained by decomposing $G$ over the cutset
  return $(C, decompose(G_1), \ldots, decompose(G_k))$
end if
```

The output of *decompose*(G) can be seen as a tree of height H with the leaves as tree-structured factor graphs returned at the termination of the algorithm. Every other node at height h represents the cutset required in the $(H-h+1)^{th}$ step of the decomposition. Consider the factor graph G in Figure 3.10. The output of *decompose*(G) is shown in

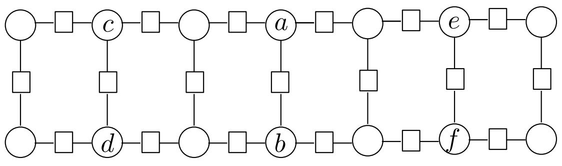

FIGURE 3.10: Factor graph.

Figure 3.11.

The second part of the procedure corresponds to the *Marginal* algorithm where we formalize the computations over the decomposition tree to find exact marginals in the original graph. We now introduce some data-structures and notations required for the algorithm. We use β_G, an $N \times D$ matrix where N is the number of variables in G and D is the largest domain size of all variables. We also assume that the domain space

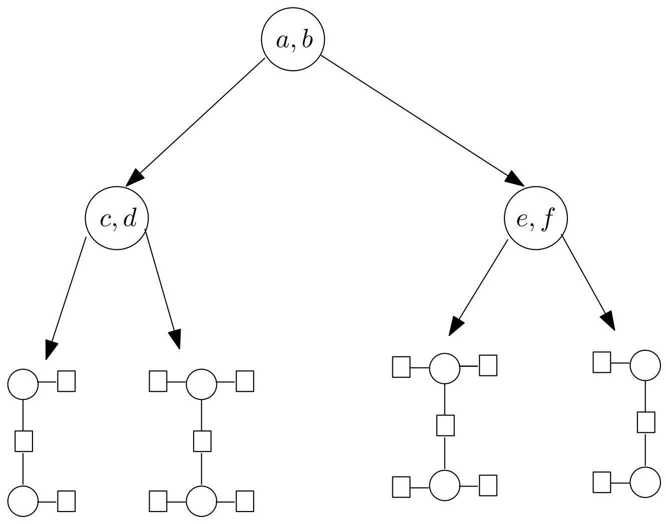

FIGURE 3.11: Diagrammatic representation of the output of decompose(G).

of all variables is discrete (the case for continuous domains can be handled too with some modifications) and finite. The entry $\beta_G[i][j]$ corresponds to the belief of node i corresponding to the event "$X_i = j$". Let C be the cut-set corresponding to G. We represent by $\tilde{G}_i$, the subgraph (i^{th} child node) of G obtained by clamping variables in C with respect to some configuration α_c. Then, $\beta_{\alpha_c,\tilde{G}_i}$ corresponds to the belief matrix for $\tilde{G}_i$. We now present the $Marginal(T, G)$ algorithm where G is a factor graph and T is the corresponding decomposition tree. In Theorem 3.2, we prove that the beliefs returned by $Marginal(T, G)$ are the exact marginals of variables in G.

Theorem 3.2. *The algorithm $Marginal(T, G)$ returns the exact marginal of a variables in G in time $O(2^M N^2)$ where M is the maximum number of variables clamped along a path from root to some leaf and N is the number of variables in G.*

Proof. We will prove the above theorem by induction on the height(H) of tree T obtained by the decomposition of G. For $H = 0$, G is a tree and exact marginals are obtained by $BPTree(G)$. The running time is $O(N)$.

Suppose that we can compute the exact marginals corresponding to the nodes at any height $h < H$. We further assume that for a node i at height h, the computation cost is $O(2^{M_i} N_i^2)$ where M_i is the maximum number of nodes clamped along any path from node i to some leaf. This means that for the clamped graph $\tilde{G}_i$, in which all variables from root to that node are clamped (to some configuration), the exact marginals can be computed in time $O(2^{M_i} N_i^2)$.

Now these marginals are combined to form the exact marginals for the root at height H by (3.25) and (3.26). The computation cost at the root can be divided into two parts. The first part is incurred in computing exact marginals at each of the child nodes for all

Algorithm 5 $Marginal(T, G)$

```
Input: Factor graph $G = (V, E)$ and $T = decompose(G)$
Output: Belief matrix, $\beta_G$
 1: Parse $T$ /*$T = (C, D_1, D_2, \cdots, D_k)$*/
 2: if k=0 then
 3:     Call BPTree(C)
 4: else
 5:     /*Clamp*/
 6:     for each possible assignment $\alpha_c$ of the variables in C do
 7:         for each $i = 1 \cdots k$ do
 8:             $\tilde{G}_i = G[D_i]$ /*clamped graph corresponding to $D_i$ and $\alpha_c$*/
 9:             $\beta_{\alpha_c,\tilde{G}_i} \leftarrow Marginal(D_i, \tilde{G}_i)$
10:         end for
11:     end for
12:     for each $i = 1 \cdots k$ do
13:         /*Joint marginal for the clamped variables in each $\tilde{G}_i$ (3.21) */
14:         $P_{\alpha_c,\tilde{G}_i}(C) \leftarrow \sum_{j=1}^{D} \beta_{\alpha_c,\tilde{G}_i}[m][j]$ /* $m$ is some variable node in $\tilde{G}_i$*/
15:     end for
16:     for each node $l \in G$ do
17:         for each $i = 1 \cdots k$ do
18:             if $l \in I(\tilde{G}_i)$ then
19:                 $\beta_{\alpha_c,G}[l] \leftarrow \beta_{\alpha_c,\tilde{G}_i}[l] \left( \prod_{j\in[k]\setminus i} P_{\alpha_c,\tilde{G}_j}(C) \right)$
20:                 $\beta_G[l] \leftarrow \sum_{\alpha_c} \beta_{\alpha_c,G}[l]$
21:             end if
22:         end for
23:         if $l \in C$ then
24:             $\beta_{\alpha_c,G}[l] \leftarrow \prod_i P_{\alpha_c,\tilde{G}_i}(C)$
25:             $\beta_G[l] \leftarrow \sum_{\alpha_c \setminus x_l} \beta_{\alpha_c,G}[l]$
26:         end if
27:     end for
28:     return $\beta_G$
29: end if
```

clampings (possible configurations of the clamped variables in the root) which is given by

$$A_1 = 2^{C_r} \cdot (O(2^{M_1} N_1^1) + O(2^{M_2} N_2^2) + \ldots O(2^{M_k} N_k^2)).$$

Here, C_r is the number of clamped variables at the root and k is the total number of child nodes of the root. The other part of the cost is due to summing up the conditioned marginals (for all configurations of clamped variables) to compute the exact marginal for each node. It is given by

$$A_2 = 2^{C_r} \cdot N \cdot k. \tag{3.27}$$

Also $k < N$. By construction of the clamped graphs, we have $N = N_1 + \ldots + N_k + C_r$. Let $M = \max_i (C_r + M_i), \forall i \in [k]$. The total cost is then given by

$$\begin{aligned} A =& A_1 + A_2 \\ =& 2^{C_r} \cdot (O(2^{M_1} N_1^2) + O(2^{M_2} N_2^2) + \ldots O(2^{M_k} N_k^2)) + 2^C \cdot N \cdot k \\ =& O(2^M N^2). \end{aligned} \tag{3.28}$$

□

We observe that the efficiency of the above method depends on the number of variables clamped along the height of the decomposition tree of the graph. The other issue is the efficient computation of the cut-sets required to build the decomposition tree. To apply this method, we need to look at certain classes of factor graphs which not only allow efficient decomposition but also assure that the cut-set at any time will not be too large. Motivated by the results of [7], we now use the above theorem to bound the running time for minor-free graphs.

Corollary 3.3. *Let G be a factor graph with N variable nodes that does not contain K_h as a minor. Then the marginal probabilities for G can be computed in time $O(2^{h^{3/2}\sqrt{N}} N^2)$.*

Proof. To compute marginal probabilities for G, we first build the decomposition tree T by the procedure *decompose*(G). We will first show that this can be done in polynomial time.

By definition, a **separation** of a graph G is a pair (A, B) of subsets of $V(G)$ with $A \cup B = V(G)$, such that no edge of G joins a vertex in $A - B$ to a vertex in $B - A$. Also, [7] proves that there is a separation (A, B) in G of order $\leq h^{\frac{3}{2}} N^{\frac{1}{2}}$ such that $A - B, B - A \leq \frac{2}{3} N$. Moreover this separation can be computed in polynomial time. Thus, we can decompose G using these separations as the cut-sets, in polynomial time. Also, the number of variable nodes in any clamped graph at any level in T can be at most $\frac{2}{3} n$, where n is the number of variables in the graph corresponding to the parent node. So the maximum number of variables clamped from the root to some leaf in T is given by

$$\begin{aligned} M &\leq h^{3/2} \cdot \left(\sqrt{N} + \sqrt{\frac{2}{3}N} + \sqrt{\left(\frac{2}{3}\right)^2 N} + \dots \right) \\ &= O(h^{3/2}\sqrt{N}). \end{aligned} \tag{3.29}$$

We can now run procedure $Marginal(T, G)$ to obtain exact marginal probabilities in G. By Theorem 3.2, the time required to compute exact marginals in G is $O(2^{h^{3/2}\sqrt{N}} N^2)$.

$\square$

In the next chapter, we will again go back to Loopy Belief Propagation and discuss its convergence and accuracy for some Graphical Models.

Chapter 4

Properties of Loopy Belief Propagation

4.1 Introduction

In this chapter, we demonstrate the connection between Belief Propagation and Bethe approximation of statistical physics. We also explain how this knowledge is used in deriving Generalized Belief Propagation algorithms. Such an approach is the work of [5]. We also discuss the known results for the convergence and correctness of loopy BP for graphical models with a single loop [8] and Gaussian graphical models of arbitrary topology [9].

CSPs appear in a large spectrum of scientific disciplines especially in physics .We can view CSP in a physical system as follows. Suppose we have a system of N particles. Each of the particle i can take be in one of the possible states x_i. The state of the system is described as $\mathbf{x} = (x_1, x_2, \cdots, x_n)$. We will denote by S the set of all possible states of the system. The system in state x has energy $E(\mathbf{x})$ associated with it. So, we need to find the state of the system in which it is the most stable i.e. it has minimum energy. In thermal equilibrium, the probability of a state is given by *Boltzmann′s Law*

$$p(\mathbf{x}) = \frac{1}{Z(T)} e^{-E(\mathbf{x})/T}. \tag{4.1}$$

Here, T is the temperature (an auxiliary parameter) and $Z(T)$ is the normalization constant known as the partition function

$$Z(T) = \sum_{\mathbf{x} \in S} e^{-E(\mathbf{x})/T}. \tag{4.2}$$

We now try to relate the above with the CSP in a non physical system. For our factor graph representation, the probability distribution is given by

$$p(\mathbf{x}) = \frac{1}{Z}\prod_{a=1}^{M} f_a(X_a). \tag{4.3}$$

From (4.1) and (4.3) and by setting $T = 1$ we get the energy of our system

$$E(\mathbf{x}) = -\sum_{a=1}^{M} \ln(f_a(X_a)). \tag{4.4}$$

Let $b(\mathbf{x})$ be the approximate joint probability distribution (trial distribution) for our factor graph. We further assume that this trial distribution corresponds to the one computed by the Belief Propagation Algorithm. We now try to see how accurate is the distribution $b(\mathbf{x})$ with respect to the required distribution $p(\mathbf{x})$. The differences between two probability distributions can be measured by means of Kullback-Leibler distance [10]. By definition, the Kullback-Leibler distance between $p(\mathbf{x})$ and $b(\mathbf{x})$ is given by

$$D(b\|p) = \sum_{\mathbf{x}} b(\mathbf{x}) \ln \frac{b(\mathbf{x})}{p(\mathbf{x})}. \tag{4.5}$$

Using (4.1) for $p(\mathbf{x})$ (with $T = 1$) we get

$$D(b\|p) = \sum_{\mathbf{x}} b(\mathbf{x})E(\mathbf{x}) + \sum_{\mathbf{x}} b(\mathbf{x}) \ln b(\mathbf{x}) + \ln Z. \tag{4.6}$$

We also know that Kullback-Leibler distance is always non-negative and is zero if and only if the two probability functions $b(\mathbf{x})$ and $p(\mathbf{x})$ are equal. So our approximate probability function $b(\mathbf{x})$ will be equal to the exact probability function when $D(b\|p)$ achieves its minimum value of zero.

In the next section we will derive conditions for the minimization of the distance between the two probability distributions for a tree-factor graph. It turns out that the results correspond to the standard BP algorithm.

4.2 Fixed Points of BP

In this section, we show that the BP message equations (in case of a tree factor graph) satisfy the stationary conditions derived for the minimization of $D(b\|p)$. As in literature, we denote the quantity

$$G(b(\mathbf{x})) = \sum_{\mathbf{x}} b(\mathbf{x})E(\mathbf{x}) + \sum_{\mathbf{x}} b(\mathbf{x}) \ln b(\mathbf{x}) = U(b(\mathbf{x})) - H(b(\mathbf{x})) \tag{4.7}$$

by Gibbs free energy. Here,

$$U(b(\mathbf{x})) = \sum_{\mathbf{x}} b(\mathbf{x}) E(\mathbf{x}) \tag{4.8}$$

is the average energy and

$$H(b(\mathbf{x})) = \sum_{\mathbf{x}} b(\mathbf{x}) \ln b(\mathbf{x}) \tag{4.9}$$

is the entropy of the system. Also, $D(b\|p)$ will be zero if $G(b(\mathbf{x}))$ achieves its minimum value of $F = -\ln Z$ which is also known as "Helmholtz free energy". We take $b(\mathbf{x})$ to be the joint probability distribution given by the BP algorithm. We can now formulate our problem as a minimization problem as follows

$$\min_{\mathbf{x}} G(b(\mathbf{x})) \tag{4.10}$$

such that

$$\sum_{x_i} b_i(x_i) = 1 \quad \forall i, \tag{4.11}$$

$$\sum_{X_a} b_a(X_a) = 1 \quad \forall a, \tag{4.12}$$

$$b_i(x_i) = \sum_{X_a \setminus x_i} b_a(X_a) \quad \forall i, \forall a \ s.t \ a \in N(i). \tag{4.13}$$

Recall that the Equations (4.11),(4.12) state the normalization conditions while (3.4) gives the marginalization condition for the beliefs. For a factor graph without cycles we can show that the exact joint distribution $p(\mathbf{x})$ can be factorized [10] as

$$p(\mathbf{x}) = \frac{\prod_a p_a(X_a)}{\prod_i (p_i(x_i))^{d_i - 1}}, \tag{4.14}$$

where d_i equals the number of neighbors of node i.

Now if beliefs are equal to the exact marginals, we can simplify $G(b(\mathbf{x}))$ as follows

$$\begin{aligned}
G(b(\mathbf{x})) &= \sum_{\mathbf{x}} b(\mathbf{x}) E(\mathbf{x}) + \sum_{\mathbf{x}} b(\mathbf{x}) \ln b(\mathbf{x}) \\
&= -\sum_{\mathbf{x}} b(\mathbf{x}) \ln \prod_a f_a(X_a) + \sum_{\mathbf{x}} b(\mathbf{x}) \ln \left(\prod_a b_a(X_a) \prod_i (b_i(x_i))^{1-d_i} \right) \\
&= -\left(\sum_{\mathbf{x}} b(\mathbf{x}) \sum_a \ln f_a(X_a) \right) + \sum_{\mathbf{x}} b(\mathbf{x}) \left(\sum_a \ln b_a(X_a) + \sum_i \ln(b_i(x_i))^{1-d_i} \right) \\
&= -\left(\sum_a \sum_{\mathbf{x}} b(\mathbf{x}) \ln f_a(X_a) \right) + \sum_a \sum_{\mathbf{x}} b(\mathbf{x}) \ln b_a(X_a) + \sum_i \sum_{\mathbf{x}} b(\mathbf{x}) \ln(b_i(x_i))^{1-d_i}
\end{aligned}$$

$$\begin{aligned}
&= -\sum_a \sum_{X_a} \sum_{\mathbf{x}\backslash X_a} b(\mathbf{x}) \ln f_a(X_a) + \sum_a \sum_{X_a} \sum_{\mathbf{x}\backslash X_a} b(\mathbf{x}) \ln b_a(X_a) \\
&\quad + \sum_i \sum_{x_i} \sum_{\mathbf{x}\backslash x_i} b(\mathbf{x}) \Big((1-d_i) \ln b_i(x_i) \Big) \\
&= -\sum_a \sum_{X_a} b_a(X_a) \ln f_a(X_a) + \sum_a \sum_{X_a} b_a(X_a) \ln b_a(X_a) \\
&\quad - \sum_i (d_i - 1) \sum_{x_i} b_i(x_i) \ln b_i(x_i).
\end{aligned} \tag{4.15}$$

We here point out that such a simplification of Gibbs free energy is known as Bethe approximation in literature.

Now we solve (4.10) by forming a Langragian L as follows

$$\begin{aligned}
L =& G(b(\mathbf{x})) + \sum_i \gamma_i (\sum_{x_i} b_i(x_i) - 1) + \sum_a \gamma_a (1 - \sum_{x_a} b_a(x_a)) \\
&+ \sum_a \sum_{i \in N(a)} \sum_{x_i} \lambda_{a_i,x_i} \Big\{ \sum_{X_a \backslash x_i} b_i(x_i) - b_a(X_a) \Big\}.
\end{aligned} \tag{4.16}$$

Here, γ_i and γ_a enforce the normalization constraints over the beliefs of variable and factor nodes respectively and λ_{a_i,x_i} enforce the marginalization constraint for factor a and variable i. Now setting $\frac{\partial L}{\partial b_i(x_i)} = 0$ gives

$$\begin{aligned}
(d_i - 1)(1 + \ln b_i(x_i)) =& \gamma_i + \sum_{a \in N(i)} \lambda_{a_i,x_i} \\
\Rightarrow b_i(x_i) =& \frac{1}{d_i - 1} \exp\Big\{ \sum_{a \in N(i)} \lambda_{a_i,x_i} \Big\} \cdot \exp\Big\{ \frac{\gamma_i - 1}{d_i - 1} \Big\} \\
\Rightarrow b_i(x_i) \propto& \Big(\prod_{a \in N(i)} \exp\{\lambda_{a_i,x_i}\}^{\frac{1}{d_i - 1}} \Big).
\end{aligned}$$

The equation $\frac{\partial L}{\partial b_a(X_a)} = 0$ gives

$$\begin{aligned}
\ln b_a(X_a) =& \ln f_a(X_a) + \sum_{i \in N(a)} \lambda_{a_i,x_i} + \gamma_a - 1 \\
\Rightarrow b_a(X_a) =& f_a(X_a) \exp\Big\{ \sum_{i \in N(a)} \lambda_{a_i,x_i} + \gamma_a - 1 \Big\} \\
\Rightarrow b_a(X_a) \propto& f_a(X_a) \cdot \exp\Big\{ \sum_{i \in N(a)} \lambda_{a_i,x_i} \Big\}.
\end{aligned}$$

Now if we set

$$\lambda_{a_i,x_i} = \ln \prod_{b \in N(i) \backslash a} m_{b \to i}(x_i),$$

we get

$$b_a(X_a) \propto f_a(X_a) \cdot \prod_{i \in N(a)} \prod_{b \in N(i) \backslash a} m_{b \to i}(x_i) \tag{4.17}$$

and

$$b_i(x_i) \propto \left(\prod_{a \in N(i)} \prod_{b \in N(i) \backslash a} m_{b \to i}(x_i) \right)^{\frac{1}{d_i - 1}} \tag{4.18}$$

$$\Rightarrow b_i(x_i) \propto \prod_{a \in N(i)} m_{a \to i}(x_i). \tag{4.19}$$

We note that in case the factor graph has no cycles, minimization of the free energy is exact and correspondingly the equivalent BP equations give us the correct result. In case of a general graph, factorization of the joint distribution as in Equation (4.14) does not hold and is only an approximation. In the next section we discuss how the method of free energy minimizations has been applied to derive Generalized Belief Propagation algorithms.

4.3 Generalized Belief Propagation

We can now extend the above method to general graphs. We decompose the given factor graph into regions in such a way that the free energy of the whole system is equal to the sum of the free energies of the regions. Such a decomposition of the graph results in a region graph. For such a graph, Gibbs free energy can be approximated as follows

$$G_R(\{b_r\}) = \sum_{r \in R} c_r G_r(b_r). \tag{4.20}$$

Here R denotes the set of valid regions, c_r is the counting number for region r such that for every factor node a and every variable node i, the following holds

$$\sum_{r \in R} \delta(a \in A_r) c_r = \sum_{r \in R} \delta(i \in V_r) c_r = 1. \tag{4.21}$$

Here, A_r and V_r denote the set of the factor and variable nodes respectively in region r. The indicator function $\delta(z)$ is equal to 1 if the condition z is satisfied and equal to 0 otherwise.

We note that region based average energy $U_R(\{b_r\})$ can be evaluated as

$$U_R(\{b_r\}) = \sum_{r \in R} c_r U_r(b_r). \tag{4.22}$$

From Equations (4.8) and (4.4) we have

$$U_r(b_r) = \sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) \sum_{a \in A_r} \ln f_a(X_a). \tag{4.23}$$

Therefore,

$$U_R(\{b_r\}) = \sum_{r \in R} c_r \sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) \sum_{a \in A_r} \ln f_a(X_a). \tag{4.24}$$

The region-based entropy is given by

$$H_R(\{b_r\}) = \sum_{r \in R} c_r H_r(b_r) \tag{4.25}$$

$$= - \sum_{r \in R} c_r \sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) \ln b_r(\mathbf{x}_r). \tag{4.26}$$

Note that the region based entropy will be approximate even if the beliefs are exactly equal to the corresponding true marginal probabilities unless the region graph is a tree. The same argument (as for Equation (4.14)) about the factorization of exact marginals holds here.
The approximate graph free energy can thus be written as

$$G_R(\{b_r\}) = \sum_{r \in R} c_r \sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) \sum_{a \in A_r} \ln f_a(X_a) + \sum_{r \in R} c_r \sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) \ln b_r(\mathbf{x}_r). \tag{4.27}$$

Generalized belief propagation (GBP) algorithms can now be constructed whose message update equations correspond to the stationary points of the free energy associated with the region graph. Such a method of deriving GBP algorithms is termed as *region graph method*. The junction graph method [11] and cluster variation method [12] can be considered as special cases of the region graph method. The reported experimental results for the GBP algorithms suggest that they are more accurate than standard BP for some problems like error-correcting codes.

In the next section, we discuss BP as an inference tool which computes the posterior distribution of the variables when some evidence (data) is combined with the known prior distribution. We give a brief overview of the known results of the correctness of loppy BP when the variables involved obey Gaussian distributions.

4.4 Correctness of Loopy BP for some Graphical Models

Loopy BP has been applied to a wide variety of applications such as error correcting codes, speech recognition and medical diagnosis. In such problems, a prior probability distribution is assumed over the set of random variables which is represented by a graphical model. Values for some variables are provided. These are called the observed variables or the evidence nodes in the respective graphical model. The task is to infer the values of the unobserved variables given the observed ones. This task can be further reduced to finding the correct posteriors for the unobserved variables in presence of new evidence.

Single loop network. We consider here the pairwise undirected graphical model (Markov network) with the joint distribution given by the product of potentials over the edges of the graph. We show an example of such a graphical model, $G = (V, E)$ in Figure 4.1 where the gray nodes correspond to the observed variables while the others are unobserved ones. The posterior marginal probability for node A_1 given the evidence (due to

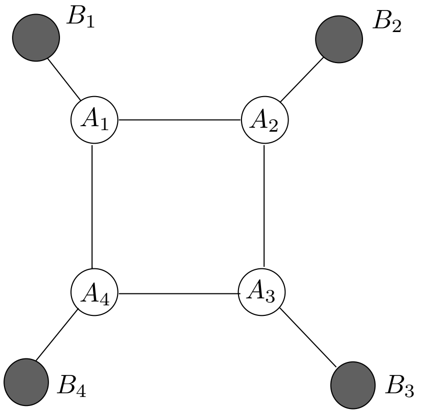

FIGURE 4.1: A Markov network.

the observed nodes) is given by

$$Pr(A_1 = a_1 | B_1 = b_1, B_2 = b_2, B_3 = b_3, B_4 = b_4) = \frac{1}{Z} \sum_{a_2, a_3, a_4} \prod_{e \in E} \Psi_e(X_e), \tag{4.28}$$

where $\Psi_e(X_e)$ is the potential defined over each edge e. Here X_e is the configuration of variables connected by edge e. We can easily convert the above model into a factor graph by introducing a factor node corresponding to each edge as shown in Figure 4.4. The factor function will then correspond to the respective edge potential. Now by message update rules for Belief Propagation, the message from F_4 to A_1 is given by

$$m_{F_4 \to A_1} = \sum_{a_4} \Psi_{A_4, A_1}(a_1, a_4) \prod_{f \in N(A_4)} m_{f \to A_4}. \tag{4.29}$$

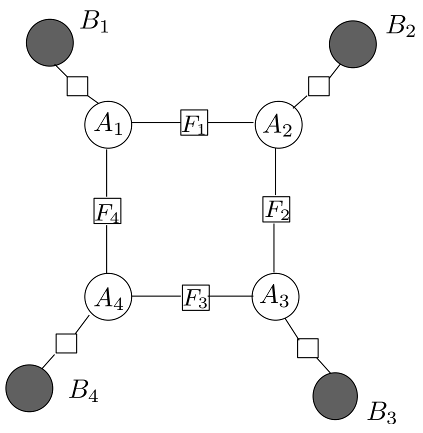

FIGURE 4.2: Factor Graph corresponding to a pairwise Markov network.

We note that message from A_4 to A_1 in Figure 4.1 will correspond to message from F_4 to A_1 in the corresponding factor graph in Figure 4.4. Let $\mathbf{V}_{A_4 \to A_1}$ be the message vector from A_4 to A_1. The component $\mathbf{V}_{A_4 \to A_1}(i)$ will correspond to message sent from A_4 to A_1 for i^{th} value of A_1. Let $M_{A_1 A_4}$ correspond to transition matrix corresponding to edge (A_1, A_4) defined as

$$M_{A_1 A_4}(i,j) = \Psi(A_1 = i, A_4 = j).$$

Then,

$$\mathbf{V}_{A_4 \to A_1} = \alpha M_{A_1 A_4} \left(\bigodot_{A \in N(A_4) \setminus A_1} \mathbf{V}_{A \to A_4} \right), \tag{4.30}$$

where $\bigodot$ corresponds to component-wise multiplication of vectors i.e for vectors $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$, we have

$$\mathbf{z} = \mathbf{x} \odot \mathbf{y} \leftrightarrow \mathbf{z}(i) = \mathbf{x}(i)\mathbf{y}(i).$$

Also, $\alpha \mathbf{x}$ denotes normalizing the components of $\mathbf{x}$ so that they sum to 1.
Let $\mathbf{b}_{A_1}$ corresponds to the belief vector for A_1 such that the component $\mathbf{b}_{A_1}(k)$ corresponds to belief of node A for its k^{th} value. Then

$$\mathbf{b}_{A_1} = \alpha \left(\bigodot_{A \in N(A_1)} \mathbf{V}_{A \to A_1} \right).$$

Note that the message from an observed node to an unobserved node will be a vector of constant messages for each value of the unobserved node. For the i^{th} observed node, we define a diagonal matrix D_i such that the elements of D_i are the constant messages sent from the observed node B_i to the unobserved node A_i. Also for any vector $\mathbf{x}$, we have $D_i \mathbf{x} = \mathbf{V}_{B_i \to A_i} \odot \mathbf{x}$.

Assuming that the messages are updated sequentially, we have

$$\begin{aligned}
\mathbf{V}_{A_4\to A_1} &= \alpha M_{A_1A_4}\left(\bigodot_{A\in N(A_4)\setminus A_1}\mathbf{V}_{A\to A_4}\right)\\
&= \alpha M_{A_1A_4}\left(\mathbf{V}_{B_4\to A_4}\odot\mathbf{V}_{A_3\to A_4}\right)\\
&= \alpha M_{A_1A_4}D_4M_{A_3A_4}\left(\mathbf{V}_{B_3\to A_3}\odot\mathbf{V}_{A_2\to A_3}\right)\\
&= \alpha M_{A_1A_4}D_4M_{A_3A_4}D_3M_{A_2A_3}\left(\mathbf{V}_{B_2\to A_2}\odot\mathbf{V}_{A_1\to A_2}\right)\\
&= \alpha M_{A_1A_4}D_4M_{A_3A_4}D_3M_{A_2A_3}D_2M_{A_1A_2}\left(\mathbf{V}_{B_1\to A_1}\odot\mathbf{V}_{A_4\to A_1}\right)\\
&= \alpha M_{A_1A_4}D_4M_{A_3A_4}D_3M_{A_2A_3}D_2M_{A_1A_2}D_1\mathbf{V}_{A_4\to A_1}.
\end{aligned} \tag{4.31}$$

Let $C_{A_1} = M_{A_1A_4}D_4M_{A_3A_4}D_3M_{A_2A_3}D_2M_{A_1A_2}D_1$, then for any time t, we obtain

$$\mathbf{V}^{t+4}_{A_4\to A_1} = \alpha\ C_{A_1}\mathbf{V}^{t}_{A_4\to A_1}. \tag{4.32}$$

In general, for a cycle of N nodes we obtain

$$\mathbf{V}^{t+N}_{A_N\to A_1} = \alpha\ C_{A_1}\mathbf{V}^{t}_{A_N\to A_1}. \tag{4.33}$$

If all elements of C_{A_1} are non-zero, [8] proves the following statements

1. $\mathbf{V}_{A_N\to A_1}$ converges to the principle eigenvector of C_{A_1}.
2. $\mathbf{V}_{A_2\to A_1}$ converges to the principle eigenvector of $D_1^{-1}C_{A_1}D_1$.
3. The convergence rate of the messages is governed by the ratio of the largest eigenvalue of C_{A_1} to the second largest eigenvalue.
4. The diagonal elements of C_{A_1} give the correct posteriors: $P_{A_1}(i) = \alpha\ C_{A_1}(i,i)$.
5. The steady-state belief $\mathbf{b}_{A_1}$ is related to the correct posterior marginal $\mathbf{P}_{A_1}$ by: $\mathbf{b}_{A_1} = \beta\ \mathbf{P}_{A_1} + (1-\beta)\mathbf{Q}_{A_1}$ where β is the ratio of the largest eigenvalue of C_{A_1} to the sum of all eigenvalues and $\mathbf{Q}_{A_1}$ depends on the eigenvectors of C_{A_1}.

In the following, we consider MRFs of arbitrary topology but with Gaussian distribution.

Gaussian Markov Random Fields. A Gaussian MRF (GMRF) is an MRF in which the joint distribution is Gaussian. In a GMRF, let $\mathbf{y} = \{y_1,\ldots,y_d\}$ be observed variables such that each y_i is connected to the unobserved node x_i. The corresponding joint distribution $Pr(\mathbf{x},\mathbf{y})$ is given by

$$Pr(\mathbf{x},\mathbf{y}) = \frac{1}{Z}\exp\left\{-\frac{1}{2}(\mathbf{x}^T,\mathbf{y}^T)\begin{pmatrix}V_{xx} & V_{xy}\\ V_{yx} & V_{yy}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}\right\}, \tag{4.34}$$

where Z is the normalization constant. The problem here is to compute each posterior mean and variance of a hidden node x_i, given evidence (data) $\mathbf{y}$. The conditional mean vector $\mu_{\mathbf{x}|\mathbf{y}}$ and covariance matrix $\Sigma_{\mathbf{x}|\mathbf{y}}$ are computed by solving

$$V_{xx}\mu_{\mathbf{x}|\mathbf{y}} = - V_{xy}\mathbf{y}, \tag{4.35}$$

$$\Sigma_{\mathbf{x}|\mathbf{y}} = V_{xx}^{-1}. \tag{4.36}$$

For large-scale Gaussian graphical models, direct computations as in the above equations become computationally intractable. Loopy Belief Propagation (LBP) can be employed over such GMRFs. In such a case, the messages and beliefs are all Gaussian and the updates can be written directly in terms of the means and covariance matrices. Each node sends and receives a mean vector and covariance matrix to and from each neighbor. The beliefs at a node are computed by combining the means and covariances of all the incoming messages. It has been shown that for Gaussian LBP, if LBP converges, then it computes the exact posterior means but incorrect variances. We refer [9] for proofs and further discussion.

In the next chapter we will describe a relatively new and exciting technique known as Survey Propagation which has been designed to solve random satisfiability problems. Survey Propagation is a message passing algorithm which was originally derived from cavity method in statistical physics but has been discovered to be a special form of Belief Propagation.

Chapter 5

Survey Propagation

5.1 Introduction

In this chapter, we present another message passing algorithm known as Survey Propagation (SP). The SP algorithm [13, 14] has been shown to find satisfying assignments for highly dense and large random $K-SAT$ formulas, i.e. random formulas with high clause-variable ratio and large number of variables, n. The computational cost has been estimated to be roughly scaling as $n \log n$ [14]. We show that SP can be derived from BP with an extended variable space and a relaxed notion of satisfiability. Though the SP and BP equivalence has been shown in [15, 16] in different forms, we contribute by giving a simplified interpretation of the same.

Throughout this chapter, we focus on $K-SAT$ formulas which we explain below. Given a formula F in conjunctive normal form (CNF), the satisfiability (SAT) problem asks whether there exists a truth assignment under which F evaluates to true. $K-SAT$ refers to the SAT problem in which all clauses contain K literals. $K-SAT$ was the first known NP-complete problem for $K >= 3$, as proved by Stephen Cook [17] in 1971. This means that no known algorithm can solve the worst case instances of the problem in polynomial time. Since, in real world applications, worst cases may not be relevant, the typical model studied is the random $K-SAT$. In this model, we fix the total number of clauses to be αn where n is the number of variables and α is a positive constant. Each clause is generated by choosing K variables randomly and negating each one randomly and independently.

Message passing algorithms like BP can be used to find a satisfying assignment for a CNF formula F as follows. The uniform distribution over the satisfying assignments in F is a Markov random field represented by a factor graph. Thus, Belief Propagation

Algorithm can be used to estimate the marginal probabilities of variables in this factor graph. Suppose the beliefs returned for a variable to be in 1 and 0 state are p_1 and p_0 respectively. We now choose the variable with the greatest bias, i.e., with maximum $|p_1 - p_0|$ and set it to 1 if $p_1 > p_0$ and 0 otherwise. We remove the variable and the satisfied constraints from the problem, and repeat the process on the reduced problem. This strategy of assigning variables in steps is known as decimation. It is known that a decimation procedure using the marginals returned by BP can find satisfying assignments to random 3-SAT when $\alpha <= 3.92$. The validity of BP relies on the assumption that messages into a node from its neighbors are independent. To understand this, we consider factor graph for a random $K - SAT$ formula. Suppose that a factor node a wants to send a message to a variable node i. Now the minimal distance between two of the variables $x_j,\ j \in N(a)\backslash i$ is typically of the order $\log N$ for a random factor graph. Thus the factor graphs for random $K - SAT$ behave locally as trees and the variables $x_j,\ j \in N(a)\backslash i$ can be considered weakly correlated. This might explain the convergence of BP for random 3-SAT problems with $\alpha < 3.9$. But in case $\alpha > 3.9$, the failure of BP approximations indicates the existence of long range correlations inspite of the local tree-like structure of the factor graph. In other words, we can no longer approximate $N(a)\backslash i$ to be independent even when they are far apart in the factor graph. Statistical physicists attribute these long range correlations to the geometry of the solution space of highly dense formulas. With the introduction of some basic background, we explain their hypothesis about this solution space geometry in the next section.

5.2 Solution Space of HARD-SAT Problems

We start by introducing the notion of a solution cluster. We denote by $S(F)$ the set of satisfying assignments of a CNF formula F. A solution graph for F is an undirected graph, where nodes correspond to solutions and are neighbors if they differ on the assignment of exactly one variable. In other words, the assignments $\sigma, \tau \in S(F)$ are adjacent if they have the Hamming distance equal to 1. A **cluster** is then a connected component of $S(F)$. Figure 5.1 illustrates how the structure of the solution space of random 3-SAT evolves as the density of a random formula increases. We summarize the physicists hypothesis of the solution space of random SAT through the following conjectures.

1. There exists a SAT-UNSAT phase transition in the solution space of these problems. More specifically, for clause-variable ratio greater than its critical value α_s, solutions cease to exist.

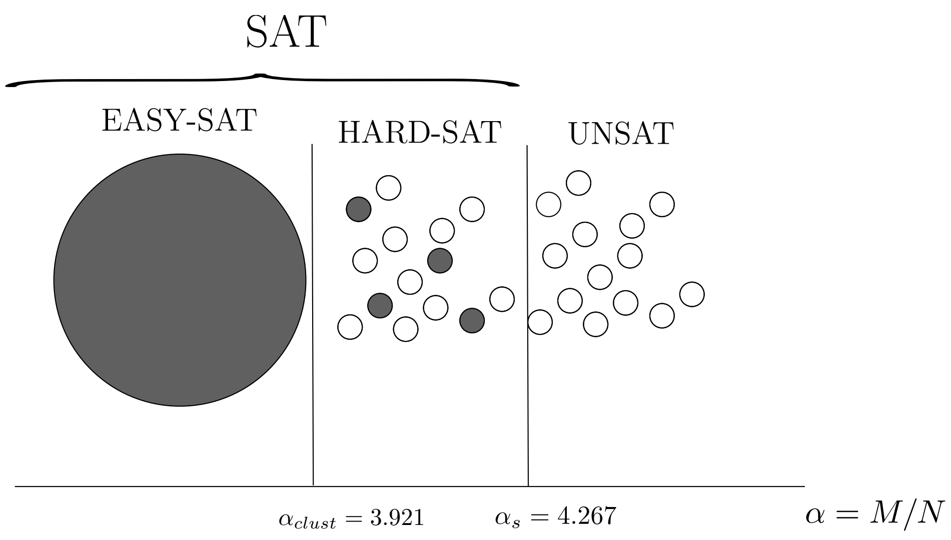

FIGURE 5.1: Diagrammatic representation of the solution space of random 3-SAT with respect to α. Satisfying clusters are represented by gray and the unsatisfying by white.

2. Also there exists another threshold α_{clust}, which divides the SAT region into two separate regions, called EASY-SAT and HARD-SAT. In the EASY-SAT region, solutions are thought to be close together such that one can reach from one solution to another with a number of single-bit flips. More specifically, each pair of satisfying assignments are close together in Hamming distance. In this region, local search algorithms and other simple heuristics can easily find a solution. In the HARD-SAT region, the solution space breaks up into many smaller clusters. Each cluster contains a set of satisfying assignments close together in Hamming distance. Also, the satisfying assignments in different clusters are far apart i.e. one cannot reach from a satisfying assignment in one cluster to the one in another without changing a large number of variables.

3. In the HARD-SAT region, there exists an exponentially large number of *metastable* clusters, i.e. clusters containing almost satisfying assignments (the assignments which violate a few clauses). Also, these almost satisfying solutions violate exactly the same number of clauses. The number of solution clusters are exponential in number of variables N, and the number of metastable clusters too are exponential in N but have a larger growth rate. This suggests that the simpler algorithms like BP get trapped within these most numerous metastable clusters. Moreover, most of the variables in a cluster are frozen, i.e., they take the same value in all the satisfying assignments of the cluster.

To describe the Survey Propagation algorithm, which is known empirically to find solutions in the HARD-SAT region, we explain further in detail in the next section, the physical picture of the clustering of configurations.

5.3 Solutions Clusters and Covers

Let γ be a cluster in the solution space of a CNF formula F. We denote by $S(\gamma)$, the satisfying assignments in γ. A variable x can be thought of being in 3 possible states, either (i) it is frozen to 0, i.e. $x = 0$ in all $\sigma \in S(\gamma)$ or (ii) it is frozen to 1 in $S(\gamma)$, or (iii) it switches between 0 and 1, i.e. it takes both of its values in $S(F)$. To account for the latter case, we extend the domain of x to include a third state "$*$" which is also known as the joker state. The above description attributes to each cluster, a single string in $\{0, 1, *\}^n$. We will call such a string over an extended variable space $\{0, 1, *\}$, as a **generalized assignment**. In a generalized assignment, a variable can be termed as constrained or unconstrained by the following definition.

Definition 5.1. Given a string $\mathbf{y} \in \{0, 1, *\}^n$, we will say that variable y_i is *free* or *unconstrained* in $\mathbf{y}$ if in every clause c containing y_i or $\neg y_i$, at least one of the other literals in c is assigned true or $*$. A variable is constrained if it uniquely satisfies some clause c , i.e. all other literals in c are assigned false.

A generalized assignment $\hat{\sigma} \in \{0, 1, *\}^n$ is called a **cover** [18] of F if

1. every clause of F has one satisfying literal or at least two $*$, and
2. every unconstrained variable in σ is assigned $*$.

Informally, a generalized assignment $\hat{\sigma}$ is a cover if under $\hat{\sigma}$, either each clause in F is satisfied or it leaves enough variables undecided, which can be set later on to generate a satisfying truth assignment. It is interesting to note that each satisfying assignment in F can be reduced to a unique cover by the coarsening procedure [16] as follows.

1. Given a satisfying assignment, σ, change one of its unconstrained variable to $*$.
2. Repeat until there are no unconstrained variables.

To understand the above procedure, consider the formula $F = (x \vee \neg y) \wedge (y \vee \neg z)$ and its satisfying assignment (000). Now, x is the only unconstrained variable under this assignment. Setting x to $*$ gives us the generalized assignment $(*00)$. Now under this new assignment, y is not constrained as it is no longer the unique satisfying literal for any clause. We thus obtain $(**0)$. The variable z ceases to be the unique satisfying literal for the second clause because of the previous step. We can now change it to $*$. The resulting generalized assignment is now $(***)$ which cannot be further reduced. This corresponds to a cover of the formula F.

It is now easy to see that all assignments within a cluster will correspond to a unique cover, also known as *core* [16] of the cluster.

5.4 The SP Algorithm

SP is a powerful technique for solving satisfiability problems, derived from the rather complex physics methods, notably the cavity method developed for the study of spin glasses. We refer to [19] for the original derivation of the SP equations by the *replica symmetric* method in statistical physics. Here, we will describe the SP algorithm and show that for tree-structured formulas, SP computes marginals over covers of the formula.

We start by describing the notations required for the SP algorithm. The basic input to the SP algorithm is a factor graph for a SAT (random) instance. For each variable node i in the factor graph, we denote the set of factor nodes in which the variable appears positively by $N_+(i)$, and by $N_-(i)$ the set consisting of factor nodes in which i appears negated. Here $N(i) = N_+(i) \cup N_-(i)$. Let $N_a^s(i)$ be the set of neighbors of i except a which tend to satisfy a, i.e., i occurs with the same polarity in any $b \in N_a^s(i)$ as in a. Also $N_a^u(i)$ represents the set of neighbors which are unsatisfying with respect to a.

SP like BP works by passing messages along the edges of the factor graph. A message

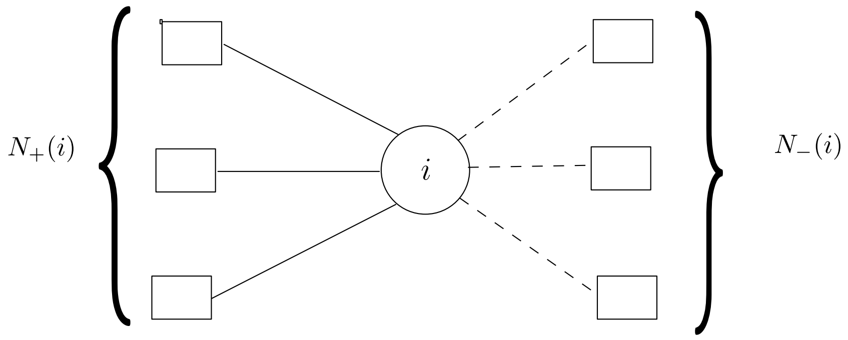

FIGURE 5.2: A part of a factor graph showing variable node i and its neighboring factor nodes.

from a factor node a to variable node i, known as a *survey*, is a real number $\eta_{a\to i} \in [0, 1]$. Each variable i can be in either of the three states in $\{0, 1, *\}$. A variable takes value 0 or 1 if it is constrained by a clause, i.e., if it satisfies uniquely at least one of the clauses and a joker state otherwise. Each variable node i sends a vector of three real numbers, $\Pi_{i\to a} = \{\Pi^u_{i\to a}, \Pi^s_{i\to a}, \Pi^*_{i\to a}\}$ to each of its neighboring factor node a. The survey $\eta_{a\to i}$

over edge $\{a, i\}$ is given by

$$\eta_{a\to i} = \prod_{j\in N(a)\setminus i} \frac{\Pi^u_{j\to a}}{\Pi^u_{j\to a} + \Pi^s_{j\to a} + \Pi^*_{j\to a}}. \tag{5.1}$$

The survey $\eta_{a\to i}$ is interpreted as a probability that a warning is sent from a factor node a to a variable node i [14]. A warning from a factor node a to a variable node i can be understood as a signal that a requires i to take a value that satisfies a which is due to the fact that all variables in $N(a) \setminus i$ have been assigned values that unsatisfy a.

The variable node i sends a message to factor node a corresponding to each of the three symbols $\{u, s, *\}$. The symbol "s" corresponds to its state which is satisfying for a, "u" stands for the state unsatisfying for a. It sends the symbol "$*$" when it is in the undecided or joker state. The corresponding SP equations are

$$\Pi^u_{i\to a} = \prod_{\beta\in N^s_a(i)} (1-\eta_{\beta\to i})\left[1 - \prod_{\beta\in N^u_a(i)} (1-\eta_{\beta\to i})\right], \tag{5.2}$$

$$\Pi^s_{i\to a} = \prod_{\beta\in N^u_a(i)} (1-\eta_{\beta\to i})\left[1 - \prod_{\beta\in N^s_a(i)} (1-\eta_{\beta\to i})\right], \tag{5.3}$$

$$\Pi^*_{i\to a} = \prod_{\beta\in N(i)\setminus a} (1-\eta_{\beta\to i}). \tag{5.4}$$

We will now derive the SP equations for tree-structured factor graphs. The justification for general graphs with cycles suffer from the similar problem as in BP, i.e, the incoming messages to a node can not be considered as independent. The remarkable performance of SP on hard SAT problems suggests the weak correlation among the input messages (to a node). The underlying reason for the same is still unknown.

5.4.1 Deriving SP Equations for Trees

Message Passing algorithms on trees can be thought of as dynamic programming solutions, where a node combines the results (messages) from the neighboring sub-parts of the problem and pass on to the next neighbor which repeats the process. In this section, we show that the messages in BP (for tree-structured SAT formulas) can be interpreted as combinatorial objects counting the number of solutions for some sub-part of the problem. We further show how this counting procedure can be extended to counting *covers* for the CNF formula. We then prove that the modified message update rules correspond to the SP update equations.

In the last paragraph, we claimed that a BP message counts the number of solutions for some sub-part of the problem i.e for some sub-formula of the original formula. To

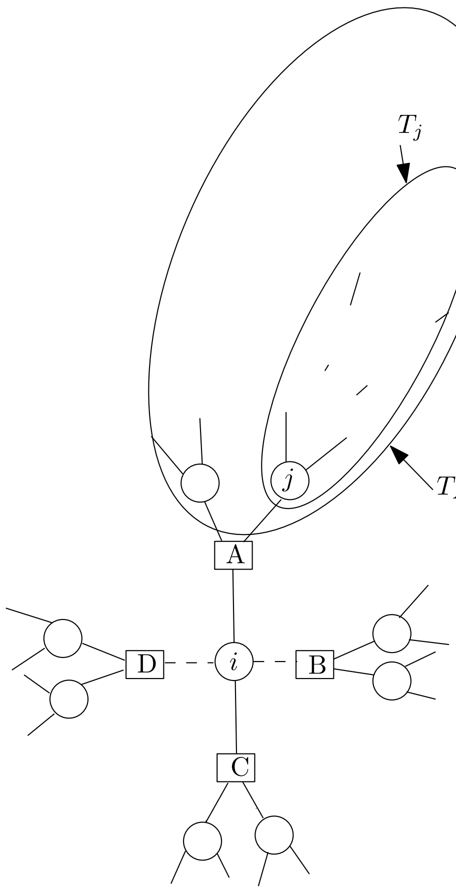

FIGURE 5.3: A part of tree-factor graph for a SAT formula.

understand this, consider a variable j in a formula F and suppose that it wants to send the message to its neighboring factor node a about its value x_j. We construct a sub-formula f as follows. We pick variable j and all clauses in which j participates except the clause a. We further include the variables participating in the included clauses and so on. Now the message sent from j to a is the number of solutions of f in which j assumes the value x_j.

To understand a clause to variable message, we assume that the factor node a has only two neighbors. Now suppose that the node a wants to send a message to its other neighbor i about its value x_i. We extend the sub-formula f to include the clause a and the variable i. The message that the factor node a sends to i counts the number of solutions (with i assuming the value x_i) for the extended sub-formula.

For SP, we will derive the corresponding message rules according to which a message counts the number of covers instead of solutions. The new beliefs will then correspond to the cover marginals instead of solution marginals.

We will now formally present the derivation of the SP equations from BP message update rules. We start by introducing some notations. We consider the tree factor graph $T = (V, E)$ corresponding to a SAT formula F. Figure 5.3 shows a part of T with variable node i and its four neighboring factor nodes. We denote by $I(T)$ and $A(T)$, the set of variable and factor nodes (each representing a clause) respectively in T. Recall that the dashed edge between node i and B signifies that the variable i is negated in the clause corresponding to factor node B.

Variable to clause messages. Consider the message from variable node j to factor node A. We denote the part of factor graph rooted at j and separated from rest of graph by A (shown in Figure 5.3) by T_j. Also $X_{T_j} = \{x_i : i \in I(T_j)\}$ represents a configuration of variables in T_j.

$$\begin{aligned} m_{j\to A}(x_j) &= \prod_{a\in N(j)\setminus A} m_{a\to j}(x_j) && (5.5)\\ &= \sum_{X_{T_j}\setminus x_j} \prod_{a\in A(T_j)} f_a(X_a), && (5.6)\end{aligned}$$

Note that the product term corresponding to each configuration X_{T_j} will be 1 if all the clauses in T_j are satisfied and 0 otherwise. The sum operation will in turn count the number of the non-zero terms of the product. Let $S(T_j)$ denotes the set of satisfying assignments in T_j in which j assumes the fixed value x_j. Then the variable to clause message $m_{j\to A}(x_j)$ determines $|S(T_j)|$.

We now modify the problem of counting satisfying assignments into that of counting the covers. Let $x^s_{a,j}/x^u_{a,j}$ denote the satisfying/unsatisfying value of j with respect to a, for example, $x^s_{A,i} = 1$ (with respect to Figure 5.3). Also a variable is assigned "$*$" if it is not constrained by any of the clauses. We now define our new message, $M_{j\to A}(x_j)$ such that it counts the covers of T_j, in which j assumes the value x_j.

Consider the message $M_{j\to A}(x^s_{A,j})$. This message is supposed to count the covers in T_j such that j assumes the value $x^s_{A,j}$ under the corresponding generalized assignment, say σ_g. This further implies that there exists at least one clause $c \in N^u_A(j)$ such that j is the unique satisfying variable for c under σ_g. Additionally, it is unconstrained by all clauses in $N^u_A(j)$.

$$\begin{aligned} M_{j\to A}(x^s_{j,A}) &= \prod_{a\in N(j)\setminus A}\Big(M_{a\to j}(x^s_{j,A}) + M_{a\to j}(*)\Big) - \prod_{a\in N(j)\setminus A} M_{a\to j}(*)\\ &= \prod_{a\in N^s_A(j)}\Big(M_{a\to j}(x^s_{j,a}) + M_{a\to j}(*)\Big)\prod_{a\in N^u_A(j)}\Big(M_{a\to j}(x^u_{j,a}) + M_{a\to j}(*)\Big)\\ &\quad - \prod_{a\in N^s_A(j)} M_{a\to j}(*)\prod_{a\in N^u_A(j)} M_{a\to j}(*) \end{aligned} \tag{5.7}$$

Similarly,

$$M_{j\to A}(x^u_{j,A}) = \prod_{a\in N^s_A(j)} \Big(M_{a\to j}(x^u_{j,a}) + M_{a\to j}(*)\Big) \prod_{a\in N^u_A(j)} \Big(M_{a\to j}(x^s_{j,a}) + M_{a\to j}(*)\Big) - \prod_{a\in N^s_A(j)} M_{a\to j}(*) \prod_{a\in N^u_A(j)} M_{a\to j}(*). \tag{5.8}$$

A node j will assume "$*$", if it is unconstrained in all clauses . Therefore,

$$M_{j\to A}(*) = \prod_{a\in N^s_A(j)} M_{a\to j}(*) \prod_{a\in N^u_A(j)} M_{a\to j}(*). \tag{5.9}$$

Clause to variable messages. Now consider the following clause to variable message as in BP

$$m_{A\to i}(x_i) = \sum_{X_A} f_A(X_A) \prod_{j\in N(A)\setminus i} m_{j\to A}(x_j). \tag{5.10}$$

We have seen that the variable to clause message $m_{j\to A}(x_j)$ counts the number of satisfying configurations for T_j, with j assuming value x_j. Therefore, the product $\prod_{j\in N(A)\setminus i} m_{j\to A}(x_j)$ counts the number of satisfying assignments in T_A for a particular configuration of variables in $N(A)\setminus i$. Now, let $S(T_A)$ denote the set of satisfying assignments in T_A for all possible values of variables in $N(A)\setminus i$. Let variable node i assume value x_i. We extend $S(T_A)$ to $\bar{S}(T_A)$ by including x_i such that the clause A is satisfied in $\bar{S}(T_A)$. The size of $\bar{S}(T_A)$ is precisely the message $m_{A\to i}(x_i)$.

We now give a parallel argument when the product of variable messages $\prod_{j\in N(A)\setminus i} M_{j\to A}(x_j)$ counts the number of covers of T_A. Let $\mathcal{C}(T_A)$ denote the set of covers of T_A. We extend $\mathcal{C}(T_A)$ to $\bar{\mathcal{C}}(T_A)$ by including x_i such that clause A is satisfied in $\bar{\mathcal{C}}(T_A)$. Let γ be any generalized assignment in $\bar{\mathcal{C}}(T_A)$. We now replace x_i in γ by "$*$" if clause A has been satisfied by variables other than i or there exists at least one more variable in A which has been assigned "$*$". We obtain $\bar{\mathcal{C}}_g(T_A)$ by repeating the above process for each $\gamma \in \bar{\mathcal{C}}(T_A)$. The message $M_{A\to i}(x_i)$ counts the number of generalized assignments in $\bar{\mathcal{C}}_g(T_A)$ with i taking value x_i.

By construction of $\bar{\mathcal{C}}_g(T_A)$, x_i equals $x^s_{A,i}$ in some $\gamma \in \bar{\mathcal{C}}_g(T_A)$ only when each of the variables $j \in N(A)\setminus i$ assumes the value $x^u_{A,j}$, i.e., all variables are unsatisfying except i. Mathematically, this can be stated as

$$M_{A\to i}(x^s_{i,A}) = \prod_{j\in N(A)\setminus i} M_{j\to A}(x^u_{j,A}). \tag{5.11}$$

Also, there exists no $\gamma \in \bar{\mathcal{C}}_g(T_A)$ such that $x_i = x^u_{A,i}$ under γ. Therefore,

$$M_{A\to i}(x^u_{i,A}) =0. \tag{5.12}$$

The message $M_{A\to i}(*)$ will count all the generalized assignments in $\bar{\mathcal{C}}_g(T_A)$ such that A is either satisfied without i or there exists atleast one other variable which is assigned $*$, i.e.,

$$M_{A\to i}(*) = \sum_{X_A \setminus x_i} F_A(X_A) \prod_{j\in N(A)\setminus i} M_{j\to A}(x_j), \tag{5.13}$$

where

$$F_a(X_a) = \begin{cases} 1, & \text{if there exists some variable } i \in N(a) \text{ such that } x_i = x^s_{i,A} \\ & \text{or there exists at least two variables } k, l \in N(a) \text{ with } x_k = * \text{ and } x_l = * \\ 0, & \text{otherwise} \end{cases} . \tag{5.14}$$

In other words, $M_{A\to i}(*)$ counts all generalized assignments in $\bar{\mathcal{C}}_g(T_A)$ except the ones in which each $j \in N(A) \setminus i$ assumes the value $x^u_{j,A}$. We can then write (5.13) as

$$M_{A\to i}(*) = \prod_{j\in N(A)\setminus i} \Big(M_{j\to A}(x^s_{j,A}) + M_{j\to A}(x^u_{j,A}) + M_{j\to A}(*) \Big) - \prod_{j\in N(A)\setminus i} M_{j\to A}(x^u_{j,A}). \tag{5.15}$$

The message update rules obtained by the extended cover counting procedure are summarized in Figure 5.4. We now derive the beliefs corresponding to the modified message update rules.

Beliefs. We consider again the example in Figure 5.3. The belief $B_i(x_i)$ now corresponds to the number of covers of F in which i assumes value x_i. We obtain such covers for i by combining appropriate covers from each of T_{N_i}. For example, consider $B_i(0)$. We need to count covers of F such that i is constrained to value 0. This means that atleast one clause from $N(i)$ constrains i to 0. Obviously, clauses A and C cannot constrain i to be in the 0 state and, either B or D or both should constrain i to 0. Therefore,

$$\begin{aligned} B_i(0) =& M_{A\to i}(*)M_{C\to i}(*)M_{B\to i}(0)M_{D\to i}(0) + M_{A\to i}(*)M_{C\to i}(*)M_{B\to i}(*)M_{D\to i}(0) \\ &+ M_{A\to i}(*)M_{C\to i}(*)M_{B\to i}(0)M_{D\to i}(*) \\ =& \prod_{a\in N_+(i)} M_{a\to i}(*) \left[\prod_{a\in N_-(i)} \Big(M_{a\to i}(x^s_{a,i}) + M_{a\to i}(*) \Big) - \prod_{a\in N_-(i)} M_{a\to i}(*) \right]. \end{aligned} \tag{5.16}$$

Variable to Clause Messages

$$M_{i\to a}(x^s_{a,i}) = \prod_{b\in N^s_a(i)} \big(M_{b\to i}(x^s_{b,i}) + M_{b\to i}(*)\big) \prod_{b\in N^u_a(i)} \big(M_{b\to i}(x^u_{b,i}) + M_{b\to i}(*)\big) - \prod_{b\in N^s_a(i)} M_{b\to i}(*) \prod_{b\in N^u_a(i)} M_{b\to j}(*)$$

$$M_{i\to a}(x^u_{i,a}) = \prod_{b\in N^s_a(j)} \big(M_{b\to i}(x^u_{b,i}) + M_{b\to i}(*)\big) \prod_{b\in N^u_a(i)} \big(M_{b\to i}(x^s_{a,i}) + M_{b\to j}(*)\big) - \prod_{b\in N^s_a(i)} M_{b\to i}(*) \prod_{b\in N^u_a(i)} M_{b\to i}(*)$$

$$M_{i\to a}(*) = \prod_{b\in N^s_a(i)} M_{b\to i}(*) \prod_{b\in N^u_a(i)} M_{b\to i}(*)$$

Clause to Variable Messages

$$M_{a\to i}(x^s_{a,i}) = \prod_{j\in N(a)\setminus i} M_{j\to a}(x^u_{a,j})$$

$$M_{a\to i}(x^u_{a,i}) = 0$$

$$M_{a\to i}(*) = \prod_{j\in N(a)\setminus i} \big(M_{j\to a}(x^s_{a,j}) + M_{j\to a}(x^u_{a,j}) + M_{j\to a}(*)\big) - \prod_{j\in N(a)\setminus i} M_{j\to a}(x^u_{a,j})$$

Figure 5.4: Message update equations for enumerating covers of a formula.

Similarly, we obtain

$$B_i(1) = \prod_{a\in N_-(i)} M_{a\to i}(*) \left[\prod_{a\in N_+(i)} \Big(M_{a\to i}(x^s_{a,i}) + M_{a\to i}(*)\Big) - \prod_{a\in N_+(i)} M_{a\to i}(*)\right]. \tag{5.17}$$

Now, i will assume the joker state $*$ in a cover of F, if it is unconstrained in each of the clauses A, B, C, D. Thus, we obtain

$$\begin{aligned} B_i(*) =& M_{A\to i}(*)M_{C\to i}(*)M_{B\to i}(*)M_{D\to i}(*) \\ =& \prod_{a\in N(i)} M_{a\to i}(*). \end{aligned} \tag{5.18}$$

We now establish the equivalence of the above described cover counting procedure with the SP update equations for a tree structured SAT formula.

Theorem 5.2. *Let F be a CNF formula whose factor graph corresponds to a tree. The message update equations (5.4) derived for enumerating covers of F are equivalent to SP update equations for F.*

Proof. Enforcing the normalization such that $M_{a\to i}(x^s_{i,a}) + M_{a\to i}(*) = 1$ we get

$$M_{a\to i}(x^s_{i,a}) = \prod_{j\in N(a)\setminus i} \frac{M_{j\to a}(x^u_{j,a})}{M_{j\to a}(x^s_{j,a}) + M_{j\to a}(x^u_{j,a}) + M_{j\to a}(*)}. \tag{5.19}$$

Also, the variable to clause message now becomes

$$\begin{aligned} M_{j\to a}(x^s_{j,a}) &= \prod_{b\in N^s_a(j)} (1) \prod_{b\in N^u_a(j)} \big(0 + M_{b\to j}(*)\big) \\ &\quad - \prod_{b\in N^s_a(j)} M_{b\to j}(*) \prod_{b\in N^u_a(j)} M_{b\to j}(*) \\ &= \left[1 - \prod_{b\in N^s_a(j)} M_{b\to j}(*)\right] \prod_{b\in N^u_a(j)} M_{b\to j}(*) \\ &= \left[1 - \prod_{b\in N^s_a(j)} \Big(1 - M_{b\to j}(x^s_{j,b})\Big)\right] \prod_{b\in N^u_a(j)} \Big(1 - M_{b\to j}(x^s_{j,b})\Big). \end{aligned} \tag{5.20}$$

Similarly,

$$M_{j\to a}(x^u_{j,a}) = \left[1 - \prod_{b\in N^u_a(j)} \Big(1 - M_{b\to j}(x^s_{j,b})\Big)\right] \prod_{b\in N^s_a(j)} \Big(1 - M_{b\to j}(x^s_{j,b})\Big) \tag{5.21}$$

and

$$M_{j\to a}(*) = \prod_{b\in N(j)} \Big(1 - M_{b\to j}(x^s_{j,b})\Big). \tag{5.22}$$

Now setting $\eta_{a\to i} = M_{a\to i}(x^s_{i,a})$, $\Pi^s_{i\to a} = M_{i\to a}(x^s_{i,a})$, $\Pi^u_{i\to a} = M_{i\to a}(x^u_{i,a})$ and $\Pi^*_{i\to a} = M_{i\to a}(*)$ we obtain the following SP equations

$$\eta_{a\to i} = \prod_{j\in N(a)\setminus i} \frac{\Pi^u_{j\to a}}{\Pi^u_{j\to a} + \Pi^s_{j\to a} + \Pi^*_{j\to a}}, \tag{5.23}$$

$$\Pi^u_{i\to a} = \prod_{b\in N^s_a(i)} (1 - \eta_{b\to i})\left[1 - \prod_{b\in N^u_a(i)} (1 - \eta_{b\to i})\right], \tag{5.24}$$

$$\Pi^s_{i\to a} = \prod_{b\in N^u_a(i)} (1 - \eta_{b\to i})\left[1 - \prod_{b\in N^s_a(i)} (1 - \eta_{b\to i})\right], \tag{5.25}$$

$$\Pi^*_{i\to a} = \prod_{b\in N(i)\setminus a} (1 - \eta_{b\to i}). \tag{5.26}$$

$\square$

We have seen that SP update equations can be used to find the marginals over covers of a formula F if the associated factor graph is a tree. But, these equations can still be applied to general graphs, hoping that they converge and the fixed point beliefs can be used to approximate cover marginals. We now present the SP algorithm for general graphs and then briefly discuss its success in the HARD-SAT region.

Algorithm 6 *SP*

Input: A factor graph of a Boolean formula in conjunctive normal form; a maximum number of iterations t_{max}; a requested precision ϵ

Output: Satisfying assignment of the formula (if one exists)

```
 1: For t=0, randomly initialize the surveys, $\eta^0_{a\to i}$ .
 2: Sweep the set of edges in some order and update sequentially the messages on all
    edges of the graph, generating a new set of surveys $\eta^t_{a\to i}$, using message update
    Equations (5.1) (5.2) (5.3) (5.4).
 3: for $t = 1$ to $t = t_{max}$ do
 4:   if $|\eta^t_{a\to i} - \eta^{t-1}_{a\to i}| < \epsilon$ for each edge $\{a, i\}$ then
 5:     break;
 6:   end if
 7: end for
 8: if $t = t_{max}$ then
 9:   return UNCONVERGED
10: else
11:   if the fixed point surveys $\eta^*_{a\to i}$ are nontrivial i.e.$\{\eta \neq 0\}$ then
12:     Compute for each variable node $i$, the biases $\{B^+_i, B^-_i\}$ defined as
```

$$B^+_i = \frac{\pi^+_i}{\pi^+_i + \pi^-_i + \pi^*_i} \tag{5.27}$$

$$B^-_i = \frac{\pi^-_i}{\pi^+_i + \pi^-_i + \pi^*_i} \tag{5.28}$$

where

$$\pi^+_i = \left[1 - \prod_{b\in N_+(i)} (1 - \eta_{b\to i})\right] \prod_{b\in N_-(i)} (1 - \eta_{b\to i}), \tag{5.29}$$

$$\pi^-_i = \left[1 - \prod_{b\in N_-(i)} (1 - \eta_{b\to i})\right] \prod_{b\in N_+(i)} (1 - \eta_{b\to i}), \tag{5.30}$$

$$\pi^*_i = \prod_{b\in N(i)} (1 - \eta_{b\to i}). \tag{5.31}$$

```
13:     DECIMATION - Fix the variable with the largest bias, $|B^+_i - B^-_i|$ to the
        value 1 if $B^+_i > B^-_i$, to 0 if $B^+_i < B^-_i$. Reduce the factor graph, by removing
        the variable and the factor nodes corresponding to the satisfied clauses. If no
        contradiction is found Go to Step 1, otherwise STOP.
14:   else if all the surveys are trivial then
15:     Solve the simplified formula by a local search process.
16:   end if
17: end if
```

5.5 Discussion

We now discuss the behavior of the SP algorithm with respect to the known empirical results in random SAT problems. The surveys in the SP algorithm may converge at some stage to give a trivial solution, i.e., a $\{*\}^n$ solution. Such problems may be under-constrained and can be easily solved by other simpler algorithms. The SP can be run again with a different set of initializations, to find a non-trivial fixed point. Numerical experiments show that on large random instances of 3-SAT, in the hard SAT region, but not too close to the satisfiability threshold, the algorithm converges to a non-trivial fixed point. In the other case, the SP algorithm may not converge at some stage, even if the initial problem was satisfiable. This happens in general very close to the satisfiability threshold. It is not yet clear why such a problem arises.

The validity of the SP algorithm relies on the conjectures about the phase structure of the solution space of SAT problems. When the solutions organize themselves in well separated clusters, it is believed that the simpler algorithms like BP fail because they get trapped in various local optimal configurations of the sub-parts of the problem. These local optimal solutions can no longer be combined into a global solution. SP is supposed to take into account the clustering phenomena by finding the core of a cluster. Also, it is conjectured that most of the variables inside such cores are frozen, i.e., we have fewer undecided variables. It is then easy to assign the undecided variables by some other simpler method.

Bibliography

[1] R. Kindermann and J. Laurie Snell. *Markov Random Fields and Their Applications.* American Mathematical Society, 1980.

[2] S. L. Lauritzen. *Graphical Models.* Clarendon Press, Oxford, 1996.

[3] G. M. Grimmet. A theorem about random fields. *Bulletin of London Mathematical Society*, 5:81–84, 1973.

[4] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[5] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *Exploring artificial intelligence in the new millennium.* Morgan Kaufmann Publishers Inc., 2003.

[6] Adnan Darwiche. Conditioning methods for exact and approximate inference in causal networks. In *In Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 99–107. Morgan Kauffman, 1995.

[7] Noga Alon, Paul Seymour, and Robin Thomas. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society*, 3(4):801–808, 1990. URL http://www.jstor.org/stable/1990903.

[8] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Comput.*, pages 1–41, 2000. doi: http://dx.doi.org/10.1162/089976600300015880.

[9] Yair Weiss and William T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Comput.*, 13(10):2173–2200, 2001. ISSN 0899-7667. doi: http://dx.doi.org/10.1162/089976601750541769.

[10] Thomas M. Cover and Joy A. Thomas. *Elements of information theory.* Wiley-Interscience, New York, NY, USA, 1991. ISBN 0-471-06259-6.

[11] Srinivas M. Aji and Robert J. Mceliece. The generalized distributive law and free energy minimization. In *Proceedings of the 39th Allerton Conference*, 2001. URL http://www.mceliece.caltech.edu/publications/Allerton01.pdf.

[12] Alessandro Pelizzola. Cluster variation method in statistical physics and probabilistic graphical models. *J.PHYS.A*, 38:R309, 2005. URL doi:10.1088/0305-4470/38/33/R01.

[13] Marc Mézard and Riccardo Zecchina. Random k-satisfiability problem: From an analytic solution to an efficient algorithm. *Phys. Rev. E*, 66(5):056126, Nov 2002. doi: 10.1103/PhysRevE.66.056126.

[14] A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: An algorithm for satisfiability. *Random Struct. Algorithms*, 27(2):201–226, 2005. ISSN 1042-9832. doi: http://dx.doi.org/10.1002/rsa.v27:2.

[15] A. Braunstein and R. Zecchina. Survey propagation as local equilibrium equations, 2004. URL doi:10.1088/1742-5468/2004/06/P06007.

[16] Elitza Maneva, Elchanan Mossel, and Martin J. Wainwright. A new look at survey propagation and its generalizations. *J. ACM*, 54(4):17, 2007. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/1255443.1255445.

[17] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM. doi: http://doi.acm.org/10.1145/800157.805047.

[18] Dimitris Achlioptas and Federico Ricci-Tersenghi. On the solution-space geometry of random constraint satisfaction problems. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 130–139. ACM, 2006. ISBN 1-59593-134-1. doi: http://doi.acm.org/10.1145/1132516.1132537.

[19] Rémi Monasson and Riccardo Zecchina. Statistical mechanics of the random k-satisfiability model. *Phys. Rev. E*, 56(2):1357–1370, Aug 1997. doi: 10.1103/PhysRevE.56.1357.