MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Bethe free energy, Kikuchi approximations, and belief propagation algorithms

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss

TR-2001-16   May 2001

Abstract

This is an updated and expanded version of TR2000-26, but it is still in draft form.

Belief propagation (BP) was only supposed to work for tree-like networks but works surprisingly well in many applications involving networks with loops, including turbo codes. However, there has been little understanding of the algorithm or the nature of the solutions it finds for general graphs.

We show that BP can only converge to a stationary point of an approximate free energy, known as the Bethe free energy in statistical physics. This result characterizes BP fixed-points and makes connections with variational approaches to approximate inference.

More importantly, our analysis lets us build on the progress made in statistical physics since Bethe's approximation was introduced in 1935. Kikuchi and others have shown how to construct more accurate free energy approximations, of which Bethe's approximation is the simplest. Exploiting the insights from our analysis, we derive generalized belief propagation (GBP) versions of these Kikuchi approximations. These new message passing algorithms can be significantly more accurate than ordinary BP, at an adjustable increase in complexity. We illustrate such a new GBP algorithm on a grid Markov network and show that it gives much more accurate marginal probabilities than those found using ordinary BP.

This is an updated and expanded version of TR2000-26, but is still in draft form as of its release on April 4, 2001. Small further update (section 2.3) on May 15, 2001.

Bethe free energy, Kikuchi approximations and belief propagation algorithms

Jonathan S. Yedidia
MERL
201 Broadway
Cambridge, MA 02139
Phone: 617-621-7544
yedidia@merl.com

William T. Freeman
MERL
201 Broadway
Cambridge, MA 02139
Phone: 617-621-7527
freeman@merl.com

Yair Weiss
Computer Science Division
UC Berkeley, 485 Soda Hall
Berkeley, CA 94720-1776
Phone: 510-642-5029
yweiss@cs.berkeley.edu

Abstract

Belief propagation (BP) was only supposed to work for tree-like networks but works surprisingly well in many applications involving networks with loops, including turbo codes. However, there has been little understanding of the algorithm or the nature of the solutions it finds for general graphs.

We show that BP can only converge to a stationary point of an approximate free energy, known as the Bethe free energy in statistical physics. This result characterizes BP fixed-points and makes connections with variational approaches to approximate inference.

More importantly, our analysis lets us build on the progress made in statistical physics since Bethe's approximation was introduced in 1935. Kikuchi and others have shown how to construct more accurate free energy approximations, of which Bethe's approximation is the simplest. Exploiting the insights from our analysis, we derive generalized belief propagation (GBP) versions of these Kikuchi approximations. These new message passing algorithms can be significantly more accurate than ordinary BP, at an adjustable increase in complexity. We illustrate such a new GBP algorithm on a grid Markov network and show that it gives much more accurate marginal probabilities than those found using ordinary BP.

1 Introduction

Local "belief propagation" (BP) algorithms such as those introduced by Pearl are guaranteed to converge to the correct marginal posterior probabilities in tree-like graphical models. For general networks with loops, the situation is much less clear.

On the one hand, a number of researchers have empirically demonstrated good performance for BP algorithms applied to graphs with loops. One dramatic case is the near Shannon-limit performance of "Turbo codes", whose decoding algorithm is equivalent to BP on a loopy graph [3, 10]. Other successful cases include computer vision problems and medical diagonsis [3, 2, 12] suggesting the success of BP on loopy graphs is not limited to coding applications. On the other hand, for other graphs with loops, BP may give poor results or fail to converge [12].

For a general graph, little has been understood about what approximation BP represents, and how it might be improved. This paper's goal is to provide that understanding and introduce a set of new algorithms resulting from that understanding. We show that BP is the first in a progression of local message-passing algorithms, each giving equivalent results to a corresponding approximation from statistical physics known as the "Kikuchi" approximation to the Gibbs free energy. These algorithms have the attractive property of being user-adjustable: by paying some additional computational cost, one can obtain considerable improvement in the accuracy of one's approximation, and can sometimes obtain a convergent message-passing algorithm when ordinary BP does not converge.

2 Belief propagation fixed-points are zero gradient points of the Bethe free energy

For the purpose of analyzing and describing BP we assume that we are given an undirected graphical model of N nodes with pairwise potentials (a pairwise Markov Random Field). Any graphical model can be converted into this form before doing inference through a suitable clustering of nodes into larger nodes [17]. Through this transformation, one can apply all the results in this paper to arbitrary graphical models including those with higher-order potentials. We will give explicit examples of this in later sections.

The state of each unobserved node i is denoted by x_i, and we assume each unobserved node is connected to an observed node y_i. The joint probability distribution function is given by

$$P(x_1, x_2, ..., x_N|y) = \frac{1}{Z} \prod_{ij} \psi_{ij}(x_i, x_j) \prod_i \psi_i(x_i, y_i) \tag{1}$$

where $\psi_i(x_i, y_i)$ is the local "evidence" for node i, $\psi_{ij}(x_i, x_j)$ is the compatibility matrix between nodes i and j, and Z is a normalization constant. In what follows, we simplify notation and write $\psi_i(x_i)$ as shorthand for $\psi_i(x_i, y_i)$.

The standard BP update rules are:

$$\begin{aligned} m_{ij}(x_j) &\leftarrow k \sum_{x_i} \psi_{ij}(x_i, x_j) \psi_i(x_i) \prod_{k \in N(i) \backslash j} m_{ki}(x_i) & (2) \\ b_i(x_i) &\leftarrow k \psi_i(x_i) \prod_{k \in N(i)} m_{ki}(x_i) & (3) \end{aligned}$$

where k denotes a normalization constant and $N(i) \backslash j$ means all nodes neighboring node i, except j. Here m_{ij} refers to the message that node i sends to node j and b_i is the belief (approximate marginal posterior probability) at node i, obtained by multiplying all incoming messages to that node by the local evidence. Similarly, we can define the belief $b_{ij}(x_i, x_j)$ at the pair of nodes (x_i, x_j) as the product of the local potentials and all messages incoming to the pair of nodes:

$$b_{ij}(x_i, x_j) = k\phi_{ij}(x_i, x_j) \prod_{k \in N(i)\backslash j} m_{ki}(x_i) \prod_{l \in N(j)\backslash i} m_{lj}(x_j) \tag{4}$$

where $\phi_{ij}(x_i, x_j) \equiv \psi_{ij}(x_i, x_j)\psi_i(x_i)\psi_j(x_j)$.

For a tree, the BP iterations converge to a unique fixed point and the beliefs are equal to the posterior marginals $b_i(x_i) = P(x_i)$ (e.g. [13]). Similarly, one can show that for a tree the pairwise beliefs are equal to the posterior marginals $b_{ij}(x_i, x_j) = P(x_i, x_j)$. The following claim characterizes b_i and b_{ij} for an arbitrary graph, showing that they are equivalent to the marginal probabilities obtained in the Bethe approximation developed in statistical physics [1].

Claim 1: Let $\{m_{ij}\}$ be a set of BP messages and let $\{b_{ij}, b_i\}$ be the beliefs calculated from those messages. Then the beliefs are fixed-points of the BP algorithm if and only if they are zero gradient points of the Bethe free energy, F_β:

$$\begin{aligned} F_\beta(\{b_{ij}, b_i\}) &= -\sum_{ij}\sum_{x_i,x_j} b_{ij}(x_i,x_j)\ln\phi_{ij}(x_i,x_j) + \sum_i (q_i - 1)\sum_{x_i} b_i(x_i)\ln\psi_i(x_i) \\ &\quad + \sum_{ij}\sum_{x_i,x_j} b_{ij}(x_i,x_j)\ln b_{ij}(x_i,x_j) - \sum_i (q_i - 1)\sum_{x_i} b_i(x_i)\ln b_i(x_i) \end{aligned} \tag{5}$$

subject to the normalization and marginalization constraints: $\sum_{x_i} b_i(x_i) = 1$, $\sum_{x_i} b_{ij}(x_i, x_j) = b_j(x_j)$. ($q_i$ is the number of neighbors of node i.)

Proof: We add Lagrange multipliers to form a Lagrangian L: $\lambda_{ij}(x_j)$ is the multiplier corresponding to the constraint that $b_{ij}(x_i, x_j)$ marginalizes down to $b_j(x_j)$, and γ_{ij}, γ_i are multipliers corresponding to the normalization constraints. The equation $\frac{\partial L}{\partial b_{ij}(x_i,x_j)} = 0$ gives: $\ln b_{ij}(x_i, x_j) = \ln(\phi_{ij}(x_i, x_j)) + \lambda_{ij}(x_j) + \lambda_{ji}(x_i) + \gamma_{ij} - 1$. The equation $\frac{\partial L}{\partial b_i(x_i)} = 0$ gives: $(q_i - 1)(\ln b_i(x_i) + 1) = \ln \psi_i(x_i) + \sum_{j \in N(i)} \lambda_{ji}(x_i) + \gamma_i$. Setting $\lambda_{ij}(x_j) = \ln \prod_{k \in N(j)\backslash i} m_{kj}(x_j)$ and using the marginalization constraints, we find that the stationary conditions on the Lagrangian are equivalent to the BP fixed-point conditions. To go in the opposite direction, if we are given $b_{ij}, b_i, \lambda_{ij}(x_j)$ that correspond to a zero-gradient point, we set $m_{ij}(x_j) = \frac{b_j(x_j)}{\exp(\lambda_{ij}(x_j))}$. Because $b_{ij}, b_i, \lambda_{ij}$ satisfy the stationarity conditions, m_{ij} defined in this way must be a fixed point of BP. Thus there is a one-to-one correpsondence between fixed-points of BP and staionary points of the Bethe free energy. □

(Empirically, we find that *stable* BP fixed-points correspond to local *minima* of the Bethe free energy, rather than maxima or saddle-points.)

Using the marginalization conditions and the definition of $\phi(x_i, x_j)$ the Bethe free energy can also be written:

$$\begin{aligned} F_\beta(\{b_{ij}, b_i\}) &= -\sum_{ij}\sum_{x_i,x_j} b_{ij}(x_i,x_j)\ln\psi_{ij}(x_i,x_j) - \sum_i \sum_{x_i} b_i(x_i)\ln\psi_i(x_i) \\ &\quad + \sum_{ij}\sum_{x_i,x_j} b_{ij}(x_i,x_j)\ln b_{ij}(x_i,x_j) - \sum_i (q_i - 1)\sum_{x_i} b_i(x_i)\ln b_i(x_i) \end{aligned} \tag{6}$$

Where does the Bethe free energy come from? We will discuss the physics intuition behind it in section 3 but to make the formula less mysterious we now show that it is an exact free energy when the graph is a tree. In other words, minimizing the Bethe free energy for a tree is equivalent to performing exact inference.

The exact free energy is simply the average energy minus the entropy:

$$F(\{b(x)\}) = \sum_x b(x)E(x) + \sum_x b(x)\log b(x) \tag{7}$$

where $b(x)$ is the "belief" in a state x and $E(x)$ the "energy" of x motivated by Boltzmann's law $P(x|y) = \frac{1}{Z}e^{-E(x)}$:

$$E(x) = -\log P(x|y) - \log Z = -\sum_{<ij>} \log \psi_{ij}(x_i, x_j) - \sum_i \log \psi_i(x_i) \tag{8}$$

Note that F is (up to a constant) the KL divergence between $b(x)$ and $P(x|y)$ so that F is minimized when $b(x) = P(x|y)$.

By substituting equation 8 in the average energy we obtain:

$$\sum_x b(x)E(x) = -\sum_{ij}\sum_{x_i,x_j} b_{ij}(x_i,x_j)\ln\psi_{ij}(x_i,x_j) - \sum_i\sum_{x_i} b_i(x_i)\ln\psi_i(x_i) \tag{9}$$

so that the first two terms of the Bethe free energy (eq. 6) are exactly the average energy.

If the graph is a tree, then we can restrict the minimization to the class of $b(x)$ that satisfy the conditional independences implied by the graph. Such $b(x)$ can be written as a product of beliefs over cliques, divided by beliefs over separator sets (see e.g. [13]). In our setting, the cliques are pairs of nodes and the separator sets are single nodes so that:

$$b(x) = \frac{\prod_{<ij>} b_{ij}(x_i, x_j)}{\prod_i b_i(x_i)^{q_i - 1}} \tag{10}$$

If we substitute equation 10, into the expression for the negative entropy $\sum_x b(x)\ln b(x)$ we obtain the last two terms of the Bethe free energy (eq. 6). Thus for a tree graph, the Bethe free energy is exact and it will be minimized when $b_{ij}(x_i, x_j) = P(x_i, x_j|y)$ and $b_i(x_i) = P(x_i|y)$.

2.1 Special case: turbo codes

We follow the formulation of Turbo decoding presented in [11]. An unknown binary vector u is encoded using two constituent codes, each of which is easy to decode by itself and the results are transmitted over a noisy channel giving two observations y_1, y_2. The task of Turbo decoding is to infer the values of bits of u from the two observations y_1, y_2. The decoder knows the prior distribution over u (which we assume here is uniform) and the two conditional probabilities $p_i(u) = p(y_i|u)$.

The turbo decoding algorithm iterates the following iterations:

$$\beta_i(u_i) \leftarrow k \sum_{w:w_i=u_i} p_1(w) \prod_{j\neq i} \alpha_j(w_j) \tag{11}$$

$$\alpha_i(u_i) \leftarrow k \sum_{w:w_i=u_i} p_2(w) \prod_{j\neq i} \beta_j(w_j) \tag{12}$$

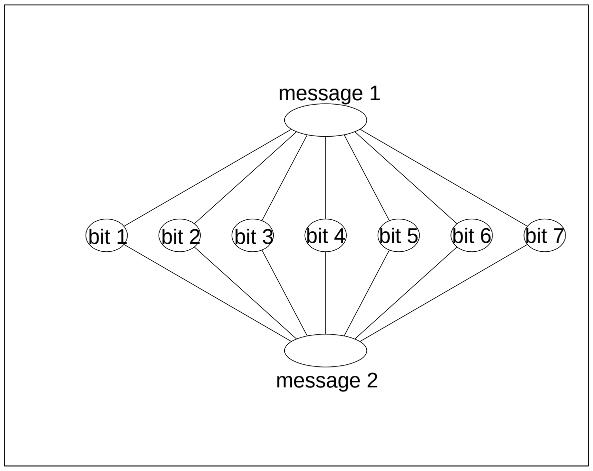


Figure 1: The Turbo code structure. Running BP on this graph is equivalent to the turbo decoding algorithm.

and the marginal probability over u_i is approximated by:

$$b(u_i) = k\alpha_i(u_i)\beta_i(u_i) \tag{13}$$

after the iterations have converged (or reached a maximal number of iterations).

Note that the Turbo decoding algorithm as formulated here requires summing over an exponentially large number of states in each iteration. The reason this is possible is that the likelihoods $p_1(w), p_2(w)$ come from an easy to decode code. In other words, there is a factorization structure on these likelihoods that correspond to a singly-connected graph: this enables turbo decoders to compute the exponential sum over states in linear time. Essentially, this is done by an algorithm equivalent to BP.

As shown in [18, 8, 10], the Turbo decoding algorithm is equivalent to belief propagation. In our undirected formalism, it is equivalent to iterating the BP equations on the graph shown in figure 1. The top and bottom nodes correspond to codewords w^1, w^2 and the nodes in the middle layer correspond to unknown bits. The potentials between nodes are set to delta functions: $\psi(w^1, u_i) = 1$ if the ith bit of w_1, w^1_i is equal to u_i and zero otherwise. Nodes w^1 and w^2 have observation potentials $p_1(w), p_2(w)$ corresponding to the likelihood of a codeword from one constituent code.

When BP is run on this graph, the messages reaching u_i are exactly α_i, β_i used in turbo decoding and equations 11–12 follow directly from the standard BP updates [16]. The BP formalism also allows us to compute beliefs over w_1, w_2 given by:

$$b_1(w) \quad = \quad kp_1(w) \prod_i \alpha_i(w_i) \tag{14}$$

$$b_2(w) = kp_2(w) \prod_i \beta_i(w_i) \tag{15}$$

$$\tag{16}$$

Applying claim 1 to the graph shown in figure 1 gives a Bethe free energy that includes joint beliefs of the form $b(w^i, u_j)$, but because the energy is infinite for any such belief for which $w^i_j \neq u_j$ such beliefs must be of the form $b(w^i)\delta(w^i_j - u_j)$ for the Bethe free energy to be finite. Using this fact, we can simplify the Bethe free energy. The simplified free energy is now only a function of $b_1(w), b_2(w)$ and $b_i(u_i)$.

Corollary 1: Let α_i, β_i be a set of Turbo decoding messages and let $\{b_i(w), b_i(u_i)\}$ be the beliefs calculated from those messages. Then the beliefs are fixed-points of the Turbo decoding algorithm if and only if they are zero gradient points of the Bethe free energy, F_β:

$$\begin{aligned} F_\beta &= -\sum_x b_1(w) \ln p_1(w) + \sum_w b_1(w) \ln b_1(w) && (17) \\ &\quad - \sum_x b_2(w) \ln p_2(w) + \sum_w b_2(w) \ln b_2(w) && (18) \\ &\quad - \sum_i \sum_{u_i} b_i(u_i) \ln b_i(u_i) && (19) \end{aligned}$$

subject to the constraints that $b_1(w)$ and $b_2(w)$ marginalize down to $b_i(u_i)$ for all i.

Proof: This follows from claim 1. Alternatively, one can prove it directly by adding Lagrange multipliers $\lambda_\alpha(u_i)$ enforces the constraint that $b_1(w)$ marginalize down to $b_i(u_i)$ and $\lambda_\beta(u_i)$ enforces the constraint that $b_2(w)$ marginalize down to $b_i(u_i)$. Setting $\lambda_\beta(u_i) = \ln \beta(u_i), \lambda_\alpha(u_i) = \ln \alpha(u_i)$ gives a set of Lagrange multipliers that satisfy the stationarity conditions if and only α, β are fixed points of the turbo decoding algorithm. $\square$

2.2 Special case: low density parity check codes

Figure 2 shows the pairwise MRF corresponding to a low density parity check code. We use Roman letters like i to denote the bits and Greek letters like α to denote the parity checks. In this example, the bits i can be in one of two states denoted by x_i, while the check bits can be in one of eight states denoted by x_α. If the ith bit is connected to the αth check node, then we use the notation $x_\alpha(i)$ to denote the state that the ith node should be in to correspond with the αth check node being in the state x_α. The pairwise potentials $\psi(x_i, x_\alpha)$ are set to one if $x_\alpha(i) = x_i$ and zero otherwise. The singleton potentials are set to $\psi_i(x_i) = p(y_i|x_i)$ for the bit nodes while $\psi_\alpha(x_\alpha)$ is one if x_α corresponds to an even parity state and zero otherwise. The BP update rules on this Markov graph give the Gallager decoding algorithm:

$$m_{\alpha i}(x_i) \leftarrow \sum_{x_\alpha : x_\alpha(i) = x_i} \psi_\alpha(x_\alpha) \prod_{k \in N(\alpha) \setminus i} m_{k\alpha}(x_\alpha) \tag{20}$$

$$m_{i\alpha}(x_\alpha) \leftarrow \psi_i(x_\alpha(i)) \prod_{\beta \in N(i) \setminus \alpha} m_{\beta i}(x_\alpha(i)) \tag{21}$$

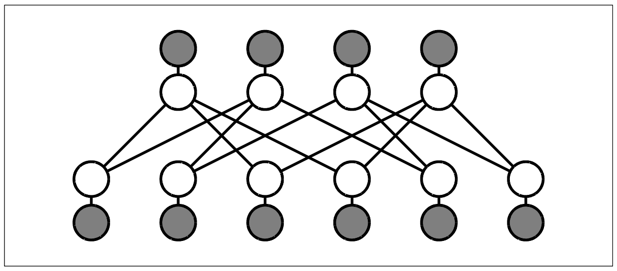


Figure 2: The pairwise Markov graph for a rate 1/3, $(3, 2)$ low density parity check code. The nodes in the bottom layer represent unknown bits and the nodes in the top layer represent triplets of bits. Each triplet node has an observation connected to it that gives zero posterior probability if the triplet has odd parity. The nodes in the bottom layer have observation nodes that correspond to noisy versions of the unknown bits. The potentials between the bit nodes and the triplet nodes are set to one or zero depending on whether the triplet agrees with the bit node.

$$b_i(x_i) \leftarrow \psi_i(x_i) \prod_{\alpha \in N(i)} m_{\alpha i}(x_i) \tag{22}$$

$$b_\alpha(x_\alpha) \leftarrow \psi(x_\alpha) \prod_{i \in N(\alpha)} m_{i\alpha}(x_\alpha) \tag{23}$$

Applying claim to figure 2 will again give a Bethe free energy that depends on joint beliefs of the form $b(x_\alpha, x_i)$ but because of the infinite energy associated with configurations for which $x_\alpha(i) \neq x_i$, $b(x_\alpha, x_i)$ must be of the form $b(x_\alpha)\delta(x_\alpha(i) - x_i)$. Thus we obtain a simplified Bethe free energy that depends only on $b(x_\alpha)$ and $b(x_i)$.

Corollary 2: A set of messages and beliefs are fixed-points of the Gallager decoding algorithm for LDPC codes if and only if they are stationary points of:

$$F_\beta = -\sum_i \sum_{x_i} b_i(x_i) \ln \psi_i(x_i) - \sum_\alpha \sum_{x_\alpha} b_\alpha(x_\alpha) \ln \psi_\alpha(x_\alpha) \tag{24}$$

$$+\sum_\alpha \sum_{x_\alpha} b_\alpha(x_\alpha) \ln b_\alpha(x_\alpha) - \sum_i (q_i - 1) \sum_{x_i} b_i(x_i) \ln b_i(x_i) \tag{25}$$

subject to the constraint that $b_\alpha(x_\alpha)$ marginalize down to $b_i(x_i)$ for all $i \in N(\alpha)$.

Proof: This again follows from claim 1 or can be proven directly by adding Lagrange multipliers $\lambda_{i\alpha}(x_i)$ that enforce the constraint that $b(x_\alpha)$ marginalize down to x_i. If we define $\lambda_{\alpha i}(x_i) = \ln m_{\alpha i}(x_i)$ we find that the fixed-point equations for BP are equivalent to the stationarity conditions for F_β. □

2.3 Special case: factor graphs

A factor graph [9] is a bipartite graph with *function* nodes f_i denoted by filled squares and *variable* nodes x_i denoted by unfilled circles. The function nodes denote a decomposition of a "global" function $g(x)$ into a product of "local" functions $f_\alpha(x)$. We will assume that g represents a joint distribution over the variable nodes.

As shown in [17], any factor graph can be converted into a pairwise Markov graph that represents the same distribution. The procedure is identical to the one described in the previous section for LDPCs: we just need to treat function nodes in the same way we treated check nodes in the LDPC. This procedure yields a pairwise Markov graph so that iterating the BP equations on this graph is equivalent to running the sum-product algorithm on the factor graph [17].

We again denote by x_α the set of variables that form the domain of function f_α. As before, the BP algorithm will calculate beliefs over individual variables $b_i(x_i)$ as well as joint beliefs over all variables that form the domain of a single function $b_\alpha(x_\alpha)$. The Bethe free energy of the pairwise Markov graph can be simplified and we obtain a Bethe free energy that depends only on $b_(x_i)$ and $b_\alpha(x_\alpha)$.

Corollary 3: A set of messages and beliefs are fixed-points of the sum-product algorithm on a factor graph if and only if they are stationary points of:

$$\begin{aligned} F_\beta &= -\sum_\alpha \sum_{x_\alpha} b_\alpha(x_\alpha) \ln f_\alpha(x_\alpha) && (26) \\ &\quad + \sum_\alpha \sum_{x_\alpha} b_\alpha(x_\alpha) \ln b_\alpha(x_\alpha) - \sum_i (q_i - 1) \sum_{x_i} b_i(x_i) \ln b_i(x_i) && (27) \end{aligned}$$

subject to the constraint that $b_\alpha(x_\alpha)$ marginalize down to $b_i(x_i)$ for all $i \in N(\alpha)$. Here q_i is the number of functions that variable i participates in.

2.4 Special case: directed graphs (Bayesian networks)

Bayesian networks are directed graphs in which the conditional probability over all hidden nodes is written as:

$$P(x|y) = \frac{1}{Z} \prod_i P(x_i|Par(x_i)) P(y_i|x_i) \qquad (28)$$

where $Par(x_i)$ means the parents of x_i. We have assumed that every hidden node x_i has an observed node y_i connected to it, and are subsuming the prior over the root node into its local evidence.

For completeness we give Pearl's algorithm using the formulation of Peot and Shacter [14]. Each node x_i computes a belief $b(x_i)$, by combining messages from its children $\lambda_{ji}(x_i)$ and messages from its parents $\pi_{ki}(x_k)$.

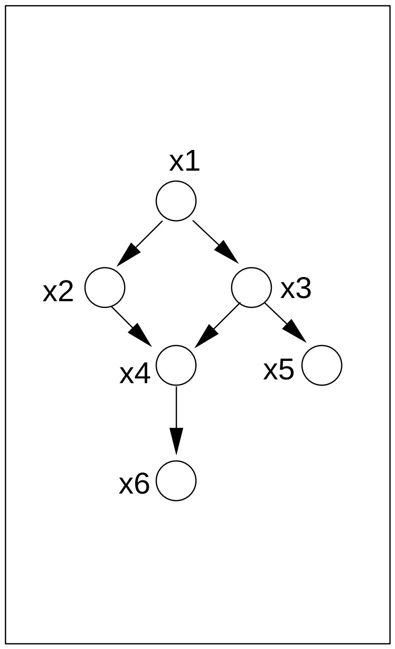
$$b(x_i) = \alpha \lambda(x_i) \pi(x_i) \qquad (29)$$

where the λ and π quantities are derived from iterating:

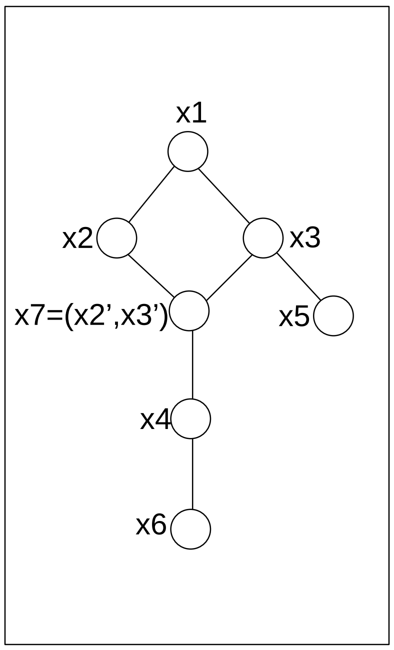
$$\lambda(x_i) \leftarrow P(y_i|x_i) \prod_j \lambda_{ji}(x_i) \qquad (30)$$

and:

$$\pi(x_i) \leftarrow \sum_{Par(x)} P(X = x|Par(x)) \prod_k \pi_{ki}(x_k) \qquad (31)$$



a



b

Figure 3: Any Bayesian network can be converted into an undirected graph with pairwise cliques by adding cluster nodes for all parents that share a common child. **a.** A Bayesian network. **b.** The corresponding undirected graph with pairwise cliques. A cluster node for (B, C) has been added. The potentials can be set so that the joint probability in the undirected network is identical to that in the Bayesian network. In this case the update rules presented in this paper reduce to Pearl's propagation rules in the original Bayesian network [16].

The message x_i passes to its parent x_k is given by:

$$\lambda_{ik}(x_k) \leftarrow \alpha \sum_{x_i} \lambda(x_i) \sum_{u:u_k=x_k} P(x_i|u) \prod_{j\neq k} \pi_{ji}(u_j) \tag{32}$$

where $\sum_{u:u_k=x_k}$ means sum over all configurations for the parents of x_i where the value of x_k is clamped.

The message x_i sends to its child x_j is given by:

$$\pi_{ij}(x_i) \leftarrow \alpha\pi(x_i)P(y_i|x_i) \prod_{k\neq j} \lambda_{ik}(x_i) \tag{33}$$

As shown in figure 3 every directed graph can be transformed into an undirected graph with pairwise potentials. Furthermore, as detailed in [16], running the undirected BP equations on the transformed graph is equivalent to running Pearl's algorithm on the directed graph. We emphasize that this equivalence holds for every iteration: every message sent by undirected BP on the transformed graph has a corresponding message in Pearl's algorithm on the directed graph. For example, for nodes having a single parent $\lambda_{ik}(x_k)$ is exactly $m_{ik}(x_k)$. For nodes having multiple parents, $\lambda_{ik}(x_k)$ is the message passed from the "cluster node"(e.g. x_7 in figure 3) to that parent.

Just as we can use undirected BP to obtain pairwise beliefs, we can use Pearl's algorithm to calculate *family beliefs*: the joint belief over a node and its parents. We define this belief, $b(x_i, Par(x_i)$, as the product of (1) all messages coming into x_i and its parents, (2) the local messages for all nodes in the family and (3) the conditional $P(x_i|Par(x_i))$. Thus:

$$b(x_i, Par(x_i)) = \alpha\lambda(x_i)P(x_i|Par(x_i)) \prod_{x_k \in Par(x_i)} \pi_{ki}(x_k) \tag{34}$$

Using this transformation, we can apply claim 1 and obtain the Bethe free energy for the transformed graph. As in the case of turbo codes and LDPCs, this free energy can be simplified because some configurations in the transformed graph have zero probability. We can then rewrite the Bethe free energy in terms of the directed graph, and obtain the following corollary.

Corollary 4: Let $\{b_i(x_i)\}$,$\{b(x_i, Par(x_i))\}$ be a set of beliefs derived from Pearl's algorithm on a directed graph with arbitrary topology. The beliefs correspond to fixed-points of Pearl's algorithm if and only if they are constrained stationary points of the Bethe free energy:

$$\begin{aligned} F_\beta \quad = \quad & -\sum_i \sum_{x_i,Par(x_i)} b(x_i, Par(x_i)) \log P(x_i|Par(x_i)) - \sum_i \sum_{x_i} b(x_i) \log P(y_i|x_i) \\ & + \sum_i \sum_{x_i,Par(x_i)} b(x_i, Par(x_i)) \log b(x_i, Par(x_i)) - \sum_i (q_i - 1) \sum_{x_i} b(x_i) \log b(x_i) \end{aligned} \tag{35}$$

The minimization is constrained so that $b(x_i, Par(x_i))$ marginalizes down to the singleton belief $b(x_j)$ for all $x_j \in (x_i, Par(x_i))$. Here q_i is the number of families that node x_i participates in, which is equal to one plus the number of children of x_i for all nonroot nodes.

2.5 Implications

The fact that $F_\beta(\{b_{ij}, b_i\})$ is bounded below implies that the BP equations always possess a fixed-point (obtained at the global minimum of F_β). To our knowledge, this is the first proof of existence of fixed-points for a general graph with arbitrary potentials (see [15] for a more proof for the special case of turbo codes).

The free energy formulation clarifies the relationship to variational approaches which also minimize an approximate free energy [6]. For example, the mean field approximation finds a set of $\{b_i\}$ that minimize:

$$F_{MF}(\{b_i\}) = -\sum_{ij}\sum_{x_i,x_j} b_i(x_i)b_j(x_j)\ln\psi_{ij}(x_i,x_j) + \sum_i\sum_{x_i} b_i(x_i)\left[\ln b_i(x_i) - \ln\psi_i(x_i)\right] \tag{36}$$

subject to the constraint $\sum_i b_i(x_i) = 1$.

The Bethe free energy includes first-order terms $b_i(x_i)$ as well as second-order terms $b_{ij}(x_i, x_j)$, while the mean field free energy uses only the first order ones. The BP free energy is exact for trees while the mean field one is not. Furthermore the optimization methods are different: typically F_{MF} is minimized directly in the primal variables $\{b_i\}$: it is easy to define local updates for $b_i(x_i)$ that minimize F_{MF} at every iteration. In contrast, the BP iterations work with the messages which are combinations of the dual variables $\{\lambda_{ij}(x_j)\}$. The BP iterations do not necessarily minimize F_β at every iteration: in fact, for intermediate iterations the b_{ij}, b_i calculated using BP do not satisfy the marginalization constrains.

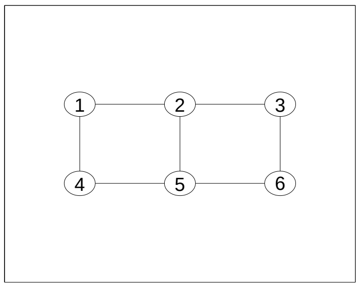
Kabashima and Saad [19] have previously pointed out the correspondence between BP and the Bethe approximation (expressed using the TAP formalism) for some specific graphical models with random disorder. Our proof answers in the affirmative their question about whether there is a "deep general link between the two methods." [19]

3 Kikuchi Approximations to the Free Energy

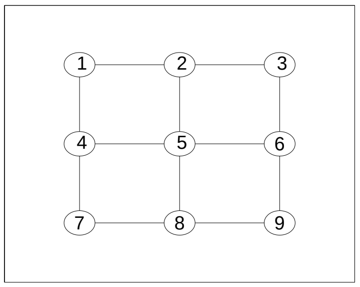
The Bethe approximation, for which the energy and entropy are approximated by terms that involve at most pairs of nodes, is the simplest version of the Kikuchi "cluster variational method." [7, 4, 5] In a general Kikuchi approximation, the energy(or entropy) is approximated as a sum of the energies(or entropies) of basic clusters of nodes, minus the energies(or entropies) of over-counted cluster intersections, minus the energies(or entropies) of the over-counted intersections of intersections, and so on.

Figure 4a shows an example. To obtain the Bethe approximation, we use as basic clusters the set of all pairs of nodes. The Bethe entropy is the sum of all entropies in the basic clusters minus the entropies of over-counted cluster intersections.

$$\begin{aligned} H_\beta(x) &= H(x_{12}) + H(x_{23}) + H(x_{45}) + H(x_{56}) && (37)\\ &\quad +H(x_{14}) + H(x_{25}) + H(x_{36}) && (38)\end{aligned}$$



(a)



(b)

Figure 4: Two examples for the Kikuchi approximations. See text for details.

$$-H(x_1) - H(x_3) - H(x_6) - H(x_4) \tag{39}$$
$$-2H(x_2) - 2H(x_5) \tag{40}$$

where we denote by $H(x_i)$ the entropy of x_i: $H(x_i) = -\sum_{x_i} P(x_i) \ln P(x_i)$. Note that this is exactly the Bethe entropy used in claim 1. Intuitively, because x_1 appears in two clusters, its entropy was over-counted so it is subtracted once. The node, x_2, on the other hand, appears in three clusters, so its entropy is subtracted twice.

A different Kikuchi approximation for the same quantity can be obtained when we use quartets of nodes as the basic clusters. The entropy in figure 4a would then be approximated as:

$$H_{K_4}(x) = H(x_{1245}) + H(x_{2356}) - H(x_{25}) \tag{41}$$

Here nodes x_2 and x_5 appear together in two clusters so we subtract their entropy.

Figure 4b shows a more generic situation. If we again use quartets of nodes as the basic clusters we have:

$$\begin{aligned} H_{K_4} &= H(x_{1245}) + H(x_{2356}) + H(x_{4578}) + H(x_{5689}) \\ &\quad -H(x_{25}) - H(x_{45}) - H(x_{56}) - H(x_{58}) \\ &\quad +H(x_5) \end{aligned} \tag{42}$$

Here the nodes x_2 x_5 appear together in two clusters so we subtract their entropy once and similarly for $4, 5$ $5, 6$ and $5, 8$. But once we do that we see that node x_5 appears in four clusters and its entropy is subtracted four times (in four different pairs of nodes) so we need to add the entropy of node x_5 again.

The Kikuchi average energy is based on a similar intuition. Define the energy of a region by:

$$E_r(x_r) \equiv -\ln \prod_{ij} \psi_{ij}(x_i, x_j) - \ln \prod_i \psi_i(x_i) \equiv -\ln \psi_r(x_r) \tag{43}$$

where the products are over all nodes or pairs of nodes that are contained in region r. And denote by $\bar{E}(x_r; b)$ the average of the energy with respect to the belief $\bar{E}(x_r; b) = \sum_{x_r} b_r(x_r) E_r(x_r)$. Then the average energy of figure 4b using quartets of nodes is:

$$\begin{aligned}\bar{E}_{K_4} \quad = \quad & \bar{E}(x_{1245}) + \bar{E}(x_{2356}) + \bar{E}(x_{4578}) + \bar{E}(x_{5689}) \\ & -\bar{E}(x_{25}) - \bar{E}(x_{45}) - \bar{E}(x_{56}) - \bar{E}(x_{58}) \\ & +\bar{E}(x_5) \qquad (44)\end{aligned}$$

(where we have suppressed the dependence of $\bar{E}$ on b)

For a general graph, let R be a set of regions that include some chosen basic clusters of nodes, their intersections, the intersections of the intersections, and so on. The choice of basic clusters determines the Kikuchi approximation–for the Bethe approximation, the basic clusters consist of all linked pairs of nodes. Let x_r be the state of the nodes in region r and $b_r(x_r)$ be the "belief" in x_r.

The Kikuchi free energy is

$$F_K = \sum_{r \in R} c_r \left(\sum_{x_r} b_r(x_r) E_r(x_r) + \sum_{x_r} b_r(x_r) \log b_r(x_r) \right) \qquad (45)$$

where c_r is the over-counting number of region r, defined by: $c_r = 1 - \sum_{s \in super(r)} c_s$ where $super(r)$ is the set of all super-regions of r. For the largest regions in R, $c_r = 1$. The belief $b_r(\alpha_r)$ in region r has several constraints: it must sum to one and be consistent with the beliefs in regions which intersect with r. In general, increasing the size of the basic clusters improves the approximation one obtains by minimizing the Kikuchi free energy.

4 Generalized belief propagation (GBP)

Minimizing the Kikuchi free energy subject to the constraints on the beliefs is not simple. Nearly all applications of the Kikuchi approximation in the physics literature exploit symmetries in the underlying physical system and the choice of clusters to reduce the number of equations that need to be solved from $O(N)$ to $O(1)$. But just as the Bethe free energy can be minimized by the BP algorithm, we introduce a class of analogous *generalized belief propagation* (GBP) algorithms that minimize an arbitrary Kikuchi free energy. These algorithms represent an advance in physics, in that they open the way to the exploitation of Kikuchi approximations for inhomogeneous physical systems.

There are in fact many possible GBP algorithms which all correspond to the same Kikuchi approximation. We present a "canonical" GBP algorithm which has the nice property of reducing to ordinary BP at the Bethe level. We introduce messages $m_{rs}(x_s)$ between all regions r and their "direct sub-regions" s. (Define the set $sub_d(r)$ of direct sub-regions of r to be those regions that are sub-regions of r but have no super-regions that are also sub-regions of r, and similarly for the set $super_d(r)$ of "direct super-regions.") It is helpful to think of this as a message from those nodes in r but not in s (which we denote by $r \backslash s$) to the nodes in s. Intuitively, we want messages to propagate information that lies outside of a region into it. Thus, for a given region r, we want the belief $b_r(x_r)$ to depend on exactly those messages $m_{r's'}$ that start outside of the region r and go into the region r. We define this set of messages $M(r)$ to be those messages $m_{r's'}(x_{s'})$ such that region $r' \backslash s'$ has no nodes in common with region r, and such that region s' is a sub-region of r or the same as region r. We also define the set $M(r,s)$ of messages to be all those messages that start in a sub-region of r and also belong to $M(s)$, and we define $M(r)\backslash M(s)$ to be those messages that are in $M(r)$ but not in $M(s)$.

The canonical generalized belief propagation update rules are:

$$m_{rs} \leftarrow \alpha \left[\sum_{x_{r\backslash s}} \psi_{r\backslash s}(x_{r\backslash s}) \prod_{m_{r''s''} \in M(r)\backslash M(s)} m_{r''s''} \right] / \prod_{m_{r's'} \in M(r,s)} m_{r's'} \quad (46)$$

$$b_r \leftarrow \alpha \psi_r(x_r) \prod_{m_{r's'} \in M(r)} m_{r's'} \quad (47)$$

where for brevity we have suppressed the functional dependences of the beliefs and messages. The messages are updated starting with the messages into the smallest regions first. One can then use the newly computed messages in the product over $M(r,s)$ of the message-update rule. Empirically, this helps convergence.

Claim 2: Let $\{m_{rs}(x_s)\}$ be a set of canonical GBP messages and let $\{b_r(x_r)\}$ be the beliefs calculated from those messages. Then the beliefs are fixed-points of the canonical GBP algorithm if and only if they are zero gradient points of the constrained Kikuchi free energy F_K.

We prove this claim by adding Lagrange multipliers: γ_r to enforce the normalization of b_r and $\lambda_{rs}(x_s)$ to enforce the consistency of each region r with all of its direct sub-regions s. This set of consistency constraints is actually more than sufficient, but there is no harm in adding extra constraints. We then rotate to another set of Lagrange multipliers $\mu_{rs}(x_s)$ of equal dimensionality which enforce a linear combination of the original constraints: $\mu_{rs}(x_s)$ enforces all those constraints involving marginalizations by all direct super-regions r' of s into s except that of region r itself. The rotation matrix is in a block form which can be guaranteed to be full rank. We can then show that the $\mu_{rs}(x_s)$ constraints can be written in the form $\mu_{rs}(x_s) \sum_{r' \in R(\mu_{rs})} c_{r'} \sum_{x_{r'}} b(x'_r)$ where $R(\mu_{rs})$ is the set of all regions which receive the message μ_{rs} in the belief update rule of the canonical algorithm. We then re-arrange the sum over all μ's into a sum over all regions, which has the form $\sum_{r \in R} c_r \sum_{x_r} b_r(x_r) \sum_{\mu_{rs} \in \tilde{M}(r)} \mu_{rs}(x_s)$. ($\tilde{M}(r)$ is a set of $\mu_{r's'}$ in one-to-one correspondence with the $m_{r's'}$ in $M(r)$.) Finally, we differentiate the Kikuchi free energy with respect to $b_r(r)$, and identify $\mu_{rs}(x_s) = \ln m_{rs}(x_s)$ to obtain the canonical GBP belief update rules, Eq. 47. Using the belief update rules in the marginalization constraints, we obtain the canonical GBP message update rules, Eq. 46.

Equation 46 may appear to be complicated, but it can be simply derived by marginalizing equation 47, which is really the key generalized belief propagation equation. Intuitively, equation 47 simply says that the belief in a region depends on the product of all the compatibility matrices and evidence that are internal to the region and all the messages that originate outside of the region and end inside it.

It is clear from this proof outline that other GBP message passing algorithms which are equivalent to the Kikuchi approximation exist. If one writes any set of constraints which are sufficient to insure the consistency of all Kikuchi regions, one can associate the exponentiated Lagrange multipliers of those constraints with a set of messages.

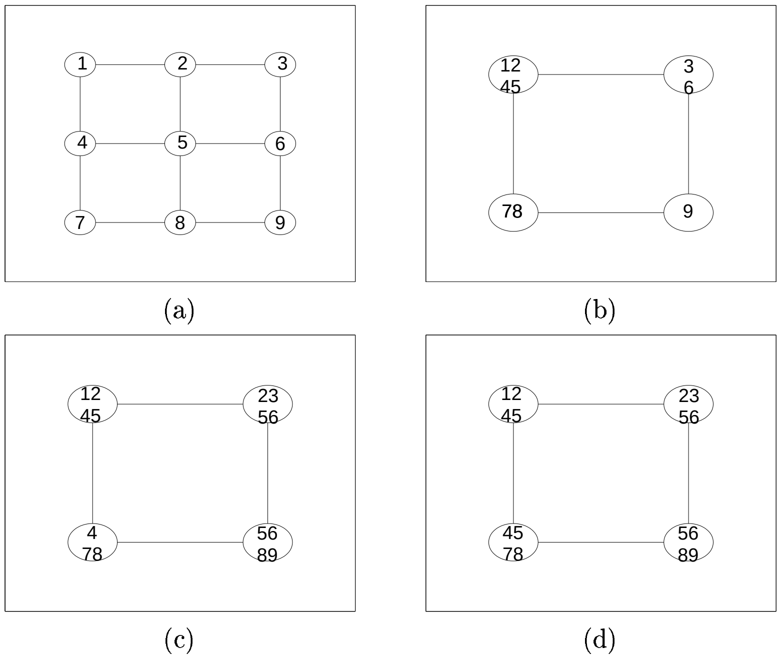


Figure 5: **a.** A simple graph. **b,c,d** Three ways of clustering the graph. Running ordinary BP on the first two clusterings (b,c) is equivalent to a Kikuchi approximation, but not for the third clustering (d)

5 Clustered Belief Propagation

In the GBP algorithm, the messages take the form of probability distributions over clusters of nodes, whereas in ordinary BP the messages take the form of probability distributions over single nodes. A different way of deriving a BP algorithm in which messages are distributions over cluster of nodes is illustrated in figure 5. We first group cluster of nodes into "super-nodes" and then run ordinary BP on the clustered graph. When the clusters are large enough so that the cluster graph is a tree, this is just Pearl's method of clustering for exact inference in any graphical model [13] (equivalent to the junction tree algorithm). When the cluster graph still has loops in it, however, the beliefs are not exact. How are these beliefs related to the Kikuchi approximation?

The simplest case to analyze is when the clusters are non-overlapping as in figure 5b.

Corollary 5: Let G^c be a graph obtained by clustering nodes in a graph G. Assume the potentials in G^c are set so that the joint distribution defined by G^c is equivalent to that defined by G. If the clusters are non-overlapping then the beliefs calculated by running BP on G^c are the same as those calculated by a Kikuchi approximation on G where the regions are all pairs of nodes in G^c.

This corollary follows directly from claim 1.

When the clusters are overlapping, the situation is more complicated. When we run BP on the graph in figure 5c we find that the beliefs are indeed the same as those calculated using the Kikuchi approximation with these clusters as basic regions. When we run BP on the graph in figure 5d, however, we obtain beliefs that are quite different from the Kikuchi beliefs.

One difference between the clusters in figure 5c and these in figure 5d has to do with the overcounting numbers c_r. In figure 5c, the Kikuchi entropy is of the form:

$$\begin{aligned} H &= H(x_{1245}) + H(x_{2356}) + H(x_{5689}) + H(x_{478}) \quad (48) \\ &\quad -H(x_{25}) - H(x_{56}) - H(x_8) - H(x_4) \quad (49) \end{aligned}$$

thus except for the basic clusters, no other region has positive double counting number. This property is not shared by the clusters in figure 5d, however. As equation 42 shows, the region corresponding to the single node x_5 has positive double counting number.

Claim 3: Let R be a set of Kikuchi regions (basic clusters, intersections, intersection of intersections, etc.) with double counting numbers c_r. If $c_r > 0$ only for the basic clusters, then the beliefs calculated using the Kikuchi approximation are identical to those that would be calculated using ordinary BP on a graph G^c whose nodes are the basic clusters.

Proof: To construct the graph G^c we add edges between two regions r_1, r_2 depending on the double counting number of their intersection c_s for $s = r_1 \cap r_2$. If $c_s = -1$ we simply connect the corresponding two regions. If $c_s < -1$ (i.e. there are multiple pairs of regions that have s as their intersection) we choose $|c_s|$ such pairs and connect them. We choose the edges so that all regions whose intersection is s form a connected graph. The compatability matrices $\Psi_{r_1,r_2}(x_{r_1}, x_{r_2})$ are δ functions that require x_{r_1} and x_{r_2} to agree on the value of x_s. Because of this form of the Ψ functions, the message from r_1 to r_2 depends only on the value of their intersection x_s. We now add Lagrange multipliers. If r_1, r_2 are connected in the graph we add two lagrange multipliers $\lambda_{r_1,r_2}(s), \lambda_{r_2,r_1}(x_s)$. The multiplier $\lambda_{r_1,r_2}(s)$ enforces the constraint that $b(x_{r_2})$ marginalize down to $b(x_s)$. If we now define, $\lambda_{r_2,r_1}(x_s) = \ln m_{r_2,r_1}(x_s)$ we find that the stationarity conditions on the free energy and the fixed point equations for the messages are equivalent. $\square$

A simple consequence of claim 3 is that when a set of Kikuchi regions satisfies the condition of claim 3 and the graph G^c is a tree, then the Kikuchi beliefs are exact.

5.1 Normalized Clustered Belief Propagation

In figure 5d, all clusters contain the node 5. Since the clusters are connected in a loop, and the potentials between clusters are δ functions, running BP on figure 5d will lead to "infinite double counting": any evidence a node has about x_5 will come back to it as the messages go around the loop. So if any of the nodes has evidence about x_5 the only fixed-points of BP are these where $b(x_5)$ has all its mass centered on one value of x_5. This suggests that running BP on such graphs is a bad idea.

Normalized clustered BP is an algorithm that fixes this type of "infinite double counting" and turns out to give the same answer as the Kikuchi approximation. Figure 6 illustrates the algorithm. It is equivalent to ordinary BP except that some edges have a "normalizer" module attached to them. As in ordinary clustered BP, messages along each edge correspond to probability distributions over a set of nodes. The idea of the normalizer module is to correct for the double counting of a subset

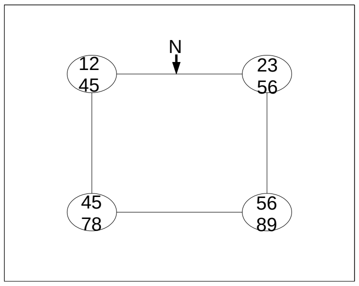


Figure 6: An illustration of the normalized clustered BP algorithm. It is identical to ordinary BP but along some edges a "normalizer" module is added (denoted by an arrow and the letter 'N'). This algorithm is equivalent to the Kikuchi approximation.

of these nodes. In figure 6a there is just one normalizer module and it corrects for $x_t = x_5$.

Nodes calculate their outgoing messages based on incoming messages as in normal BP. We call these messages $m^0_{ij}(x_j)$. In the setting of clustered BP, these messages are functions of the intersection between r_i and r_j so we denote them $m^0_{ij}(s)$ where s is the intersection.

For edges that do not have normalizer modules, these outgoing messages simply become the ingoing messages $m_{ij} = m^0_{ij}$. For edges that have normalizer modules, the messages m_{ij} and m_{ji} are both multiplied by a correction factor that is a function of x_t

$$\begin{aligned} m_{ij}(x_s) &= n_{ij}(x_t) m^0_{ij}(x_s) && (50)\\ m_{ji}(x_s) &= n_{ij}(x_t) m^0_{ji}(x_s) && (51)\\ & && (52)\end{aligned}$$

with:

$$n_{ij}(x_t) = \frac{1}{\sqrt{\sum_{x_s \backslash x_t} m^0_{ij}(x_s) m^0_{ji}(x_s)}} \tag{53}$$

Claim 4: Let R be a set of Kikuchi regions (basic clusters, intersections, intersection of intersections) with double counting numbers c_r. If $c_r > 0$ only for the basic clusters or for intersections of intersections of basic clusters, then the beliefs calculated using the Kikuchi approximation are identical to those that would be calculated using *normalized* BP on a graph G^c whose nodes are the basic clusters.

Proof: We construct the graph in the same way as in claim 3. For every intersection of intersection t that has a a positive c_t we add c_t normalizer module on edges whose intersection contains c_t. We use the same Lagrangian as in claim 3 but corresponding to each normalizer module we add a Lagrange multiplier $\gamma_{r_i,r_j}(x_t)$ that enforces the constraint that $b(x_s)$ (with x_s the intersection of x_{r_i}, x_{r_j}) marginalize down to $b(x_t)$. We now define $\gamma_{r_i,r_j}(x_t) = \ln n_{ij}(x_t)$ with $n(x_t)$ from equation 53 and λ_{ij} as in the proof of claim 3. We find that the stationarity conditions on the free energy and the fixed point equations for the messages are equivalent. □

6 Application to Specific Lattices

We illustrate the canonical GBP algorithm for the Kikuchi approximation of overlapping 4-node clusters on a square lattice of nodes. Figure 7 (a), (b), (c) illustrates the beliefs at a node, pair of nodes, and at a cluster of 4 nodes, in terms of messages propagated in the network. Vectors are the single index messages also used in ordinary BP. Vectors with line segments indicate the double-indexed messages arising from the Kikuchi approximation used here. These can be thought of as correction terms accounting for correlations between messages that ordinary BP treats as independent. (For comparison, Fig. 7 (d), (e), (f) shows the corresponding marginal computations for the triangular lattice with all triangles chosen as the basic Kikuchi clusters).

We find the message update rules by equating marginalizations of Fig. 7 (b) and (c) with the beliefs in Fig. 7 (a) and (b), respectively. Figure 8 (a) and (b) show (graphically) the resulting fixed point equations. The update rule (a) is like that for ordinary BP, with the addition of two double-indexed messages. The update rule for the double-indexed messages involves division by the newly-computed single-indexed messages. Fixed points of these message update equations give beliefs that are stationary points (empirically minima) of the corresponding Kikuchi approximation to the free energy.

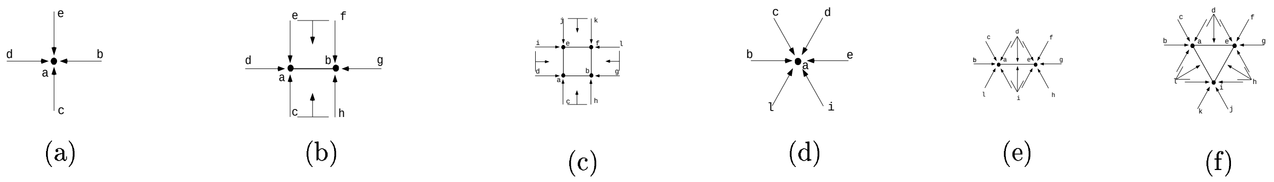


Figure 7: Marginal probabilities in terms of the node links and GBP messages. For (a) node, (b) line, (c) square cluster, using a Kikuchi approximation with 4-node clusters on a square lattice. E.g., (b) depicts (a special case of Eq. 47, written here using node labels): $b_{ab}(x_a, x_b) = \alpha\psi_{ab}(x_a, x_b)\psi_a(x_a)\psi_b(x_b)M_a^c M_a^d M_a^e M_{ab}^{ef} M_b^f M_b^g M_b^h M_{ab}^{ch}$, where super and subscripts indicate which nodes message M goes from and to. (d), (e), (f): Marginal probabilities for triangular lattice with 3-node Kikuchi clusters.

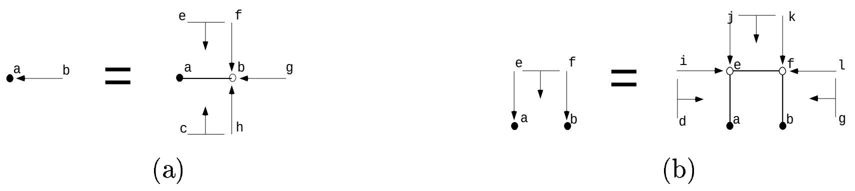


Figure 8: Graphical depiction of message update equations (Eq. 46; marginalize over nodes shown unfilled) for GBP using overlapping 4-node Kikuchi clusters. (a) Update equation for the single-index messages: $M_a^b(x_a) = \alpha \sum_{x_b} \psi_b(x_b)\psi_{ab}(x_a, x_b) M_{ab}^{ef} M_b^f M_b^g M_b^h M_{ab}^{ch}$. (b) Update equation for double-indexed messages (involves a division by the single-index messages on the left hand side).

7 Experimental Results

Ordinary BP is expected to perform relatively poorly for networks with many tight loops, conflicting interactions, and weak evidence. We constructed such a network, known in the physics literature as the square lattice Ising spin glass in a random magnetic field. The nodes are on a square lattice, with nearest neighbor nodes connected by a compatibility matrix of the form $\psi_{ij} = \begin{pmatrix} \exp(J_{ij}) & \exp(-J_{ij}) \\ \exp(-J_{ij}) & \exp(J_{ij}) \end{pmatrix}$ and local evidence vectors of the form $\psi_i = (\exp(h_i); \exp(-h_i))$. To instantiate a particular network, the J_{ij} and h_i parameters are chosen randomly and independently from zero-mean Gaussian probability distributions with standard deviations J and h respectively.

The following results are for n by n lattices with toroidal boundary conditions and with $J = 1$, and $h = 0.1$. This model is designed to show off the weaknesses of ordinary BP, which performs well for many other networks. Ordinary BP is a special case of canonical GBP, so we exploited this to use the same general-purpose GBP code for both ordinary BP and canonical GBP using overlapping square four-node clusters, thus making computational cost comparisons reasonable. We started with randomized messages and only stepped half-way towards the computed values of the messages at each iteration in order to help convergence. We found that canonical GBP took about twice as long as ordinary BP per iteration, but would typically reach a given level of convergence in many fewer iterations. In fact, for the majority of the dozens of samples that we looked at, BP did not converge at all, while canonical GBP always converged for this model and always to accurate answers. (We found that for the zero-field 3-dimensional spin glass with toroidal boundary conditions, which is an even more difficult model, canonical GBP with 2x2x2 cubic clusters would also fail to converge).

For $n = 20$ or larger, it was difficult to make comparisons with any other algorithm, because ordinary BP did not converge and Monte Carlo simulations suffered from extremely slow equilibration. However, generalized belief propagation converged reasonably rapidly to plausible-looking beliefs. For small n, we could compare with exact results, by using Pearl's clustering method on a chain of n by 1 super-nodes. To give a qualitative feel for the results, we compare ordinary BP, canonical GBP, and the exact results for an $n = 10$ lattice where ordinary BP did converge. Listing the values of the one-node marginal probabilities in one of the rows, we find that ordinary BP gives (.0043807, .74502, .32866, .62190, .37745, .41243, .57842, .74555, .85315, .99632), canonical GBP gives (.40255, .54115, .49184, .54232, .44812, .48014, .51501, .57693, .57710, .59757), and the exact results were (.40131, .54038, .48923, .54506, .44537, .47856, .51686, .58108, .57791, .59881).

Acknowledgements

YW acknowledges support by MURI-ARO-DAAH04-96-1-0341, MURI N00014-00-1-0637 and NSF IIS-9988642.

References

[1] H.A. Bethe. *Proc. Roy Soc. London A*, 150:552.

[2] W.T. Freeman and E.C. Pasztor. Learning low level vision. In *Proc. Intl. Conf. Computer Vision*, pages 1182–1189. 1999.

[3] Brendan J. Frey. *Graphical Models for Pattern Classification, Data Compression and Channel Coding.* MIT Press, 1998.

[4] J. Hijmans and J. De Boer. An approximation method for order-disorder problems. *Physica*, pages 471–484, 1955.

[5] Special issue on Kikuchi methods. *Prog. Theor. Phys.*, 115, 1994.

[6] M.I. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. In M.I. Jordan, editor, *Learning in Graphical Models.* MIT Press, 1998.

[7] R. Kikuchi. *Phys Rev*, page 81:988, 1951.

[8] F. R. Kschischang and B. J. Frey. Iterative decoding of compound codes by probability propagation in graphical models. *IEEE Journal on Selected Areas in Communication*, 16(2):219–230, 1998.

[9] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 2001. to appear.

[10] R.J. McEliece, D.J.C. MackKay, and J.F. Cheng. Turbo decoding as as an instance of Pearl's 'belief propagation' algorithm. *IEEE Journal on Selected Areas in Communication*, 16(2):140–152, 1998.

[11] R.J. McEliece, E. Rodemich, and J.F. Cheng. The Turbo decision algorithm. In *Proc. 33rd Allerton Conference on Communications, Control and Computing*, pages 366–379, Monticello, IL, 1995.

[12] K.P. Murphy, Y. Weiss, and M.I. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Proceedings of Uncertainty in AI*, 1999.

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

[14] M.A. Peot and R.D. Shachter. Fusion and propagation with multiple observations in belief networks. *Artificial Intelligence*, 48:299–318, 1991.

[15] T.J. Richardson. The geometry of turbo-decoding dynamics. *IEEE Tran Info Theory*, 46(1):9–23, 2000.

[16] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12:1–41, 2000.

[17] Y. Weiss and W.T. Freeman. On the optimality of solutions of the max-product belief propagation algorithm. *IEEE Transactions on Information Theory*, 2000. to appear.

[18] N. Wiberg. *Codes and decoding on general graphs.* PhD thesis, Department of Electrical Engineering, U. Linkoping, Sweden, 1996.

[19] Y.Kabashima and D. Saad. Belief propagation vs tap for decoding corrupted messages. *Europ. Phys. Lett*, page 44:668, 1998.