Understanding Belief Propagation and its Generalizations

Jonathan S. Yedidia
MERL
201 Broadway
Cambridge, MA 02139
yedidia@merl.com

William T. Freeman*
MERL
201 Broadway
Cambridge, MA 02139
freeman@merl.com

Yair Weiss
School of Computer Science and Engineering
The Hebrew University of Jerusalem
91904 Jerusalem, Israel
yweiss@cs.huji.ac.il

Abstract

"Inference" problems arise in statistical physics, computer vision, error-correcting coding theory, and AI. We explain the principles behind the belief propagation (BP) algorithm, which is an efficient way to solve inference problems based on passing local messages. We develop a unified approach, with examples, notation, and graphical models borrowed from the relevant disciplines.

We explain the close connection between the BP algorithm and the Bethe approximation of statistical physics. In particular, we show that BP can only converge to a fixed point that is also a stationary point of the Bethe approximation to the free energy. This result helps expain the successes of the BP algorithm, and enables connections to be made with variational approaches to approximate inference.

* Present Address: MIT Artificial Intelligence Laboratory, 200 Technology Square, room 725, Cambridge MA 02139.

The connection of BP with the Bethe approximation also suggests a way to construct new message passing algorithms based on improvements to Bethe's approximation introduced by Kikuchi and others. The new generalized belief propagation (GBP) algorithms are significantly more accurate than ordinary BP for some problems. We illustrate how to construct GBP algorithms with a detailed example.

1 INFERENCE AND GRAPHICAL MODELS

We will be describing and explaining an algorithm, called "belief propagation" (BP), which is supposed to solve "inference" problems, at least approximately. Inference problems come up in many different scientific fields, so it is not that surprising that a good algorithm to solve such problems has been repeatedly re-discovered. In fact, one can show that such apparently different methods as the forward-backward algorithm, the Viterbi algorithm, iterative decoding algorithms for Gallager codes and turbocodes, Pearl's belief propagation algorithm for Bayesian networks, the Kalman filter, and the transfer-matrix approach in physics are all special cases of the BP algorithm discovered in different scientific communities (Aji and McEliece 2000; Kschischang, Frey, and Loeliger 2001).

We will emphasize that a multi-disciplinary approach yields important insights into the BP algorithm. We therefore begin with a small survey of "inference" problems from the AI, computer vision, statistical physics, and digital communications literatures, which will also give us a chance to introduce the different kinds of graphical models that are used to describe these problems. This first section is a basic review of well-known material, but we hope that it will be useful in elucidating the deep similarity between problems in these different fields. The reader interested in other reviews of this material may want to consult (Kschischang, Frey, and Loeliger 2001) and (Frey 1998).

1.1 BAYESIAN NETWORKS

In the AI literature, "Bayesian networks" are probably the most popular type of graphical model (Pearl 1988; Jensen 1996). They are used in expert systems involving problem domains such as medical diagnosis, map learning, language understanding, heuristic search, and so on. We will take as an example the medical diagnosis problem. Suppose that we want to construct a machine which will automatically give diagnoses for patients. For each patient, we will have some (possibly incomplete) information, such as his symptoms and test results, and we would like to *infer* the probability that a given disease or set of diseases is causing his symptoms. We also assume that we know (presumably from expert advice) the statistical dependencies between different symptoms, test results, and diseases. For example, let us consider the fictional "Asia" example of Lauritzen and Spiegelhalter (Lauritzen and Spiegelhalter 1988), shown in figure 1.

In words, the qualitative statistical dependencies shown in this small Bayesian network can be described as follows:

1. A recent trip to Asia (A) increases the chances of tuberculosis (T).
2. Smoking (S) is a risk factor for both lung cancer (L) and bronchitis (B).

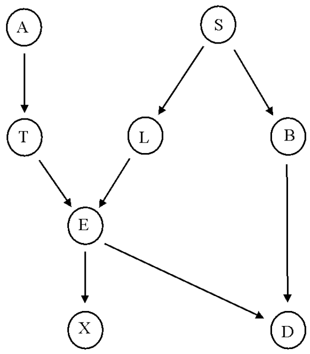

Figure 1 The fictional "Asia" Bayesian network, taken from (Lauritzen and Spiegelhalter 1988).

3. The presence of either (E) tuberculosis or lung cancer can be detected by an X-ray result (X), but the X-ray alone cannot distinguish between them.
4. Dyspnoea (D) (shortness of breath) may be caused by bronchitis (B), or either (E) tuberculosis or lung cancer.

Each node represents a variable that can be in a discrete number of possible states. We write x_i for the variable representing the different possible states of node i. In addition to the qualitative dependencies described by the Bayesian network graph, there are quantitive statistical relationships that we assign to each arrow in the graph. Associated with each arrow is a conditional probability: for example, we write $p(x_L|x_S)$ for the conditional probability of a patient having lung cancer given that he does or does not smoke. For this link, we say that the S node is the "parent" of the L node because x_L is conditionally dependent on x_S. Some nodes like the D node might have more than one parent, in which case we define their conditional probabilities in terms of all their parents; thus we write $p(x_D|x_E, x_B)$ for the conditional probability of having dyspnoea.

Note that the Bayesian network defines an independency structure: the probability that a node is in one of its states depends directly only on the states of its parents. For nodes like A or S that do not have any parents, we introduce probabilities like $p(x_S)$ that are not conditioned on any other nodes. In general, a Bayesian network (and the other graphical models that we consider) is most useful if it is sparse, which means that most of the nodes do not have a direct statistical dependence.

In our example, the over-all or "joint" probability $p(\{x\}) \equiv p(x_A, x_S, x_T, x_L, x_B, x_E, x_X, x_D)$ that the patient has some combination of symptoms, test-results, and diseases, is just the product of all the probabilities of the parent nodes and all the conditional probabilities:

$$p(\{x\}) = p(x_A)p(x_S)p(x_T|x_A)p(x_L|x_S)p(x_B|x_S)p(x_E|x_L, x_T)p(x_D|x_B, x_E)p(x_X|x_E) \quad (1)$$

More generally, a Bayesian network is a directed acyclic graph of N random variables x_i that defines a joint probability function

$$p(x_1, x_2, ..., x_N) = \prod_{i=1}^{N} p(x_i | Par(x_i)) \tag{2}$$

where $Par(x_i)$ denotes the states of the parents of node i, and if node i has no parents, we take $p(x_i|Par(x_i)) = p(x_i)$. By "directed acyclic graph," we mean that the arrows do not loop around in a cycle–it is still possible that the links form a loop when one ignores the arrows.

Our goal will be to compute certain *marginal* probabilities. For example, we might want to compute the probability that a patient has a certain disease. By "inference," we simply mean the computation of these marginal probabilities. Mathematically, marginal probabilities are defined in terms of sums over all the possible states of all the other nodes in the system. For example if we want the marginal probability of the last node $p(x_N)$, we in general need to compute

$$p(x_N) = \sum_{x_1} \sum_{x_2} ... \sum_{x_{N-1}} p(x_1, x_2, x_3, ..., x_N) \tag{3}$$

We will refer to marginal probabilities that we compute approximately as "beliefs," and denote the belief at node i by $b(x_i)$.

If we have some information about some of the nodes (e.g. we know that the patient does not smoke in our "Asia" example), then we will be able to fix the corresponding variable and we will not have to sum over the unknown states of that node. We will call such a node an "observable" node, in contrast to the other nodes which are "hidden" nodes. For all our graphical models, we will denote observable nodes by filled-in circles, and hidden nodes by empty circles.

For small Bayesian networks, we can easily do marginalization sums directly, but unfortunately, the number of terms in the sums will grow exponentially with the number of hidden nodes in the network. The virtue of the BP algorithm is that we can use it to compute marginal probabilities, at least approximately, in a time that grows only linearly with the number of nodes in the system. For that reason, BP can be used in practice as an "inference engine" acting on the statistical data encoded in a large Bayesian network. Before we turn to BP, however, we describe some other inference problems.

1.2 PAIRWISE MARKOV RANDOM FIELDS

BP has also recently begun to be used as an "engine" for low-level computer vision problems (Freeman, Pasztor, and Carmichael 2000). Humans often take for granted the solution of apparently simple computer vision problems like the segmentation and recognition of objects, or the detection and interpretation of motion. We solve these tasks so automatically that it can be surprising how difficult it is to teach a computer to solve the same tasks, given just a series of two-dimensional arrays of pixel values. A key to progress in computer vision is to find theoretically solid models that are computationally tractable.

Pairwise Markov random fields (MRF's) provide attractive theoretical models for some computer vision problems (Geman 1984). In such problems, we normally want to infer a representation of whatever is "really out there" from the data that we are given, which is ultimately a two-dimensional array of numbers representing pixel intensities. To take a concrete example, suppose that we want to infer the distance of the objects in a scene from the viewer. That is, imagine that we are given a 1000 by 1000 gray-scale image, and we pose our problem as trying to infer distance values d_i corresponding to intensity values I_i, where i ranges over the million possible pixel positions. Or, instead of distance, we might be inferring some other quantity about the scene–such as high-resolution details that are missing in the image, or the optical flow in a series of images, etc.

In general, we assume that we observe some quantities about the image y_i, and that we want to infer some other quantities about the underlying scene x_i. The indices i could represent single pixel positions, or they might represent the position of a small patch of pixels. We further assume that there is some statistical dependency between x_i and y_i at each position i, which we write as a joint compatability function $\phi_i(x_i, y_i)$. The function $\phi_i(x_i, y_i)$ is often called the "evidence" for x_i. Finally, for us to possibly be able to infer anything about the scene, there has to be some structure to the x_i. In general, if we do not assume such structure, computer vision problems are inherently hopelessly ill-posed. We encode the assumed structure of the scene by saying that the nodes i are arranged in a two-dimensional grid, and scene variables x_i should, insofar as possible, be "compatible" with nearby scene variables x_j, as represented by a compatibility function $\psi_{ij}(x_i, x_j)$, where ψ_{ij} only connects nearby positions. We then take the overall joint probability of a scene x_i and an image y_i to be

$$p(\{x\}, \{y\}) = \frac{1}{Z} \prod_{(ij)} \psi_{ij}(x_i, x_j) \prod_i \phi_i(x_i, y_i) \tag{4}$$

where Z is a normalization constant and the product over (ij) is over (for example) nearest neighbors on the square lattice.

A graphical depiction of this model is shown in figure 2. The filled-in circles represent

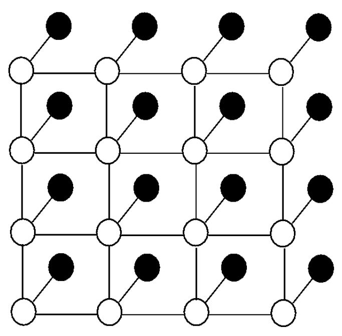

Figure 2 A square lattice pairwise Markov Random Field.

the observed image nodes y_i, while the empty circles represent the "hidden" scene nodes

x_i. The Markov random field is said to be "pairwise" because the compatibility functions only depend on pairs of sites i and j. In contrast to Bayesian networks, this graphical model is undirected. There is no notion, as is usually implicit in a Bayesian network, that the variable at one node x_i is a causal "parent" of its neighbor x_j, so we use undirected compatibility functions $\psi_{ij}(x_i, x_j)$ instead of conditional probability functions $p(x_i|x_j)$. Nevertheless, our agenda in doing inference will be very similar: we will want to compute the beliefs $b(x_i)$, for all positions i, so as to be able to infer something about the underlying unknown scene. Once again a direct computation of marginal probabilities would take exponential time (and the number of nodes is typically very large) so we need a faster algorithm like BP. We note in passing that for a restricted class of pairwise MRF's relevant to computer vision, fast algorithms based on "graph cuts" (Boykov, Veksler, and Zabih 2001) can also be used to estimate hidden states.

1.3 THE POTTS AND ISING MODELS

It is worth taking a small detour to show that the pairwise MRF described by equation (4) can easily be brought into a form recognizable to physicists as the "Potts model" (Baxter 1982). Let us define the "interaction" $J_{ij}(x_i, x_j)$ between the variables at neighboring nodes by $J_{ij}(x_i, x_j) = \ln \psi_{ij}(x_i, x_j)$ and the "field" $h_i(x_i)$ at each node by $h_i(x_i) = \ln \phi(x_i, y_i)$. (Because we do inference for a given set of y_i, we can in fact consider the y_i as fixed variables and subsume them into our definition of $h_i(x_i)$.) If we now define the Potts model "energy" as

$$E(\{x\}) = -\sum_{(ij)} J_{ij}(x_i, x_j) - \sum_i h_i(x_i) \tag{5}$$

and appeal to Boltzmann's law from statistical mechanics:

$$p(\{x\}) = \frac{1}{Z} e^{-E(\{x_i\})/T} \tag{6}$$

we see that our pairwise MRF corresponds exactly to a Potts model at a temperature T equal to 1. The normalization constant Z is known in the physics literature as the "partition function." If the number of states at each node is exactly two, the model is called the "Ising model." In this case, physicists sometimes prefer to change variables from x_i to s_i which can take on the values of 1 or -1, and to further restrict themselves to J_{ij} interactions which have a symmetric form that can be written in terms of a "spin glass" energy function (Mezard, Parisi, and Virasoro 1987)

$$E(\{s\}) = -\sum_{(ij)} J_{ij} s_i s_j - \sum_i h_i s_i \tag{7}$$

In the context of the Ising model, the inference problem of computing beliefs $b(x_i)$ can be mapped onto the physics problem of computing local "magnetizations"

$$m_i \equiv b(s_i = 1) - b(s_i = -1). \tag{8}$$

1.4 TANNER GRAPHS AND FACTOR GRAPHS

Another problem for which the BP algorithm has given excellent results is the iterative decoding of error-correcting codes (Frey and Mackay 1998; McEliece, MacKay, and Cheng

1998). BP is the decoding algorithm used to decode some of the best practical codes, including turbocodes (Berrou, Glavieux, and Thitimajshima 1993) and Gallager codes (Gallager 1963). Decoding error-correcting codes is in fact an elegant example of an "inference" problem–the receiver of the coded message that has been corrupted by a noisy channel is trying to *infer* the message that was initially transmitted. (For general backround on error-correcting coding, see (Gallager 1968).)

Gallager codes and turbocodes can be formulated as *parity-check codes* (Mackay 1999). In a block parity-check code, we will typically try to send k information bits in a block of N bits. N will be greater than k so as to provide redundancy that can be used to recover from errors induced by the noisy channel. Let us give a small example: an $N = 6$, $k = 3$ binary code, shown by its "Tanner graph" in figure 3. A Tanner graph (Tanner

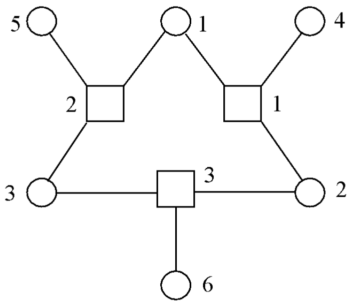

Figure 3 A Tanner graph for a small $N = 6$, $k = 3$ parity check code.

1981) is a pictorial representation of the parity-check constraints on the codewords (legal configurations of bits) of an error-correcting code. Each square represents a parity check, and each circle connected to a square represents a bit that participates in that parity check. In our example code, the first parity check forces the sum of bits #1, #2, and #4 to be even, while second parity check forces the sum of bits #1, #3, and #5 to be even, and the third parity check forces the sum of bits #2, #3, and #6 to be even. The only eight codewords that satisfy these three parity-check constraints are `000000`, `001011`, `010101`, `011110`, `100110`, `101101`, `110011`, and `111000`. Note that for this code, the first three bits are the "information" bits, so if you are transmitting messages, and you intend, for example, to transmit the `010` message, you will use those bits as the first three bits of your codeword, and then the remaining bits are uniquely determined (for that case you would transmit the `010101` codeword.)

The recipient of a coded message, after it has been corrupted by a noisy channel, may find that the received message is not a codeword after all. For example, let us suppose that the `010101` message was transmitted, but because of the noisy channel, one of the bits was flipped and the `011101` "word" was received. This word does not correspond to any codeword, and it is the job of the decoding algorithm to infer which codeword was actually sent. For our code, we can readily see that the `010101` codeword is the only one that is only a single bit-flip away from the received word, so if the bit-flip probability is small, it is reasonable to decode to that codeword. Unfortunately, for many codes, N and k will be large, so that the number of codewords is exponentially huge and we cannot simply resort to examining all the codewords and finding the closest one. As is by now becoming a familiar refrain, we will be able to turn to the BP algorithm, because it will work in a time that only grows linearly with N.

The decoding problem can be given a probabilistic formulation. We assume that we have received a sequence of N bits y_i, and we are trying to find the N bits of the transmitted codeword x_i. Let us, for the purpose of simplicity, assume that the noisy channel is "memoryless," that is each bit i is flipped independently of every other bit. That means that we can associate a conditional probability $p(x_i|y_i)$ with each received bit. For example, if the first bit is received as a `0`, and we know that the bits are flipped with a probability f, then the probability that the first bit was transmitted as a `0` is $1-f$, while the probability that it was transmitted as a `1` is f. So $p(x_1 = 0|y_1 = 0) = (1 - f)$, while $p(x_1 = 1|y_1 = 0) = f$. The overall probability of each codeword is proportional to the product of these one-node probabilities: $\prod_{i=1}^{N} p(x_i|y_i)$. But to make sure that we only consider combinations of x_i that are codewords, we write a a joint probability function that combines these conditional probabilities with the parity check constraints. For example, for our $N = 6$, $k = 3$ code given above, we write the overall joint probability distribution as

$$p(\{x\},\{y\}) = \frac{1}{Z}\psi_{124}(x_1,x_2,x_4)\psi_{135}(x_1,x_3,x_5)\psi_{236}(x_2,x_3,x_6)\prod_{i=1}^{6} p(y_i|x_i). \tag{9}$$

In the above equation, functions like $\psi_{124}(x_1,x_2,x_4)$ are parity check functions. They have the value 1 if the sum of their arguments is even, and have the value 0 if the sum of their arguments is odd.

In general, for a parity check code with transmitted bits x_i and received bits y_i, and $N-k$ parity checks, we can write the joint probability distribution as

$$p(\{x\},\{y\}) = \frac{1}{Z}\prod_{a=1}^{N-k}\psi_a(\{x\}_a)\prod_{i=1}^{N} p(y_i|x_i) \tag{10}$$

where we have used the notation $\psi_a(\{x\}_a)$ to denote the ath parity check function ψ_a and its arguments $\{x\}_a$.

A decoding algorithm for parity check codes that minimizes the number of bits that are decoded incorrectly is to compute the marginal probabilities $p(x_i)$ for every bit i, and then to threshold each bit to its most probable value. Note that taking the most probable value of each bit independently is the strategy that is guaranteed to minimize the number of incorrectly decoded bits even though it may not yield a valid codeword (Gallager 1968). As usual, a direct computation of the marginal probabilities will take an exponentially huge time, so we resort to the efficient BP algorithm. The BP algorithm, while not exact for Gallager codes or turbocodes, is very effective in practice, and has been used to achieve near-Shannon limit performance (Mackay 1999).

A *factor graph* (Kschischang, Frey, and Loeliger 2001) is a generalization of a Tanner graph, whose graph looks essentially identical, but now each square can represent *any* function of its variables (the nodes it is attached to). Each function can have varying numbers of variables, including just one. We take the joint probability distribution of a factor graph of N variables with M functions to be

$$p(\{x\}) = \frac{1}{Z} \prod_{a=1}^{M} \psi_a(\{x\}_a) \tag{11}$$

Note that we subsume any "observed" nodes y_i into functions that they generate of the "hidden" nodes x_i.

1.5 CONVERTING GRAPHICAL MODELS

It is very easy to convert arbitrary pairwise MRF's or Bayesian networks into equivalent

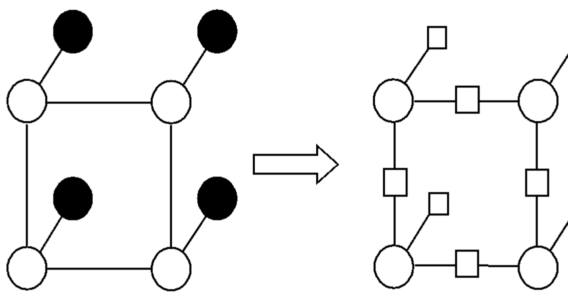

Figure 4 Converting a pairwise MRF into a factor graph.

factor graphs. In figure 4 we show a small pairwise MRF and the equivalent factor graph. Note that instead of using "observable" nodes, we introduce equivalent factor graph functions of a single variable. The factor graph functions $\psi_a(\{x\}_a)$ will be equivalent to either the two node functions $\psi_{ij}(x_i, x_j)$ if they link two "hidden" nodes, or the single node functions $\phi_i(x_i, y_i)$ if they are attached to a single "hidden" node.

In figure 5 we give a small Bayesian network and the equivalent factor graph. Notice that there is always a factor graph function linking a node and all of its parents, because that is the form of the joint probability distribution in a Bayesian network. The factor graph functions $\psi_a(\{x\}_a)$ directly correspond to the Bayesian network probabilities $p(x_i|Par(x_i))$.

Finally, in figure 6, we give the factor graph representation of the $N = 6$, $k = 3$ code that we discussed above. The received bits y_i are represented in terms of the functions that they induce on the corresponding bits x_i. There are also factor graph functions corresponding to the original parity check functions.

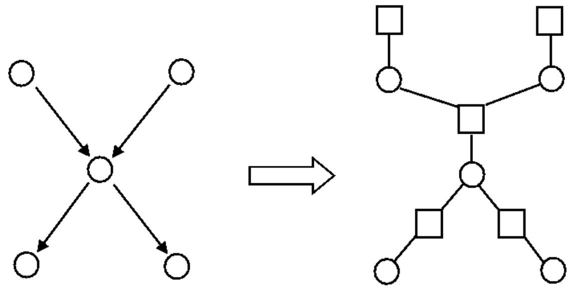

Figure 5 Converting a Bayesian Network into a factor graph.

Belief propagation algorithms specific to Bayesian networks, pairwise MRF's, and factor graphs have all been developed, and they are all mathematically equivalent, but it would certainly confusing from the pedagogical point of view to present all these different versions. In fact, as we have just seen, arbitrary Bayesian networks and pairwise MRF's can be easily converted into equivalent factor graphs, and as we shall see in a moment, one can also convert arbitrary factor graphs into equivalent Bayesian networks or pairwise MRF's. We can thus choose to work with Bayesian networks, pairwise MRF's, or factor graphs without losing any generality.

We will choose to focus on pairwise MRF's. Our reason for this choice is purely pedagogical–although all the different BP algorithms are mathematically equivalent, the version of BP for pairwise MRF's is somewhat simpler in form because it has only one kind of message, while the BP algorithms for the other graphical models are normally described using two kinds of messages. The need for two kinds of messages arises either because of the arrows in Bayesian networks, or the two kinds of nodes in factor graphs. In any case, the BP algorithm that one derives by using the pairwise MRF formulation on a converted Bayesian network or factor graph is precisely mathematically equivalent at every iteration to the BP algorithm as it is normally described for the other graphical models, so we will actually lose nothing at all by using this description.

The reader who is impatient to learn about BP can now skip ahead to the next section, but for the purposes of completeness we explain how to convert arbitrary factor graphs into equivalent Bayesian networks and pairwise MRF's. To convert a factor graph into an equivalent Bayesian network, convert every function node into an "observable" node that is observed to be in its first state. Then make the probability of the first state of

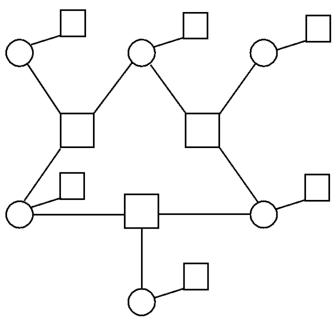

Figure 6 The factor graph representing the joint probability function of a small parity check code.

that observable node, given its parents, equivalent to the factor graph function that was eliminated (see figure 7.)

We show the conversion of a factor graph into a pairwise MRF graphically in figure 8. Each function in the factor graph is converted into a hidden node in a pairwise MRF with an observable node hanging off of it. For example, a function $\psi_a(x_i, x_j, x_k)$ of three nodes in a factor graph will be represented by a new hidden node x_a and a new observable node y_a. The node x_a can be in as many states as the product of all the variables of the corresponding function ψ_a. For example, if the factor graph function that we were converting was $\psi_a(x_i, x_j, x_k)$, and x_i, x_j, and x_k were each binary nodes taking on values `0` or `1`, then the new node x_a could take on eight values ranging from `000` to `111`. The "evidence" $\phi(x_a, y_a)$ is set to correspond to the factor graph function $\psi(\{x\})$. For example, if $\psi(\{x\})$ is a parity check function, than $\phi(x_a, y_a)$ will just be equal to 1 if x_a corresponds to a state with an even number of ones and 0 otherwise. Finally, we need to introduce the notation $x_a(i)$ to denote the state that the ith node should be in to correspond with the a node being in state x_a. The compatibility functions $\psi_{ai}(x_a, x_i)$ between a new node x_a and an ordinary node x_i should be set to one if $x_a(i) = x_i$ and zero otherwise.

2 STANDARD BELIEF PROPAGATION

For the reasons given in the preceding section, we will focus on pairwise MRF's. We can consider the "observed" nodes y_i to be fixed, write $\phi_i(x_i)$ as a short-hand for $\phi_i(x_i, y_i)$, and focus on the joint probability distribution for the unknown variables x_i:

$$p(\{x\}) = \frac{1}{Z} \prod_{(ij)} \psi_{ij}(x_i, x_j) \prod_i \phi_i(x_i) \tag{12}$$

In the BP algorithm, we introduce variables such as $m_{ij}(x_j)$, which can intuitively be understood as a "message" from a hidden node i to the hidden node j about what state node j should be in (see figure 9). The message $m_{ij}(x_j)$ will be a vector of the same

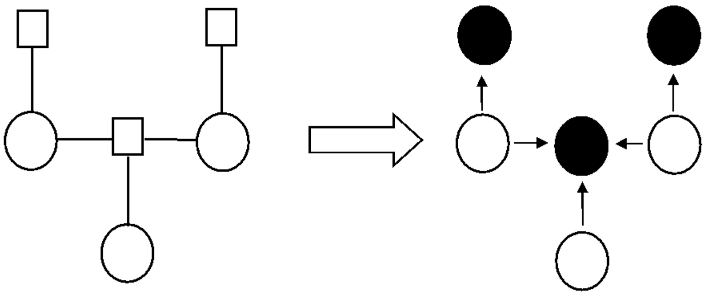

Figure 7 Converting a factor graph into a Bayesian network.

dimensionality as x_j, with each component being proportional to how likely node i thinks it is that node j will be in the corresponding state. In the BP algorithm, the belief at a node i is proportional to the product of the local evidence at that node ($\phi_i(x_i)$), and all the messages coming into node i:

$$b_i(x_i) = k\phi_i(x_i) \prod_{j \in N(i)} m_{ji}(x_i) \tag{13}$$

where k is a normalization constant (the beliefs must sum to 1) and $N(i)$ denotes the nodes neighboring i (see figure 10.) The messages are determined self-consistently by the message update rules:

$$m_{ij}(x_j) \leftarrow \sum_{x_i} \phi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k \in N(i)\backslash j} m_{ki}(x_i). \tag{14}$$

Note that on the right-hand-side, we take the product over all messages going into node i except for the one coming from node j. The message-update rule is shown diagramatically in figure 11.

Where do these rules come from? It is not too hard to show that these rules give beliefs that are exact if the pairwise MRF is "singly-connected"; that is, if there are no loops in the pairwise MRF. We will not give a proof here, but just a small example that might help convince the reader. Consider the network with four hidden nodes shown if figure 12. We will compute the belief at node 1 using the belief propagation rules. We have

$$b_1(x_1) = k\phi_1(x_1)m_{21}(x_1) \tag{15}$$

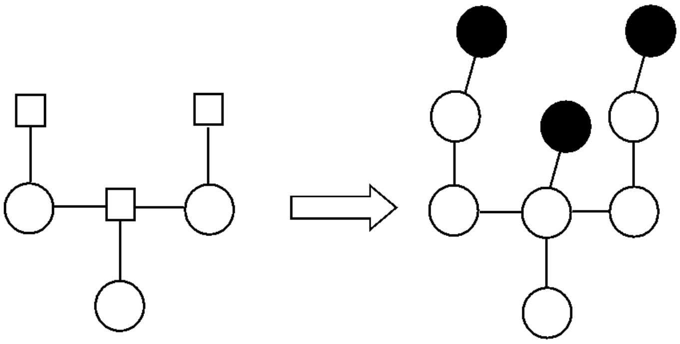

Figure 8 Converting a factor graph into a pairwise MRF.

Using the message-update rules for $m_{21}(x_1)$, we find

$$b_1(x_1) = k\phi_1(x_1)\sum_{x_2}\psi_{12}(x_1,x_2)\phi_2(x_2)m_{32}(x_2)m_{42}(x_2) \tag{16}$$

Using the message-update rules for $m_{32}(x_2)$ and $m_{42}(x_2)$ we find

$$b_1(x_1) = k\phi_1(x_1)\sum_{x_2}\psi_{12}(x_1,x_2)\phi_2(x_2)\sum_{x_3}\phi_3(x_3)\psi_{23}(x_2,x_3)\sum_{x_4}\phi_4(x_4)\psi_{24}(x_2,x_4) \tag{17}$$

Finally, by reorganizing the sums, it is easy to se that the belief at node 1 is the same as the exact marginal probability at node 1:

$$b_1(x_1) = k\sum_{x_2,x_3,x_4} p(\{x\}) = p_1(x_1) \tag{18}$$

It is easy to convince oneself, and to prove, that BP in fact gives the exact marginal probabilities for all the nodes in any singly-connected graph.

In a practical computation, one starts with the nodes at the edge of the graph, and only computes a message when one has available all the messages necessary. Thus, in our example, one would start with m_{32} and m_{42}, and then compute m_{21}, and then finally compute b_1. In general, it is easy to see that each message need only be computed once for singly connected graphs. That means that the whole computation takes a time proportional to the number of links in the graph, which is dramatically less than the exponentially large time that would be required to compute marginal probabilities naively.

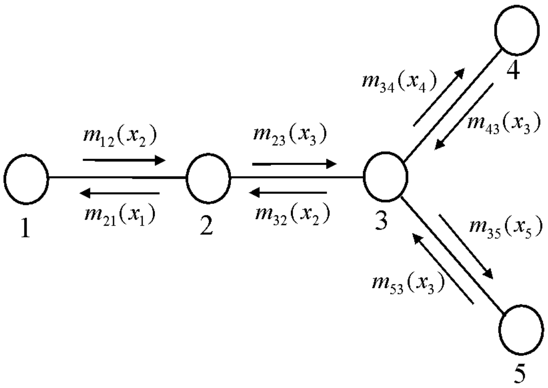

Figure 9 An illustration of the messages passed in BP. $m_{ij}(x_j)$ can be thought of as a "message" from a hidden node i to the hidden node j about what state node j should be in.

The point of view illustrated by this example suggests that belief propagation is a way of organizing the "global" computation of marginal beliefs in terms of smaller local computations. The message flow is akin to the flow of a river, and each message summarizes all the computations that occurred further upstream along the branches that feed into that message. Thus in our example, $m_{21}(x_1)$ is a summary of all the computations that occured at nodes 2,3, and 4. This is the classical point of view, and it seems to suggest that it must be very important that there be no loops in the pairwise MRF, for otherwise the entire argument for the exactness of BP breaks down.

So for the time being, let us continue our discussion of singly-connected (loop-free) pairwise MRF's. We will find it very convenient to introduce the two-node marginal probabilities $p_{ij}(x_i, x_j)$, for two neighboring sites i and j, which are obtained by marginalizing the joint probability function over every node except the two nodes i and j:

$$p_{ij}(x_i, x_j) \equiv \sum_{z:z_{ij}=(x_i,x_j)} p(\{z\}). \tag{19}$$

We write a belief equation for the two-node beliefs $b_{ij}(x_i, x_j)$ analogously to equation (13) for the one-node beliefs:

$$b_{ij}(x_i, x_j) = k\psi_{ij}(x_i, x_j)\phi_i(x_i)\phi_j(x_j) \prod_{k \in N(i)\backslash j} m_{ki}(x_i) \prod_{l \in N(i)\backslash i} m_{lj}(x_j) \tag{20}$$

where $k \in N(i)\backslash j$ means that we consider all neighbors k of node i except for node j. We describe this equation diagrammatically in figure 13. It can be justified on singly-connected graphs by working out that for such graphs the two-node beliefs will correspond to the exact two-node marginal probabilities.

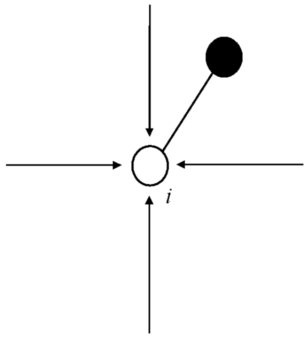

Figure 10 A diagrammatic representation of the BP belief equation $b_i(x_i) = k\phi_i(x_i)\prod_{j\in N(i)} m_{ji}(x_i)$.

It is also worth noting that that by combining equations (13) and (20) with the marginalization condition

$$b_i(x_i) = \sum_{x_j} b_{ij}(x_i, x_j) \tag{21}$$

we consistently derive the message update rules (14). (See figure 14 for a diagrammatic version of this derivation.)

The BP algorithm, as defined in terms of the belief equations (13) and (20), and the message-update rules (14), does not make reference to the topology of the graph that it is running on. Thus, there is nothing to stop us from implementing it on a graph that has loops. One starts with some initial set of messages (one usually begins with completely unbiased messages), and simply iterates the message-update rules until they (possibly) converge, and then one can read off the approximate beliefs from the belief equations. But one would not necessarily expect the algorithm to work well. As Pearl warned, "if we ignore the existence of loops and permit the nodes to continue communicating with each other as if the network were singly connected, messages may circulate indefinitely around these loops, and the process may not converge to a stable equilibrium" (Pearl 1988). One can indeed find examples of graphical models with loops, where, for certain parameter values, the BP algorithm fails to converge, or predicts beliefs that are inaccurate (Murphy, Weiss, and Jordan 1999). On the other hand, the BP algorithm has been successful as a decoding algorithm for error-correcting codes defined on Tanner graphs that have loops (Frey and Mackay 1998), and also for some computer vision problems where the underlying MRF if full of loops (Freeman, Pasztor, and Carmichael 2000). In the next section, we explain why this success might not be so surprising after all.

3 FREE ENERGIES

In this section, we will introduce the Bethe approximation to the free energy and show that the fixed points of the BP algorithm correspond to the stationary points of the Bethe free

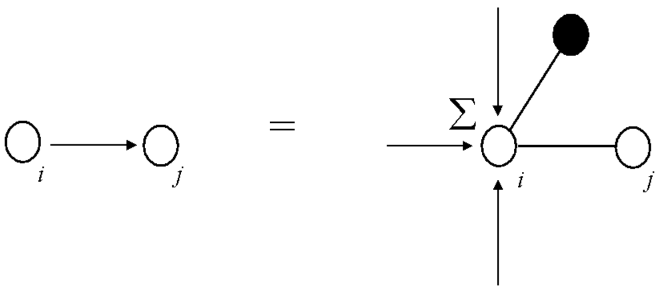

Figure 11 A diagrammatic representation of the BP message update rules $m_{ij}(x_j) \leftarrow \sum_{x_i} \phi_i(x_i)\psi_{ij}(x_i,x_j)\prod_{k\in N(i)\backslash j} m_{ki}(x_i)$. The summation symbol indicates that we are summing over all the states of node i.

energy. First we want to explain, to readers who might not have a physics background, the concept of a "free energy." (See also (Yedidia 2001) for more background.) All of our graphical models define a joint probability function $p(\{x\})$. If we have some other approximate joint probability function $b(\{x\})$, we can define a "distance" (known as the Kullback-Leibler distance) between $p(\{x\})$ and $b(\{x\})$ by

$$D\left(b(\{x\})||p(\{x\}\right) = \sum_{\{x\}} b(\{x\}) \ln \frac{b(\{x\})}{p(\{x\})} \tag{22}$$

The Kullback-Liebler distance does not have all the properties we normally associate with distances: it is not symmetric and does not satisfy the triangle inequality. Nevertheless, it is useful because it is always non-negative and it is zero if and only if the two probability functions $b(\{x\})$ and $p(\{x\})$ are equal (Cover and Thomas 1991).

Statistical physicists generally assume that Boltzmann's law $p(\{x\}) = \frac{1}{Z}e^{-E(\{x\})/T}$ is true. In our context, we can just consider it a tautology, defining the "energy" $E(\{x\})$. Furthermore, from our point of view, the "temperature" T is just a parameter that defines a scale of units for the energy, and for simplicity, we can choose our units so that $T = 1$. Substituting Boltzmann's law into our distance measure, and setting $T = 1$, we find that

$$D\left(b\{x\}||p(\{x\}\right) = \sum_{\{x\}} b(\{x\})E(\{x\}) + \sum_{\{x\}} b(\{x\}) \ln b(\{x\}) + \ln Z \tag{23}$$

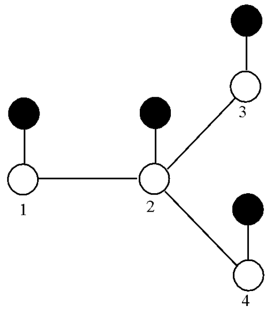

Figure 12 A pairwise MRF with four hidden nodes.

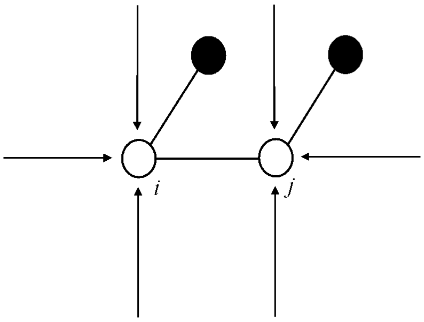

Figure 13 A diagrammatic representation of the BP two-node belief equation $b_{ij}(x_i, x_j) = k\psi_{ij}(x_i, x_j)\phi_i(x_i)\phi_j(x_j) \prod_{k \in N(i)\backslash j} m_{ki}(x_i) \prod_{l \in N(i)\backslash i} m_{lj}(x_j)$.

So we see that the Kullback-Leibler distance will be zero, and therefore the approximate probability function $b(\{x\})$ will equal to the exact probability function $p(\{x\})$, when the quantity

$$G\left(b(\{x\})\right) = \sum_{\{x\}} b(\{x\})E(\{x\}) + \sum_{\{x\}} b(\{x\}) \ln b(\{x\}) = U(b\{x\}) - S(b\{x\}) \tag{24}$$

achieves its minimal value of $F \equiv -\ln Z$. F is called the "Helmholz free energy," while the more important functional $G(b(\{x\}))$ unfortunately does not have a name that is universally agreed upon; we will use the name "Gibbs free energy." The first term in the Gibbs free energy is called the "average energy" U, while the second is the negative of the "entropy" S.

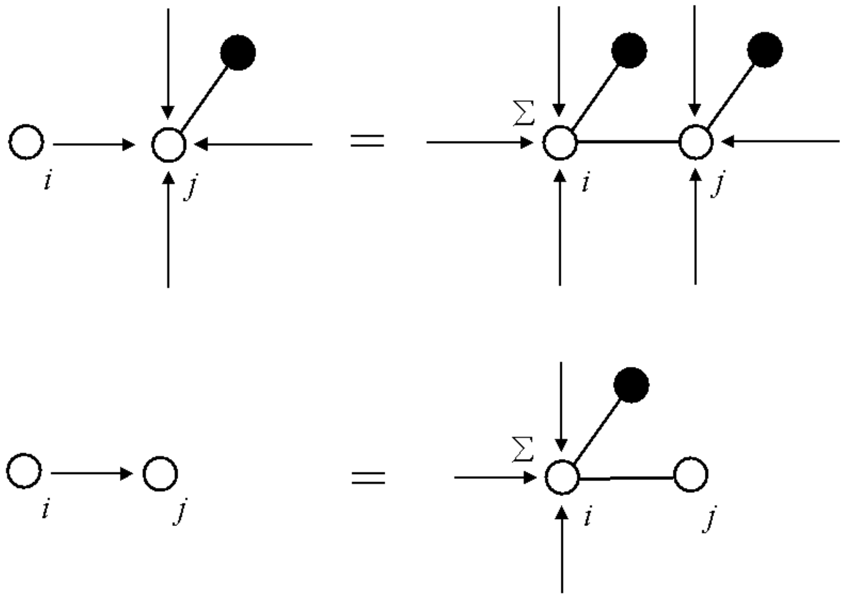

Figure 14 The top equation is a diagrammatic version of the equation $b_j(x_j) = \sum_{x_i} b_{ij}(x_i, x_j)$. By cancelling the pieces that are common on both sides of the equation, we derive the message-update rules $m_{ij}(x_j) \leftarrow \sum_{x_i} \phi_i(x_i)\psi_{ij}(x_i, x_j)\prod_{k\in N(i)\backslash j} m_{ki}(x_i)$, shown below.

3.1 THE MEAN-FIELD FREE ENERGY

Why is it useful to describe a system in terms of its Gibbs free energy G instead of directly in terms of its joint probability distribution? One reason is that it is often possible to make progress by constructing analytically tractable approximations to G (Yedidia 2001). For example, let us return to pairwise MRF's and restrict ourselves to consider approximate joint probability distributions $b(\{x\})$ that have a particularly simple form: they are factorized over the sites:

$$b(\{x\}) = \prod_i b_i(x_i) \tag{25}$$

where the $b_i(x_i)$ are subject to the constraint $\sum_i b_i(x_i) = 1$. Note that within this "mean-field" approximation, the one-node beliefs are $b_i(x_i)$ and the two node beliefs are just the products of the corresponding one-node beliefs: $b_{ij}(x_i, x_j) = b_i(x_i)b_j(x_j)$. Using this approximate joint probability function, it is easy to compute the approximate Gibbs free energy. The energy of a configuration of a pairwise MRF is

$$E(\{x\}) = -\sum_{(ij)} \ln \psi_{ij}(x_i, x_j) - \sum_i \ln \phi_i(x_i) \tag{26}$$

so the mean-field average energy is

$$U_{MF}(\{b_i\}) = -\sum_{(ij)} \sum_{x_i, x_j} b_i(x_i) b_j(x_j) \ln \psi_{ij}(x_i, x_j) - \sum_i \sum_{x_i} b_i(x_i) \ln \phi_i(x_i) \tag{27}$$

while the mean-field entropy is

$$S_{MF}(\{b_i\}) = -\sum_i \sum_{x_i} b_i(x_i) \ln b_i(x_i) \tag{28}$$

and mean-field Gibbs free energy is $G_{MF} = U_{MF} - S_{MF}$. Note that while the full Gibbs free energy is a function of the full joint probability distribution, the mean-field free energy is only a function of the one-node beliefs. Since we know the Gibbs free energy is bounded below by the Helmholz free energy, it is reasonable to search for that configuration of b_i's that minimizes G_{MF}. This is the standard variational justification for mean field theory. (See (Jordan, Ghahramani, Jaakkola, and Saul 1998) and (Jaakkola 2000) for tutorial introductions to variational methods.)

3.2 THE BETHE FREE ENERGY

The justification for the Bethe free energy is different from that behind the mean-field theory, although there are some similarities. We would like to derive a Gibbs free energy that is a function of both the one-node beliefs $b_i(x_i)$ and the two-node beliefs $b_{ij}(x_i, x_j)$. The beliefs should obey the normalization conditions $\sum_{x_i} b_i(x_i) = \sum_{x_i, x_j} b_{ij}(x_i, x_j) = 1$ and the marginalization conditions $b_i(x_i) = \sum_{x_j} b_{ij}(x_i, x_j)$. Because of the pairwise property of pairwise MRF's, the one-node and two-node beliefs are actually sufficient to determine the average energy. In other words, for any pairwise MRF and for any approximate joint probability function such that the one-node marginal probabilities are $b_i(x_i)$ and the two-node marginal probabilities are $b_{ij}(x_i, x_j)$, the average energy will have the form

$$U = -\sum_{(ij)} b_{ij}(x_i, x_j) \ln \psi_{ij}(x_i, x_j) - \sum_i b_i(x_i) \ln \phi_i(x_i) \tag{29}$$

The average energy when computed with the exact marginal probabilities $p_i(x_i)$ and $p_{ij}(x_i, x_j)$ will also have this form, so if the one-node and two-node beliefs are exact, the average energy given by equation (29) will be exact.

The entropy is not so easy to obtain, and we must normally settle for an approximation. We could compute the entropy exactly if we could explicitly express the joint distribution $b(\{x\})$ in terms of the one-node and two-node beliefs. If our graph were singly-connected, we can in fact do that. In that case, we know that the correct joint probability distribution can be written in the form

$$b(\{x\}) = \frac{\prod_{(ij)} b_{ij}(x_i, x_j)}{\prod_i b_i(x_i)^{q_i - 1}} \tag{30}$$

where q_i is the number of nodes neighboring node i (Pearl 1988). Using this form, we obtain the Bethe approximation to the entropy

$$S_{Bethe} = -\sum_{(ij)} \sum_{x_i,x_j} b_{ij}(x_i,x_j) \ln b_{ij}(x_i,x_j) + \sum_i (q_i - 1) \sum_{x_i} b_i(x_i) \ln b_i(x_i) \tag{31}$$

For a singly-connected graph then, the Bethe approximation of both the energy and the entropy will have the correct functional dependence on the beliefs, and the values of those beliefs that minimize the Bethe free energy $G_{Bethe} = U - S_{Bethe}$ will correspond to the exact marginal probabilities. For graphs with loops, the Bethe entropy and free energy will only be approximations, albeit ones that are usually quite good. In contrast to the mean-field free energy, the Bethe free energy is *not* generally an upper bound on the true free energy.

The average energy can be written in a form that is similar to that of the Bethe entropy if we introduce the local energies $E_i(x_i) = -\ln \phi_i(x_i)$ and $E_{ij}(x_i,x_j) = -\ln \psi_{ij}(x_i,x_j) - \ln \phi_i(x_i) - \ln \phi_j(x_j)$. Using the marginalization conditions on the beliefs, we obtain

$$U = \sum_{(ij)} \sum_{x_i,x_j} b_{ij}(x_i,x_j) E_{ij}(x_i,x_j) + \sum_i (q_i - 1) \sum_{x_i} b_i(x_i) E_i(x_i) \tag{32}$$

which is exactly the same form as the Bethe approximation to the entropy except that we have replaced the $\ln b$ terms with local energy E terms.

3.3 EQUIVALENCE OF BP TO THE BETHE APPROXIMATION

In the previous section, we saw that the Bethe free energy

$$\begin{aligned} G_{Bethe}(b_i(x_i), b_{ij}(x_i,x_j)) &= \sum_{(ij)} \sum_{x_i,x_j} b_{ij}(x_i,x_j) \left(E_{ij}(x_i,x_j) + \ln b_{ij}(x_i,x_j)\right) \\ &- \sum_i (q_i - 1) \sum_{x_i} b_i(x_i) \left(E_i(x_i) + \ln b_i(x_i)\right) \end{aligned} \tag{33}$$

is equal to the exact Gibbs free energy for pairwise MRF's when their graph has no loops so that the Bethe free energy is minimal for the correct marginals. We also know that BP gives correct marginals when the graph has no loops ((13) and (20).In other words, when there are no loops, the BP beliefs are the global minima of the Bethe free energy. It turns out that we can say more: *a set of beliefs gives a BP fixed point in any graph if and only if they are local stationary points of the Bethe free energy.*

To see this, we need to add Lagrange multipliers to G_{Bethe} in order to form a Lagrangian L: $\lambda_{ij}(x_j)$ is a multiplier that enforces the marginalization constraint $b_i(x_i) = \sum_j b_{ij}(x_i,x_j)$, while γ_{ij} and γ_i are multipliers that enforce the normalizations of b_{ij} and b_i. The equation $\frac{\partial L}{\partial b_{ij}(x_i,x_j)} = 0$ gives:

$$\ln b_{ij}(x_i,x_j) = -E_{ij}(x_i,x_j) + \lambda_{ij}(x_j) + \lambda_{ji}(x_i) + \gamma_{ij} - 1 \tag{34}$$

The equation $\frac{\partial L}{\partial b_i(x_i)} = 0$ gives:

$$(q_i - 1)(\ln b_i(x_i) + 1) = (1 - q_i) E_i(x_i) + \sum_{j \in N(i)} \lambda_{ji}(x_i) + \gamma_i \tag{35}$$

differentiating the Lagrangian with respect to the Lagrange multipliers gives the marginalization constraints.

Now, suppose that we have a set of messages and beliefs that are a fixed-point of BP. We define $\lambda_{ij}(x_j)$ by

$$\lambda_{ij}(x_j) = \ln \prod_{k \in N(j) \backslash i} m_{kj}(x_j) \tag{36}$$

and using the BP equations (13) and (20) it is easy to show that λ_{ij} and the beliefs satisfy the stationarity conditions (34,35). Similarly, given beliefs and Lagrange multipliers that satisfy the stationarity equations (34,35) we use equation (36) to define messages and it is easy to show that these messages and beliefs must satisfy the BP fixed-point equations.

We can use the fact that Bayesian networks, error-correcting codes, and factor graphs can be converted into pairwise MRF's to define free energies for a wide range of different kinds of models. In every case, we have an identical story: the BP algorithm for each different model corresponds to the stationary point of a Bethe free energy for that model. The reader interested in more details should consult (Yedidia, Freeman, and Weiss 2001a).

The BP algorithm for graphical models with loops is not guaranteed to converge, but because the BP fixed points correspond to Bethe free energy minima, one can simply choose to minimize the Bethe free energy directly. Such free energy minimizations are slower than the BP algorithm, but they are at least guaranteed to converge. (Welling and Teh 2001; Yuille 2001) On the other hand, empirical exploration of this idea indicates that when BP fails to converge, it is a clue that the results from minimizing the Bethe approximation will also be quite inaccurate (Welling and Teh 2001).

We refer the interested reader to the "tree-based reparameterization" algorithm developed in (Wainright, Jaakkola, and Willsky 2001), and the "expectation propapagation" algorithm developed by (Minka 2001). These algorithms provide new formulations and extensions of the BP algorithm that can also be connected to a free energy minimization framework.

4 KIKUCHI APPROXIMATIONS TO THE FREE ENERGY

Physicists, beginning with Kikuchi (Kikuchi 1951; Kikuchi 1994), have developed a method, sometimes known as the "cluster variational method," of deriving approximations, that improve on and generalize the Bethe approximation to the Gibbs free energy. Given the close relationship between the Bethe approximation and the BP algorithm, it is natural to ask whether there exist generalized BP algorithms whose fixed points correspond to the stationary points of these improved approximations. As we shall see in the next section, the answer is yes. In fact, just as ordinary BP can be defined without reference to the Bethe approximation, we shall be able to define generalized BP algorithms without referring directly to the Kikuchi approximations to which they correspond. Thus, the reader primarily interested in implementing improved BP algorithms could begin by reading the next section immediately, and only return to this section later to better understand the approximation made by the generalized BP algorithms.

In a general "Kikuchi" approximation, the free energy is approximated as a sum of the local free energies of a set of regions of nodes. The "cluster variational method" provides one way to choose that set of regions. One begins with a basic set of clusters of nodes which include every interaction and node in the pairwise MRF, and then subtracts the free energies of over-counted intersection regions, and then adds back the free energies of the over-counted intersections of intersections, and so on.

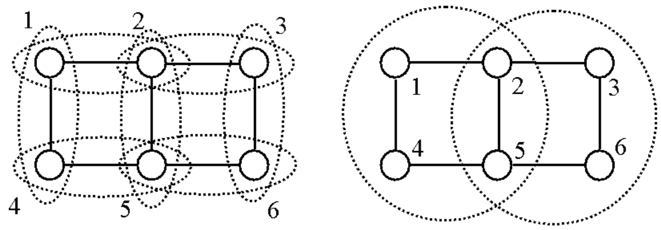

Figure 15 The basic clusters in the Bethe approximation (left) and a Kikuchi approximation (right) for a small six-node pairwise MRF.

Figure 15 shows an example of the Bethe approximation and a Kikuchi approximation for a small pairwise MRF. In this figure we have omitted the observed nodes that are understood to be connected to each hidden node. The Bethe approximation can be considered as a particular Kikuchi approximation, where we choose the basic clusters of the cluster variational method to be the set of all pairs of hidden nodes. We define the local free energy involving a single node i by

$$G_i(b_i(x_i)) = \sum_{x_i} b_i(x_i)(\ln b_i(x_i) + E_i(x_i)) \tag{37}$$

and the local free energy involving two nodes by

$$G_{ij}(b_{ij}(x_i, x_j)) = \sum_{x_i, x_j} b_{ij}(x_i, x_j)(\ln b_{ij}(x_i, x_j) + E_{ij}(x_i, x_j)) \tag{38}$$

In the Bethe approximation for this example pairwise MRF, the basic clusters are all the pairs of connected nodes, and the intersection regions are all the single nodes. Notice that when we start by summing over all local free energies of pairs of nodes, we overcount the local free energy of nodes 1, 3, 4, and 6 once, and the local free energies of nodes 2 and 5 twice. Taking node 2 as an example, we count it in the free energies of the pairs [12], [23], and [25], and we should have only counted it once, so we need to subtract $G_2(b_2(x_2))$ twice. Subtracting each of the single-node local free energies appropriately, we find that for this example,

$$\begin{aligned} G_{Bethe} &= G_{12} + G_{23} + G_{45} + G_{56} + G_{14} + G_{25} + G_{36} \\ &- G_1 - G_3 - G_4 - G_6 - 2G_2 - 2G_5 \end{aligned} \tag{39}$$

where we have suppressed the explicit dependence of each local free energy on its local beliefs. Notice that this equation is just a particular case of equation (33).

Better Kikuchi approximations are derived by just extending this kind of logic. For example, if we consider the cluster of four nodes [1245] in figure 15, we can define its local free energy to be

$$\begin{aligned} & G_{1245}(b_{1245}(x_1, x_2, x_4, x_5)) \\ = & \sum_{x_1, x_2, x_4, x_5} b_{1245}(x_1, x_2, x_4, x_5)(\ln b_{1245}(x_1, x_2, x_4, x_5) + E_{1245}(x_1, x_2, x_4, x_5) \end{aligned} \tag{40}$$

where the local energy $E_{1245}(x_1, x_2, x_4, x_5)$ is

$$\begin{aligned} E_{1245}(x_1, x_2, x_4, x_5) = & - \ln \psi_{12}(x_1, x_4) - \ln \psi_{14}(x_1, x_4) - \ln \psi_{25}(x_2, x_5) - \ln \psi_{45}(x_4, x_5) \\ & - \ln \phi_1(x_1) - \ln \phi_2(x_2) - \ln \phi_4(x_4) - \ln \phi_5(x_5). \end{aligned} \tag{41}$$

This definition of the local energy is a generalization of our previous notion: we just include all the compatibility matrices and evidence terms that influence only nodes in our cluster.

In our example of a Kikuchi approximation in figure 15, we take our basic clusters to be the four-node clusters [1245] and [2356], and we subtract off the intersection region [25], which will be over-counted. The Kikuchi free energy in this case will be

$$G_{Kikuchi} = G_{1245} + G_{2356} - G_{25} \tag{42}$$

where we have again suppressed the explicit dependence of the local free energies on the local beliefs.

Figure 16 shows a more generic situation. (Again we omitted the observed nodes for clarity.) When we use the indicated quartets of nodes as our basic clusters in the cluster variational method, we find

$$\begin{aligned} G_{Kikuchi} &= G_{1245} + G_{2356} + G_{4578} + G_{5689} \\ &- G_{25} - G_{45} - G_{56} - G_{58} + G_5 \end{aligned} \tag{43}$$

Notice that the pairs of nodes [25],[45], [56], and [58] each appear in two basic clusters, so that we must subtract off their local free energies once. But once we do that we see that node 5 appears in four basic clusters and then was subtracted in each of the intersection regions, so that its local free energy must be added again.

For a general pairwise MRF, we define a Kikuchi approximation by a set of regions R and a set of *counting numbers* c_r for each region $r \in R$. The counting numbers must satisfy

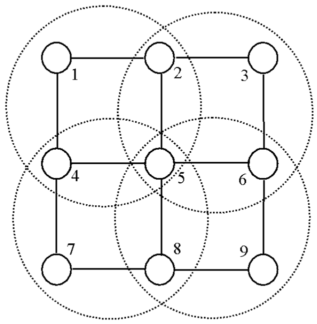

Figure 16 Four-node basic clusters in a a Kikuchi approximation for a small nine-node pairwise MRF.

the condition that each interaction and node is ultimately counted once when one includes the contributions of every region. Formally, this means that we require that

$$\sum_{r \in R} c_r \, [i \in r] = \sum_{r \in R} c_r \, [i, j \in r] = 1 \tag{44}$$

for all nodes i and pairs of nodes i, j in the pairwise MRF. Here $[i \in r]$ is an indicator function equal to one if node i is in region r and equal to zero otherwise. In the cluster variational method, this is condition can be guaranteed to be satisified by defining

$$c_r = 1 - \sum_{s \in super(r)} c_s \tag{45}$$

where $super(r)$ is the set of all super-regions of r.

Let $\{x\}_r$ be a state of all the nodes in region r, and $b_r(\{x\}_r)$ be the belief in the state $\{x\}_r$. Define the energy of a region by

$$E_r(\{x\}_r) \equiv -\ln \prod_{(ij)} \psi_{ij}(x_i, x_j) - \ln \prod_i \phi_i(x_i) \tag{46}$$

where the products are over all nodes or pairs of nodes that are contained entirely in region r. Then the Kikuchi free energy is

$$G_K = \sum_{r \in R} c_r \left(\sum_{x_r} b_r(\{x\}_r) E_r(\{x\}_r) + \sum_{x_r} b_r(\{x\}_r) \ln b_r(\{x\}_r) \right) \tag{47}$$

Of course, the beliefs $b_r(\{x\}_r)$ in region r must sum to one and be consistent with the beliefs that intersect with r.

In general, increasing the size of the basic clusters improves the approximation one obtains by minimizing the Kikuchi free energy. As we saw, the Bethe free energy for a pairwise

MRF already has an average energy term that is exact, and this continues to be true of all improved Kikuchi approximations. The improvement arises in the treatment of the entropy, which becomes increasingly accurate as the basic clusters become larger. In the limit where a basic cluster covers all the nodes in the system, the Kikuchi approximation for the entropy becomes exact.

5 GENERALIZING BELIEF PROPAGATION

In ordinary BP, all messages are always from a single node to another single node. It is natural to expect that messages from groups of nodes to other groups of nodes could be more informative, and thus lead to better inference. That is the basic intuitive idea behind generalized belief propagation (GBP). The mathematical justification of GBP algorithms is that, if we define messages and message-update rules appropriately, we can show that the fixed points of a GBP algorithm are equivalent to the stationary points of a corresponding Kikuchi approximation to the free energy. In this paper, we will not give the procedure to construct a GBP algorithm in the general case, or the proof of the equivalence to a Kikuchi approximation. We refer the interested reader to (Yedidia, Freeman, and Weiss 2001a). Instead, to illustrate the ideas involved, we will work step-by-step through the "canonical" method to construct a GBP algorithm for an example pairwise MRF.

Specifically, let us return to figure 16 and try to construct a GBP algorithm that corresponds to the Kikuchi approximation described in the last section. The basic clusters had four nodes each: [1245], [2356], [4578], and [5689]. The first step in constructing a GBP algorithm is to find all the intersection regions of the basic clusters, and all their intersection regions, and so on. We find the intersection regions [25], [45], [56], and [58], and the single region that is an intersection of intersections: [5].

The next step in constructing a GBP algorithm is to organize all the regions into a *region graph*: a hierarchy of regions and their "direct" sub-regions. A direct sub-region s of a region r is a sub-region of R that is not also a sub-region of another sub-region of r. In figure 17, we give the hierarchy of regions for our example. Note that region [5] is not a direct sub-region of region [1245], because it is also a sub-region of region [25].

The next step is to construct messages connecting all regions r to all their direct sub-regions s. In other words, we associate a message with each line in figure 17. Consider for example the message connecting region [1245] to region [25]. We can consider this to be a message from nodes 1 and 4 to nodes 2 and 5. We will denote this message by $m_{14\to25}(x_2, x_5)$. In general, we can consider a message connecting a region r and a sub-region s to be a message from those nodes in r that are not in s to the nodes in s.

The next step is very important. We construct belief equations for every region r, according to the rule that the belief $b_r(\{x\}_r)$ is proportional to the product of every compatibility matrix and evidence term contained completely in the region, and every message that goes into nodes in the region from nodes outside the region. For example, for the region consisting just of node 5, the belief equation is

$$b_5 = k\,[\phi_5]\,[m_{2\to5}m_{4\to5}m_{6\to5}m_{8\to5}]\,. \tag{48}$$

where, k is a normalization constant, and we have suppressed all the obvious functional dependences on the states $\{x\}$. This equation is illustrated in figure 18 on both the region

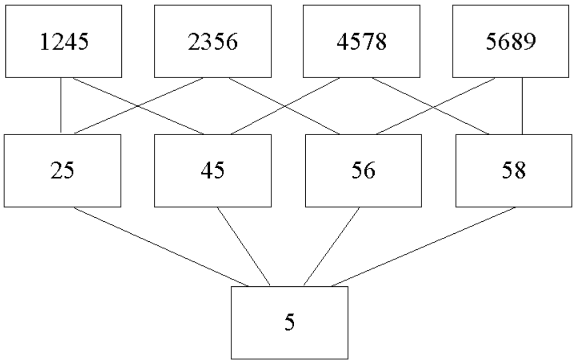

Figure 17 Region graph showing the hierarchy of regions in our nine-node example. Regions are connected to their direct sub-regions beneath them.

graph and the original pairwise MRF. Taking the region [45] as an example of a two-node region, its belief equation is:

$$b_{45} = k\left[\phi_4\phi_5\psi_{45}\right]\left[m_{12\to 45}m_{78\to 45}m_{2\to 5}m_{6\to 5}m_{8\to 5}\right], \tag{49}$$

which we illustrate in figure 19. Taking the region [1245] as an example of a four-node region, its belief equation is:

$$b_{1245} = k\left[\phi_1\phi_2\phi_4\phi_5\psi_{12}\psi_{14}\psi_{25}\psi_{45}\right]\left[m_{36\to 25}m_{78\to 45}m_{6\to 5}m_{8\to 5}\right], \tag{50}$$

which is illustrated in figure 20. The region belief equations are natural generalizations of the Bethe belief equations, but one might still wonder about their justification. Although we are omitting the proof here, the point is that this natural way to construct belief equations is precisely what you need for the GBP fixed points to be stationary points of the corresponding Kikuchi free energy.

The next, and final, step in constructing a GBP algorithm is to enforce the marginalization condition relating each pair of regions that are connected in the hierarchy shown in figure 17. For example the marginalization condition connecting the region [5] with the region [45] is $b_5(x_5) = \sum_{x_4} b_{45}(x_4, x_5)$. If we combine that with our previous belief equations (48) and (49), we find, by cancelling common terms, the message update rule

$$m_{4\to 5}(x_5) \leftarrow k\sum_{4_2}\phi_4(x_4)\psi_{45}(x_4,x_5)m_{12\to 45}(x_4,x_5)m_{78\to 25}(x_2,x_5) \tag{51}$$

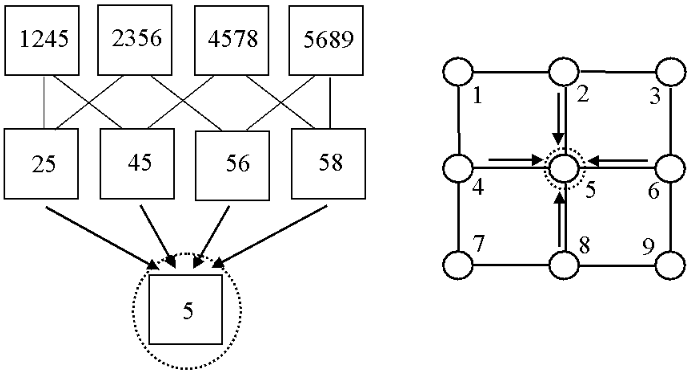

Figure 18 The belief equation $b_5 = k[\phi_5][m_{2\to5}m_{4\to5}m_{6\to5}m_{8\to5}]$ for the region [5], illustrated both on the region graph (left) and on the original pairwise MRF (right).

The collection of all the belief equations and message update rules defines our GBP algorithm. A GBP algorithm runs in the same way as the BP algorithm. One normally initializes all the messages to their unbiased states, and then iterates the message update rules until they (hopefully) converge. Occasionally, it is helpful from the point of view of convergence to only move part-way at each iteration towards the new values of the messages. When convergence of the messages is achieved, the desired beliefs can be read off from the belief equations.

How well do GBP algorithms work? For a longer answer with details, the reader is referred to (Yedidia, Freeman, and Weiss 2001b), where we describe experiments where GBP algorithms can significantly out-perform ordinary BP on two-dimensional pairwise MRF's, and for decoding error-correcting codes. The short answer is that GBP algorithms nearly always improve, at least slightly, over the performance of ordinary BP, and they can significantly out-perform ordinary BP if the graphical model under consideration has short loops.

As for the complexity of GBP, the bad news is that it grows exponentially with the size of the basic clusters that are chosen. The good news is that if the basic clusters encompass the shortest loops in the graphical model, one usually eliminates nearly all the error associated with the BP algorithm. For many graphical models, (e.g. square lattice pairwise MRF's), using such basic clusters actually involves only minimally more computation than ordinary BP.

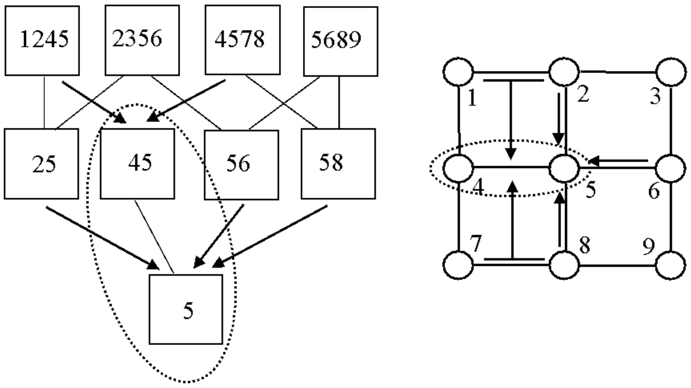

Figure 19 The belief equation $b_{45} = k\,[\phi_4\phi_5\psi_{45}]\,[m_{12\to 45}m_{78\to 45}m_{2\to 5}m_{6\to 5}m_{8\to 5}]$, for the region [45], illustrated both on the region graph (left) and on the original pairwise MRF (right). Note that we include all messages that impinge upon the region [45] or its subregion [5].

5.1 GBP, CLUSTERING AND JUNCTION TREES

A standard algorithm for exact inference in graphical models is the *junction tree* algorithm (Cowell 1998), which we briefly illustrate in figure 21a-d. The graph is first *triangulated* (i.e. we add edges so that every cycle of length > 3 has a chord). In our example, this gives the graph in figure 21b. We then find the maximal cliques of the junction tree and connect these in a tree (figure 21c). The potentials on the spanning tree are set so that the tree defines an equivalent MRF to the original problem. Once we have an equivalent tree, we can simply run BP on the junction tree to obtain marginals on the cliques (and an additional marginalization on each clique will give us marginals on individual nodes). However, since the nodes in the junction tree correspond to overlapping sets of variables in the original graph, one would like a BP algorithm that takes advantage of this fact to reduce the size of the messages. In the Shafer-Shenoy algorithm(Shafer and Shenyoy 1990) messages between cliques are functions of the state of the *separators* or interesections between cliques. This algorithm is equivalent to to running BP on a tree in which we have added extra nodes to denote the separators (figure 21d). This algorithm is known as the generalized distributive law in the information theory literature (Aji and McEliece 2000).

A related method is Pearl's method of clustering(Pearl 1988) illustrated in figure 21e. In this method we form clusters of nodes from the original graph and construct a new graph from the clusters such that the new graph represents the same probability distribution. BP is then run on the cluster graph.

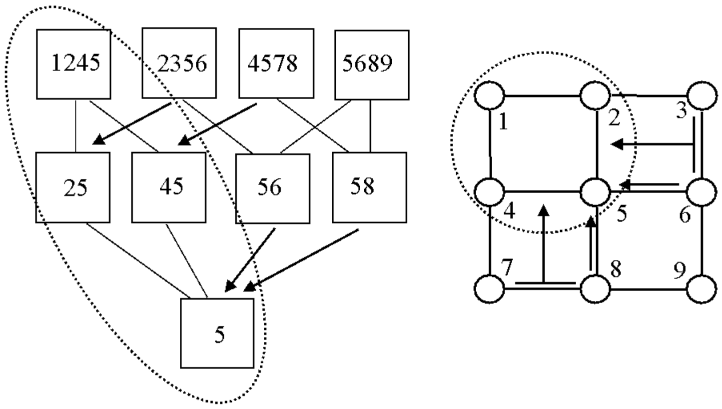

Figure 20
The belief equation $b_{1245} = k\left[\phi_1\phi_2\phi_4\phi_5\psi_{12}\psi_{14}\psi_{25}\psi_{45}\right]\left[m_{36\to25}m_{78\to45}m_{6\to5}m_{8\to5}\right]$, for the region [1245], illustrated both on the region graph (left) and on the original pairwise MRF (right).

Both the junction tree and the clustering algorithms pass messages that are functions of clusters of nodes. How are these algorithms related to GBP?

The short answer is that both the junction tree and the clustering algorithms are special cases of Kikuchi approximations. For example, the junction tree algorithm in figure 21d approximates the Gibbs free energy using:

$$G_{Kikuchi} = G_{123} + G_{235} + G_{245} - G_{23} - G_{25} \tag{52}$$

while the clustering method in figure 21e approximates the Gibbs free energy using:

$$G_{Kikuchi} = G_{123} + G_{2345} - G_{23} \tag{53}$$

Both equations (52) and equation (53) are cases where the Kikuchi approximation is no longer an approximation: it is an exact expression. It can be shown that whenever the region graph (e.g. figure 17) contains only two levels (i.e. regions and their intersections) and the region graph has no cycles, the Kikuchi approximation is exact (Yedidia, Freeman, and Weiss 2001a; Aji and McEliece 2001).

While the junction tree and clustering methods can yield exact Kikuchi expansions, for many practical problems they require region sizes that are enormous. The goal of the general Kikuchi framework is to find expansions that are of reasonable accuracy but of far less complexity than the junction tree. In general, by choosing an appropriate Kikuchi

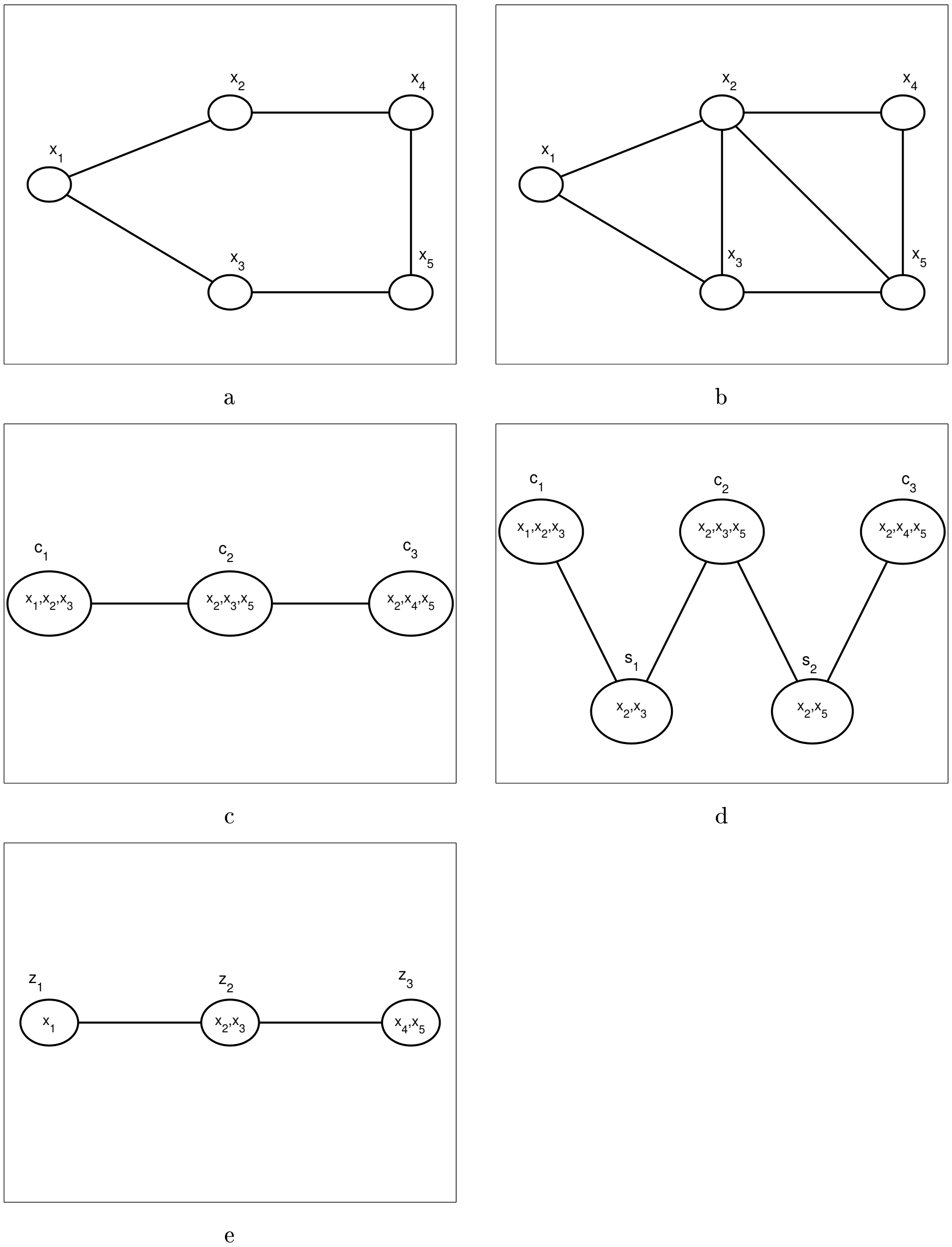

Figure 21 **a.** an undirected graph with loops. **b.** the same graph after triangulation: edges have been added so that every cycle of length > 3 has a chord **c.** the junction tree is a spanning tree of the maximal cliques of the triangulated graph. **d.** inference in the junction tree can be performed by running BP on a graph that includes the cliques and the separators. **e.** an alternative way to convert the original graph into a tree is the method of clustering.

approximation and corresponding generalized BP algorithm, one can adjust the trade-off between accuracy and complexity. As a practical matter, how to choose the "optimal" Kikuchi approximation is still more an art than a science. We only offer the advice that one should try to insure that the shortest loops in the graph are entirely included in Kikuchi regions, so that they are handled exactly.

6 SUMMARY

The success of BP and GBP algorithms is exciting, because it means that many different kinds of problems that seemed so difficult to handle, involving graphs with many nodes and loops, can actually be handled using efficient and systematically correctable algorithms. These algorithms are much faster than Monte Carlo approaches, and the approximations to the free energy that they are effectively implementing are more sophisticated and accurate than "mean-field" approximations. These algorithms give a principled framework for propagating, in parallel, information and uncertainty between nodes in a network. It would be exciting if they had relevance or modeling power for neural computational systems.

References

Aji, S.M. and McEliece, R.J. (2000) The Generalized Distributive Law. *IEEE Transactions on Information Theory* 46:325–343.

Aji, S.M. and McEliece, R.J (2001) The Generalized Distributive Law and Free Energy Minimization. To be published in the *Proceedings of the 39th Annual Allerton Conference on Communication, Control, and Computing.*

Baxter, R. J. (1982). *Exactly Solved Models in Statistical Mechanics.* Academic Press.

Berrou, C., A. Glavieux, and P. Thitimajshima (1993). Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *Proceedings 1993 IEEE International Conference on Communications*, Geneva, Switzerland, pp. 1064–1070.

Boykov, Y., O. Veksler, and R. Zabih (2001). Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions in Pattern Analysis and Machine Intelligence 23*(11),1222–1239.

Cover, T. M. and J. A. Thomas (1991). *Elements of Information Theory.* John Wiley and Sons.

Cowell, R. (1998). Introduction to Inference for Bayesian Networks. In M. Jordan (Ed.), *Learning in Graphical Models.* MIT Press.

Freeman, W. T., E. C. Pasztor, and O. T. Carmichael (2000). Learning low-level vision. *Intl. J. Computer Vision 40*(1), 25–47.

Frey, B. J. (1998). *Graphical Models for Machine Learning and Digital Communication.* MIT Press.

Frey, B. J. and D. J. C. Mackay (1998). A revolution: Belief propagation in graphs with cycles. In M. Jordan, M. S. Kearns, and S. A. Solla (Eds.), *Adv. in Neural Information Processing Systems*, Volume 10. MIT Press.

Gallager, R. G. (1963). *Low-density parity check codes.* MIT Press.

Gallager, R. G. (1968). *Information Theory and Reliable Communication.* John Wiley and Sons.

Geman, S. and D. Geman (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions in Pattern Analysis and Machine Intelligence 6*(6),721–741.

Jaakkola, T. (2000). Tutorial on variational approximation methods. available online at http://www.ai.mit.edu/people/tommi/papers.html.

Jensen, F. (1996). *An Introduction to Bayesian Networks*. Springer.

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

Kikuchi, R. (1951). *Phys. Rev. 81*, 988.

Kikuchi, R. (1994). Special issue in honor of R. Kikuchi. Progr. Theor. Phys. Suppl., vol. 115, 1994.

Kschischang, F. R., B. J. Frey, and H.-A. Loeliger (2001). Factor graphs and the sum-product algorithm. *IEEE Trans. Info. Theory 47*, 498–519.

Lauritzen, S. L. and D. J. Spiegelhalter (1988). Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society, Series B 50*, 157–224.

Mackay, D. J. C. (1999). Good error-correcting codes based on very sparse matrices. *IEEE Trans. Info. Theory 45*, 399–431.

McEliece, R. J., D. J. C. MacKay, and J. F. Cheng (1998). Turbo decoding as an instance of Pearl's 'belief propagation' algorithm. *IEEE J. on Sel. Areas in Comm. 16*(2), 140–152.

Mezard, M., G. Parisi, and M. A. Virasoro (1987). *Spin glass theory and beyond*. World Scientific.

Minka, T.P. (2001). Expectation Propagation for Approximate Bayesian Inference. UAI 2001.

Murphy, K., Y. Weiss, and M. Jordan (1999). Loopy belief propagation for approximate inference: an empirical study. In *Proc. Uncertainty in AI*.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Shafer G.R. and Shenoy P.P.. (1990). Probability Propagation. *Annals of Mathematics and Aritifical Intelligence* 2:327–352.

Tanner, R. M. (1981). A recursive approach to low complexity codes. *IEEE Trans. Info. Theory IT-27*, 533–547.

Wainright, M.J., T. Jaakkola, and A.S. Willsky (2001). Tree-based reparameterization framework for approximate estimation in graphs with cycles. LIDS Technical Report P-2510. Available online at http://ssg.mit.edu/group/mjwain/mjwain.shtml.

Welling, M. and Y. W. Teh (2001). Belief optimization: A stable alternative to belief propagation.

Yedidia, J. S. (2001). An idiosyncratic journey beyond mean field theory. In D. Saad and M. Opper (Eds.), *Advance Mean Field Methods–Theory and Practice*. MIT Press.

Yedidia, J. S., W. T. Freeman, and Y. Weiss (2001a). Bethe free energies, Kikuchi approximations, and belief propagation algorithms. Available online at http://www.merl.com/reports/TR2001-16/index.html.

Yedidia, J. S., W. T. Freeman, and Y. Weiss (2001b). Characterizing belief propagation and its generalizations. Available online at http://www.merl.com/reports/TR2001-15/index.html.

Yuille, A. L. (2001). A double-loop algorithm to minimize the Bethe and Kikuchi free energies. Unpublished.