MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY
and
CENTER FOR BIOLOGICAL AND COMPUTATIONAL LEARNING
DEPARTMENT OF BRAIN AND COGNITIVE SCIENCES

A.I. Memo No. 1616 November, 1997
C.B.C.L. Paper No. 155

Belief Propagation and Revision in Networks with Loops

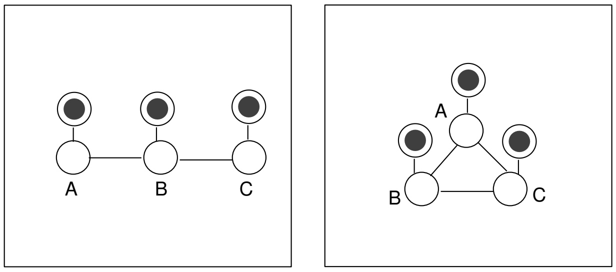
Yair Weiss
Dept. of Brain and Cognitive Sciences
MIT E10-120, Cambridge, MA 02139, USA
yweiss@psyche.mit.edu

This publication can be retrieved by anonymous ftp to publications.ai.mit.edu.

Abstract

Local belief propagation rules of the sort proposed by Pearl (1988) are guaranteed to converge to the optimal beliefs for singly connected networks. Recently, a number of researchers have empirically demonstrated good performance of these same algorithms on networks with loops, but a theoretical understanding of this performance has yet to be achieved. Here we lay a foundation for an understanding of belief propagation in networks with loops. For networks with a single loop, we derive an analytical relationship between the steady state beliefs in the loopy network and the true posterior probability. Using this relationship we show a category of networks for which the MAP estimate obtained by belief update and by belief revision can be proven to be optimal (although the beliefs will be incorrect). We show how nodes can use local information in the messages they receive in order to correct the steady state beliefs. Furthermore we prove that for all networks with a single loop, the MAP estimate obtained by belief revision at convergence is guaranteed to give the globally optimal sequence of states. The result is independent of the length of the cycle and the size of the state space. For networks with multiple loops, we introduce the concept of a "balanced network" and show simulation results comparing belief revision and update in such networks. We show that the Turbo code structure is balanced and present simulations on a toy Turbo code problem indicating the decoding obtained by belief revision at convergence is significantly more likely to be correct.

This report describes research done at the Center for Biological and Computational Learning and the Department of Brain and Cognitive Sciences of the Massachusetts Institute of Technology. Support for the Center is provided in part by a grant from the National Science Foundation under contract ASC–9217041. YW was also supported by NEI R01 EY11005 to E. H. Adelson



a b

Figure 1: **a.** An example of the types of problems typically solved using belief propagation. Observed nodes are denoted by filled circles. A link between any two nodes implies a probabilistic compatability constraint. **b.** A simple network with a loop. Although belief propagation rules can be generalized to this network, a theoretical understanding of the algorithms behavior in such a network has yet to be achieved.

1 Introduction

Problems involving probabilistic belief propagation arise in a wide variety of applications including error correcting codes, speech recognition and medical diagnosis. The basic problem we are interested in is perhaps best illustrated by a simple example. Consider figure 1. The task is to infer the states of three hidden nodes (labeled A, B, C) having observed the states of the three evidence nodes (labeled E_a, E_b, E_c). We assume a probability distribution on the variables given by:

$$P(A,B,C|E_a,E_b,E_c) = \frac{1}{Z} e^{-J(A,B,C,E_a,E_b,E_c)} \quad (1)$$

where Z is a normalization function and:

$$J = J_1(A,B)+J_2(B,C)+J_3(A,E_a)+J_4(B,E_b)+J_5(C,E_c) \quad (2)$$

Intuitively, there is a cost associated with every link which specifies the "compatibility" of the two nodes connected by that link[1]. The total cost is decomposed into five additive costs. Alternatively, the probability function P can be thought of as factoring into five multiplicative factors:

$$P = \Psi(A,B)\Psi(B,C)\Psi(A,E_a)\Psi(B,E_b)\Psi(C,E_c) \quad (3)$$

The task of "inferring the states of the hidden nodes" can have at least two meanings. In one task, the goal is to calculate the probability that a hidden node is in a particular state given all the evidence e.g. $P(A = a|E)$. In the AI literature, this is referred to as calculating the *belief function* for node A. The other task is to find the most likely sequence of states for the hidden nodes i.e. the triplet (a^*, b^*, c^*) that maximizes $P(A = a, B = b, C = c|E)$.

Both tasks can be done by exhaustive enumeration. Obviously, the most likely sequence can be found by trying all possibilities and the belief function can be calculated by marginalization:

$$P(A = a|E) = \sum_{b,c} P(A = a, B = b, C = c|E) \quad (4)$$

A naive exhaustive enumeration is, of course, exponential in the number of hidden nodes. Furthermore, it does not exploit the decomposition of the probability function. The decomposition suggests that the problem should lend itself to parallel distributed processing at the nodes of the network.

In 1986 J. Pearl [10] described local message passing schemes for inferring the states of the hidden nodes in these types of networks. The algorithm consists of simple local updates that can be executed in parallel and are guaranteed to converge to the correct answers. Furthermore, on a serial computer the complexity is linear in the number of hidden units. Pearl used the terms *belief update* to describe the scheme for computing the belief function and the term *belief revision* for the analogous scheme for finding the most likely sequence. Pearl's algorithm is equivalent to schemes proposed independently in a wide range of fields including information theory, signal processing, operations research and optimal estimation.

Pearl proved that his algorithm would work for singly connected networks, i.e. ones in which there is a single path going between any two nodes. In many applications, however, the network is multiply connected. Consider for example, the network shown in figure 1b, where we have a compatibility link between the first and last hidden nodes.

$$\begin{aligned} P(A,B,C|E) &= \Psi(A,B)\Psi(B,C)\Psi(C,A) && (5) \\ & \quad \Psi(A,E_a)\Psi(B,E_b)\Psi(C,E_c) && (6) \end{aligned}$$

Pearl showed that his algorithm is not guaranteed to work for such a network and suggested various other ways to cope with loops [10]. Despite this fact, several groups [5, 8, 14] have recently reported excellent experimental results in inference in networks with loops by using Pearl's algorithm. Perhaps the most dramatic instance of this performance is in an error correcting code scheme known as "Turbo Codes" [2]. These codes have been described as "the most exciting and potentially important development in coding theory in many years" [9] and have recently been shown [4, 7] to utilize an algorithm equivalent to belief propagation in a network with loops. Although there is widespread agreement in the coding community that these codes "represent a genuine, and perhaps historic, breakthrough" [9] a theoretical understanding of their performance has yet to be achieved.

Here we lay a foundation for an understanding of belief propagation in networks with loops. For networks with a single loop we derive a relationship between the steady state beliefs in the loopy network and the true posterior probability. Using this relationship we show a category of networks for which the MAP estimate obtained by belief update and by belief revision can be proven to be optimal (although the beliefs will be incorrect). Furthermore we prove that for all networks with

[1]We assume only pairwise compatibilities between nodes. This is slightly more restrictive than the general MRF formulation

a single loop, the MAP estimate obtained by belief revision is guaranteed to give the globally optimal sequence of states. For networks with multiple loops, we introduce the concept of a "balanced network" and derive a necessary condition for the MAP estimate at convergence to be optimal. We show simulations on such networks including a toy Turbo decoding problem, and show that belief revision is significantly more likely to give a correct decoding.

The organization of this paper is as follows. Section 2 introduces a variant of Pearl's algorithms for belief propagation in singly connected networks. Section 3 gives an intuitive explanation of why these algorithms may work in a loopy network as well. Section 4 deals with *belief update* and derives the analytical relationship between the steady state beliefs and the true posteriors. Section 5 deals with *belief revision* and proves the optimality of the MAP estimate. Finally section 6 presents simulation results on structures containing multiple loops including Turbo codes.

2 Belief Propagation and Revision in Singly connected Networks

We describe here the update rules for belief propagation and revision in singly connected networks. Since the publication of Pearl's algorithm, a number of variants of it have been published (see [12] for a review). The updates we describe here are functionally equivalent to those proposed by Pearl, but are easier to generalize to loopy networks.

Informally, the message passing scheme proceeds as follows. Every node sends a probability vector to each of its neighbors. Suppose Y has two neighbors X and Z. Roughly speaking, the message that X sends to Y is the probability that Y is in a particular state, given the information that X knows but Y does not. Node Y then takes that vector, adds its local information, and transmits a message to Z indicating the probability that Z is in a particular state given the information at X and Y. Similarly, Y takes the message from Z and after adding its information, sends a message to X. This procedure is repeated for all nodes in the network in parallel. The belief function at Y is obtained by combining the steady state values of the messages from X and Z and the local evidence at Y.

More formally, assume the probability distribution is factorized into a product form:

$$P(H) = \frac{1}{Z}\Pi\Psi(H_k, H_{k+1}) \tag{7}$$

There is a Ψ associated with each connection in the network, i.e. any two neighboring nodes in the network. The ordering of the nodes and links is somewhat arbitrary: for a Markov network such a representation can always be found but is not unique (e.g. [10]). If H_k and H_{k+1} are two nodes in the network, each of which can be in a discrete number of states, we can associate a matrix M with each link:

$$M_{ji} = \Psi(H_k = i, H_{k+1} = j) \tag{8}$$

Note that the matrix going in one direction on a link is equal by definition to the transpose of the matrix going in the other direction. Given this matrix we can define the following *belief update* procedure:

The message that node X sends to node Y is calculated as follows:

- Combine all messages coming into X except for that coming from Y into a vector v. The combination is done by multiplying all the message vectors element by element.
- Multiply v by the matrix M corresponding to the link from X to Y.
- Normalize the product Mv so it sums to 1. The normalized vector is sent to Y.

The procedure is initialized with all message vectors set to $(1, 1, \cdots 1)$. Evidence nodes do not receive messages and they always transmit the same vector, i.e. $\Psi(i, e)$ for all i. The belief vector for a node X is obtained by combining all incoming messages to X (again by multiplying the message vectors element by element) and normalizing. For a singly connected unit, these belief vectors are guaranteed to converge to the posterior probability of the node X given all the evidence.

It is easy to show that these updates are functionally equivalent to Pearl's propagation rules. Note, however, that Pearl's rules were for Bayesian networks, while here we are interested in Markov networks. The distinction is discussed at length in Pearl (1988). Roughly speaking, Bayesian networks have arrows on the links, and of any two neighboring nodes, one is considered to be the cause and the other is the effect. Markov networks replace the notion of causal link, with a weaker notion of "compatibility" which is symmetric in the two neighboring nodes. For singly connected nets, The probability distributions represented by Markov networks are a subset of those that can be represented by Bayesian nets. The reason we are focusing here on Markov nets, is that when the network has loops, assuming causal relationships between neighboring nodes may lead to logical contradictions (e.g. consider the simple loop in figure 1b). Another difference between the algorithm presented here and Pearl's original algorithm is in the normalization. In Pearl's algorithm messages going in one direction are normalized while those in the other direction are not. As pointed out by Pearl, the normalization step does not influence the final beliefs but ensures the stability of the message passing scheme. Special cases of these update rules are also functionally equivalent to the Baum Welch reestimation procedure in HMMs [11] and optimal smoothing [6].

Belief update will give the probability of every node being in a particular state given all the evidence. If, however, we set each node equal to the state that maximizes its belief function, the global sequence obtained will not necessarily maximize the posterior probability. In order to obtain the most likely sequence, we need to calculate a revised belief function. This can be done with the *belief revision* procedure.

The belief revision is identical to that described earlier for belief update. The only difference is the matrix multiplication Mv in step 3 above. We define the operator $M_\infty v$ as follows:

$$(M_\infty v)_i = \max_j M_{ij} v_j \tag{9}$$

Note that $M_\infty v$ is similar to regular matrix multiplication, with the sum replaced with the maximum operator. It can be shown that when the belief revision procedure is run on a singly connected network, the belief function will converge to:

$$B_X(i) = \alpha \max_{\vec{S} s.t. X=i} P(\vec{S}|E) \tag{10}$$

where α is a normalizing constant independent of i and $\vec{S}$ denotes the a sequence of values for all hidden nodes. By choosing at each node the value i^* that maximizes the revised belief, we are guaranteed to obtain the most likely sequence.

The belief revision rules described here are equivalent to those described by Pearl (1988) except for the normalization. In Hidden Markov Models, belief revision is equivalent to the Viterbi update rules [11], and is a special case of concurrent dynamic programming [3].

As can be seen from the previous discussion, update procedures of the type described by Pearl have been analyzed in many areas of optimization and applied mathematics. However, to the best of our knowledge, in all these contexts the network is assumed to be singly connected. Note, however, that these procedures are perfectly well defined for any Markov network – all they require is a decomposition of the posterior probability into pairwise compatibilities. Analyzing what happens when the update procedures are applied to networks with loops is the goal of this paper.

3 Intuition - why does loopy propagation work?

Before launching into the details of the analysis, it is worthwhile to consider the examples in figures 1 and obtain some intuition.

As Pearl has pointed out, in order for a message passing scheme to be successful, it needs to avoid "double counting" – a situation in which the same evidence is passed around the network multiple times and mistaken for new evidence. In a singly connected network (e.g. figure 1a) this is accomplished by Pearl's algorithm. Thus in figure 1a, node B will receive from A a message that involves the local evidence at A and send that information to C.While the message it sends to A will involve the information at C but *not* the information at A. Thus A never receives its own information back again, and double counting is avoided.

In a loopy graph (e.g. 1b) double counting can not be avoided. Thus B will send A's information to C, but in the next iteration, C will send that information back to A. Thus it seems that belief propagation in such a net will invariably give the wrong answer. How then, can we explain the good performance reported experimentally?

Intuitively, the explanation is that although the evidence is "double counted", all evidence is double counted in equal amounts. Therefore, as we prove below, for networks such as these, the MAP estimates will be correct although the beliefs will be numerically wrong. Furthermore, we will show that the convergence rate of each node's messages tells the node something about the amount of double counting. Thus each node can correct for the double counting and get the exact beliefs from the propagated messages.

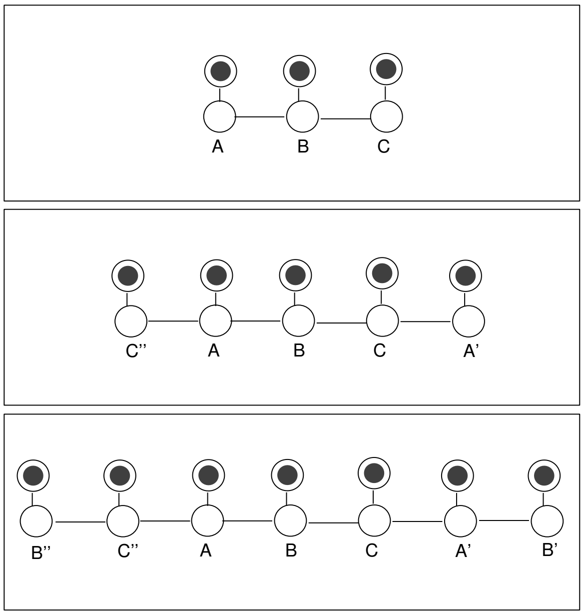


Figure 2: Unwrapped networks corresponding to the simple loop figure shown in figure 1. The messages received by node B after t iterations in the loopy network are equivalent to those that would be received by B in the unwrapped network. The unwrapped networks for the first three iterations are shown.

An intuitive way of understanding belief propagation in networks with loops is to examine what we call the "unwrapped network" corresponding to a loopy network. The unwrapped network is a singly connected network constructed such that performing belief update in the unwrapped network is equivalent to performing belief update in the loopy network. The exact construction method of the unwrapped network is given in the appendix, but the basic idea is to replicate the evidence nodes as shown in the examples in figure 2. As we show in the appendix (see also [14, 8]), for any number of iterations of belief propagation in the loopy network, there exists an unwrapped network such that the messages received by a node in the unwrapped network are equivalent to those that would be received by a corresponding node in the loopy network. This concept is illustrated in figure 2. The messages received by node B after $1, 2$ and 3 iteration of propagation in the loopy network, are identical to those received by node B after convergence in the unwrapped networks shown in figure 2.

It seems that all we have done is convert a finite, loopy problem into an infinite network without loops. What have we gained? The importance of the unwrapped network, is that since it is a polytree, belief propagation on it is *guaranteed to give the correct beliefs*. Thus assuming the loopy propagation converges after N iterations, the steady state beliefs in the loopy network are guaranteed to be the correct posterior probabilities for the unwrapped problem of size N. The usefulness of this estimate now depends on the similarity between the probability distribution induced by the unwrapped problem and the original loopy problem.

In subsequent sections we formally compare the probability distribution induced by the loopy network to that induced by the original problem. Roughly speaking, if we denote the original probability induced on the hidden nodes in figure 1b by:

$$P(a, b, c) = \alpha e^{-J(a,b,c)} \tag{11}$$

The most likely sequence for the unwrapped problem maximizes:

$$\tilde{P}(a, b, c) = \alpha e^{-kJ(a,b,c)} \tag{12}$$

for some positive constant k. That is, if we are considering the probability of a sequence of states the unwrapped problem induces a probability which is a monotonic transformation of the true probability. In statistical physics terms, the unwrapped network has the same energy function but at different temperature. This is the intuitive reason that *belief revision* which finds the most likely sequence in the unwrapped network, is guaranteed to find the most likely sequence in loopy networks. However, *belief update* which finds marginal distributions of particular nodes may give an incorrect decoding – the marginals of $\tilde{P}$ are not necessarily a monotonic transformation of the marginals of P.

Since every iteration of loopy propagation gives the correct beliefs for a different problem, it is not immediately clear why this scheme should ever converge. Note, however, that the unwrapped network of iteration $n+1$ is simply the unwrapped network of size n with an additional finite number of nodes added at the boundary. Thus loopy belief propagation will converge when the addition of these additional nodes at the boundary will not alter the posterior probability of the node in the center. In other words, convergence is equivalent to the independence of center nodes and boundary nodes in the unwrapped network.

There is a large body of research on the conditions under which the probability of center nodes in a Markov Random Fields are independent of the boundary nodes. Unfortunately, the situation is rather complex - small changes in the parameters of the network may cause a phase transition from convergent to nonconvergent. For the unwrapped networks of networks with a single cycle, the situation is simpler. A trivial example of convergence is if one of the hidden nodes is observed. Then its replicas in the unwrapped network form a Markov blanket around the center nodes, and they are trivially independent of the boundary nodes. A less trivial example is when none of the nodes are observed, but a large number of their replicas essentially form a Markov blanket. In other words, as the number of iterations increases, and the boundary nodes are separated by a large number of hidden nodes, the influence of the boundary nodes goes to zero. In the subsequent sections, we formalize this intuition.

4 Belief update in networks with a single loop

4.1 The case of a loop network

We start with the simplest loopy network - a simple loop (as in figure 1b). The main result is that for binary hidden state variables, the probabilities will be wrong but the MAP estimate is guaranteed to be correct. In the general case, we derive an expression that relates the correct belief function and that calculated by loopy belief update.

We label the hidden nodes with numbers $H_1 \cdots H_N$ and the evidence nodes E_i. At each node we have a diagonal matrix D_i whose diagonal elements are $\Psi(H_i, E_i)$. There are also N transition matrices M_n such that $M_n(i,j) = \Psi(H_n = i, H_{n+1} = j)$ (where the term $n+1$ is evaluated cyclically such that $N+1 = 1$). We focus on node H_1 in the network and define p the correct posterior probability vector for a that node and $\tilde{p}$ the one estimated at convergence by the loopy belief propagation algorithm.

The essence of the proof is as follows. The two messages node H_1 receives after the system reaches steady state are shown to be principal eigenvectors of two matrices determined by the potentials Ψ. The true probabilities are then shown to be the diagonal elements of these same matrices. Basic linear algebra then gives the connection between the two estimates.

Claim:

Define the matrix $C = M_1^t D_2 \cdots M_{n-1}^t D_n M_n^t D_1$ then the message that node H_2 sends to node H_1 at convergence is in the direction of the principal eigenvector of C.

Proof: Denote by $v(t)$ the message that node H_2 sends to node H_1 at time t then from the update rules it is easy to show:

$$v(t) = \alpha C v(t-N) \tag{13}$$

Thus $v(kN) = \alpha C^k v(0)$ which means that the stationary vector is in the direction of the principal eigenvector of C.

Similarly if we define:

$$C_2 = M_n D_n M_{n-1} D_{n-1} \cdots M_1 D_1 \tag{14}$$

Then the message that node H_N sends to node H_1 at convergence is in the direction of the principal eigenvector of C_2.

The convergence properties of equations similar to 13 are well understood. Roughly speaking, if the principal eigenvector has positive eigenvalue and there is no other eigenvector with that eigenvalue, the iterations will converge to the direction of the principal eigenvalue. The ratio between the magnitude of the largest eigenvalue and the second largest determines the rate of convergence - convergence will be geometric in this ratio. Note that C and $C2$ have identical eigenvalues, since:

$$C_2 = D_1^{-1} C^t D_1 \tag{15}$$

We now show that the correct probabilities can be obtained by the diagonal elements of the matrix C. We denote by p_i the probability that node H_1 is in state i

given the evidence and e_i the vector that is zero everywhere except for a 1 at the ith component then:

$$\begin{aligned}
p_i &= k \sum_{H_2 \cdots H_n} P(H_1 = i, H_2, \cdots H_n) && (16)\\
&= k \sum_{H_2 \cdots H_n} \Psi(i, H_2)\Psi(i, E_1)\Psi(H_2, H_3) && (17)\\
&\quad \Psi(H_2, E_2) \cdots \Psi(H_n, i)\Psi(H_n, E_n) && (18)\\
&= k \sum_{H2} \Psi(i, H_2)\Psi(H_2, E_2) \sum_{H3} \Psi(H_2, H_3) && (19)\\
&\quad \cdots \sum_{H_n} \Psi(H_n, E_n)\Psi(H_n, i)\Psi(i, E_1) && (20)\\
&= k e_i^t M_1^t D_2 M_2^t D_3 \cdots M_n^t D_1 e_i && (21)\\
&= k e_i^t C e_i && (22)
\end{aligned}$$

Note that we can also calculate the normalizing factor k in terms of the matrix C, thus:

$$p_i = \frac{e_i^t C e_i}{trace(C)} \quad (23)$$

Now, recall that the belief function estimated by the loopy network is given by:

$$\tilde{p}_i = \alpha u_i D_1(i,i) v_i \quad (24)$$

where u is the message sent from node H_2 to node H_1 and v is the message sent from node H_N to node H_1. We now express this message in terms of the matrix C. First we write $C = P \Lambda P^{-1}$ where the columns of P contain the eigenvectors of C. We order P such that the principal eigenvector is in the first column, thus $P_{i1} = \beta u_i$. Surprisingly, the first *row* of P^{-1} is related to the product $D_1 v$:

$$C = P \Lambda P^{-1} \quad (25)$$

$$C_2 = D_1^{-1} C^t D_1 = D_1^{-1} (P^{-1})^t \Lambda P^t D_1 \quad (26)$$

Thus the first row of $P^{-1} D_1^{-1}$ gives the principal eigenvector of C_2 or:

$$D_1(i,i) v_i = \gamma P_{1i}^{-1} \quad (27)$$

where the constant γ is independent of of i. Substituting into equation 24 gives:

$$\tilde{p}_i = \alpha P_{i1} P_{1i}^{-1} \quad (28)$$

where α is again a normalizing constant. In fact, this constant is equal to unity since for any invertible matrix P:

$$\sum_i P_{i1} P_{1i}^{-1} = 1 \quad (29)$$

Finally, we can express the relationship between p and $\tilde{p}$:

$$\begin{aligned}
p_i &= \frac{e_i^t C e_i}{trace(C)} && (30)\\
&= \frac{e_i^t P \Lambda P^{-1} e_i}{\sum_j \lambda_j} && (31)\\
&= \frac{\sum_j P_{ij} \lambda_j P_{ji}^{-1}}{\sum_j \lambda_j} && (32)\\
&= \frac{\lambda_1 \tilde{p}_i + \sum_{j=2} P_{ij} \lambda_j P_{ji}^{-1}}{\sum_j \lambda_j} && (33)
\end{aligned}$$

Thus p_i can be written as a weighted average of $\tilde{p}_i$ and a second term, which we will denote by q_i:

$$p_i = \frac{\lambda_1}{\sum_j \lambda_j} \tilde{p}_i + \left(1 - \frac{\lambda_1}{\sum_j \lambda_j}\right) q_i \quad (34)$$

with:

$$q_i = \frac{\sum_{j=2} P_{ij} \lambda_j P_{ji}^{-1}}{\sum_{j=2} \lambda_j} \quad (35)$$

The weight given to $\tilde{p}_i$ is the maximum eigenvalue λ_1. Thus the error in loopy belief propagation is small when the maximum eigenvalue dominates the eigenvalue spectrum:

$$p_i - \tilde{p}_i = \left(1 - \frac{\lambda_1}{\sum_j \lambda_j}\right)(q_i + p_i) \quad (36)$$

Note again the importance of the ratio between the subdominant eigenvalue and the dominant one. When this ratio is small loopy belief propagation converges rapidly, and furthermore the approximation error is small.

In the case of $n = 2$, e.g. binary valued hidden nodes, we can show that even if $\tilde{p}$ is wrong, the state that maximizes $\tilde{p}$ is guaranteed to be the state that maximizes p. In other words, $\tilde{p}$ is ordinally correct.

To show that, note that in this case:

$$\begin{aligned}
p_i &= \frac{\lambda_1 P_{i1} P_{1i}^{-1} + \lambda_2 P_{i2} P_{2i}^{-1}}{\lambda_1 + \lambda_2} && (37)\\
&= \frac{\lambda_1 \tilde{p}_i + \lambda_2 (1 - \tilde{p}_i)}{\lambda_1 + \lambda_2} && (38)
\end{aligned}$$

Thus:

$$p_i - p_j = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} (\tilde{p}_i - \tilde{p}_j) \quad (39)$$

Thus $p_i - p_j$ will be positive if and only if $\tilde{p}_i - \tilde{p}_j > 0$ since the multiplicative factor on the left side of equation is always positive ($\lambda_1 > \lambda_2$, $\lambda_1 > 0$ and $trace(C) > 0$). In other words the estimated probabilities may be wrong, but are guaranteed to be on the correct side of 0.5.

Even more importantly, equation 39 allows us to write the correction term that needs to be added to the estimated probabilities $\tilde{p}_i$ in order to obtain the correct probabilities:

$$p_i = \tilde{p}_i + \frac{\lambda_2}{\lambda_1 + \lambda_2} (2\tilde{p}_i - 1) \quad (40)$$

In order to correct the beliefs, each node must have access to the number $\frac{\lambda_1}{\lambda_1 + \lambda_2}$. However, recall that the convergence rate of the messages is related to the ratio of the eigenvectors. In fact, it is easy to show that:

$$\frac{\|u(t) - u(t-N)\|}{\|u(t-N) - u(t-2N)\|} \rightarrow \frac{|\lambda_2|}{|\lambda_1|} \quad (41)$$

Thus every node, can correct it's belief function by monitoring the convergence rate of its two messages (the ratio only gives the absolute value of λ_2/λ_1, the sign can be obtained by comparing the components of the difference vectors). After correction, the beliefs will be exact.

To illustrate this analysis, figure 3a shows a network of four nodes, connected in a single loop. Figure 3b

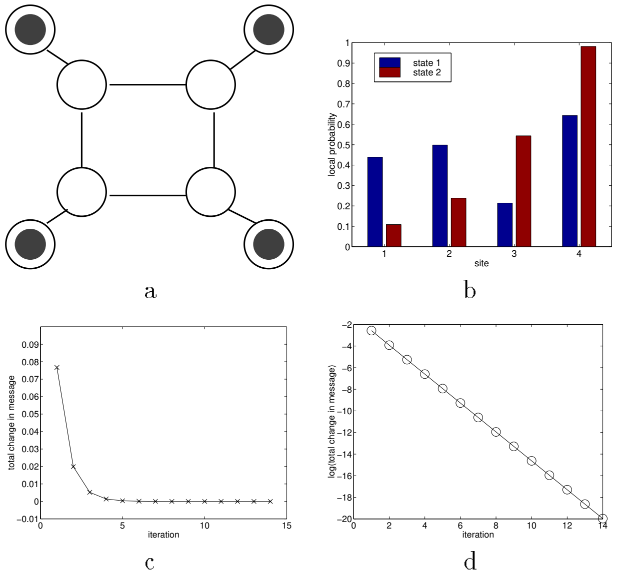


Figure 3: **a.** A simple loop structure. **b.** The local evidence probabilities used in the simulations. **c.** The ratio of differences between messages at successive iterations. Note the geometric convergence. **d.** A log plot of the ratio between messages at successive iterations. The slope of this line gives the ratio of the second and first eigenvalues.

shows the local probabilities, i.e. the probability of a node being in state 0 or 1 given only its local evidence. The transition matrices, were set to:

$$M = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} \tag{42}$$

Figure 3c shows the convergence rate of one of the messages as a function of iteration. The error decreases geometrically, as can be seen in the log plot. The slope of the error decrease, gives the ratio of the two eigenvectors.

Figure 4a shows the correct beliefs (calculated by exhaustive enumeration) and the ones estimated at convergence by the belief propagation algorithm. Note that the estimated beliefs are on the correct side of 0.5 as guaranteed by the theory, but are numerically wrong. In particular, the estimated beliefs are overly confident (closer to 1 and 0, and further from 0.5). Intuitively, this is due to the fact that the loopy network involves counting each evidence multiple times, rather than a single time, thus leading to an overly confident estimate. More formally, this is a result of λ_2 being positive (equation 39). Figure 4c shows the estimated beliefs after the correction of equation 40 has been added. The beliefs are identical to the correct beliefs up to machine precision.

The guarantee of correct decoding and the possibility of correction are only true for binary nodes. For state spaces greater than two, correct MAP estimates are not guaranteed but the relationship between the estimated beliefs and the correct ones (equation 33) predicts that the error will be small when the principal eigenvalue dominates the eigenspectrum, i.e. when convergence is rapid. Figure 5 illustrates this relationship in the simple

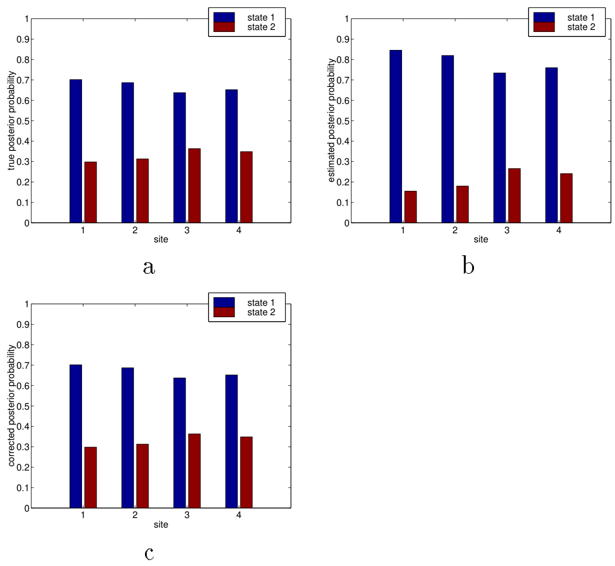


Figure 4: **a.** The correct beliefs (calculated by exhaustive enumeration) for the data in 3. **b.** The beliefs estimated at convergence by the loopy belief propagation algorithm. Note that the estimated beliefs are on the correct side of 0.5 as guaranteed by the theory, but are numerically wrong. In particular, the estimated beliefs are overly confident (closer to 1 and 0, and further from 0.5). **c.** The estimated beliefs after correction based on the convergence rates of the messages (equation 40). The beliefs are identical to the correct beliefs up to machine precision.

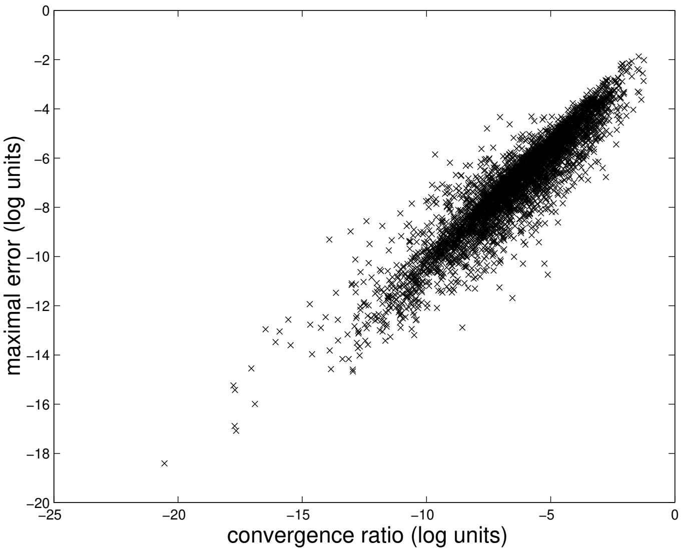


Figure 5: The maximal error between the beliefs estimated using loopy belief update and the true beliefs, plotted as a function of the convergence ratio (equation 41). Equation 36 predicts that the error will be proportional to the convergence ratio. This is illustrated by the simulations.

loop structure when the nodes are allowed to take on three values. We performed belief update on this structure with multiple randomly selected connections and local evidences. We compared the estimated beliefs to the correct ones (estimated using exhaustive enumeration). The error in the estimated beliefs is plotted as a function of the convergence rate (ratio of differences between messages at successive iterations –equation 41). As predicted by this relationship, the error is small when convergence is rapid and large when convergence is slow.

4.2 Networks including a single loop

We now extend the analysis to networks that are singly connected except for a single loop. An example is shown in figure 6a. These networks include arbitrary combinations of chains and trees, except for a single loop. The nodes in these networks can be divided into two categories: those in the loop and those that are not. For example in figure 6 nodes 1–4 are part of the loop while nodes 5–7 are not.

Let us focus first on nodes inside the loop. If all we are interested in is obtaining the correct beliefs for these nodes, then the situation is equivalent to a graph that only has the loop and evidence nodes. That is, after a finite number of iterations, the messages coming into the loop nodes from the non loop nodes will converge to a steady state value. These messages can be thought of as local evidence messages, and the analysis of the messages passed between loop nodes is equivalent to that of a simple loop structure. Again, referring to figure 6, after 2 iterations, the message that node 5 sends to node 4 will not change. This message can be merged with the local evidence at 4, and the message passing between nodes 1–4 is equivalent to a simple loop structure. In other words, the messages will converge to principal eigenvectors of matrices analogous to C and C_2 defined in the previous section, and the rate of convergence will be determined by the ratio of eigenvalues. Using the analysis methods of the previous section, we can again express the relationship between the the beliefs estimated in these nodes and the true posterior probabilities. For binary units, the beliefs estimated can be guaranteed to lie on the correct side of 0.5.

What about nodes outside the loop? Here the beliefs are not guaranteed to be on the correct side of 0.5. Consider node 5 in the figure. Its posterior probability can be factored into three independent sources. The probability given the evidence at 6, at 7 and the evidence in the loop. The messages node 5 receives from 5 and 6 in the belief update algorithm will correctly give the probability given the evidence at these nodes. However, the message it receives from node 4 will be based on the messages passed around the loop multiple times, i.e. evidence in the loop will be counted more times than evidence outside the loop. Again, the extent of the double counting depends on the ratio of the dominant eigenvalue of the matrix C.

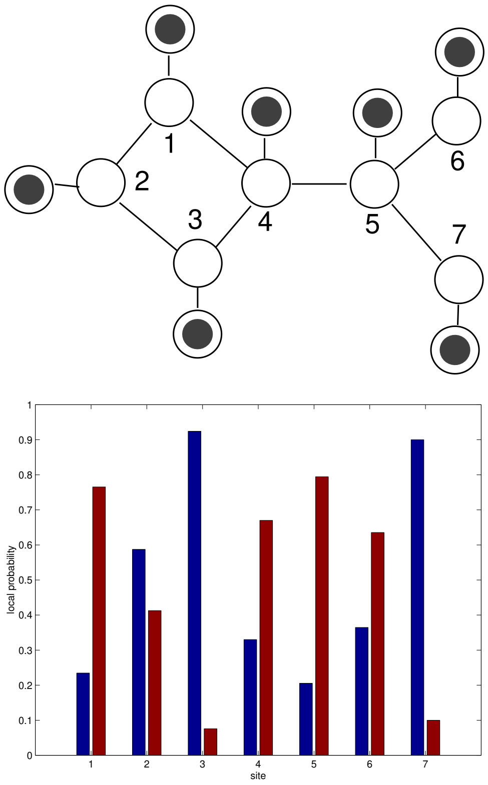
If the nodes are binary valued, then the messages can be corrected using a similar scheme to that described for the simple loop structure. By monitoring the convergence rate, all nodes in the loop can update their beliefs according to equation 40. In addition, the message that nodes in the loop send out to nodes not on the loop is also modified in a similar fashion. Using these local update rules, the network is guaranteed to converge to the correct beliefs.

To illustrate these ideas, consider figures 6–7. Figure 6b shows the local probabilities for each of the seven nodes. The transition matrix was identical to that used in the previous example. Figure 7a shows the correct posterior probabilities calculated using exhaustive enumeration. Figure 7b shows the beliefs estimated using regular belief update on the loopy network. Note that the beliefs for the nodes inside the loop are on the correct side of 0.5 but overly confident. Note however that the belief of node 6 is *not* on the correct side of 0.5. Finally, figure 6c shows the results of the corrected belief propagation algorithm, by using equation 40 for nodes in the loop. The results are identical to the correct posteriors up to machine precision.

5 Belief Revision in Networks with a single loop

For belief revision the situation is simpler than belief update. The result is: if belief revision converges in a network with a single loop, the state sequence that maximizes the steady state beliefs is guaranteed to be the most likely state sequence. The result is independent of the dimensionality of the state space, the length of the loop etc.

The proof uses the notion of the "unwrapped" network introduced in section 3. Here, we will need the notion of an unwrapped network of length n which is



a

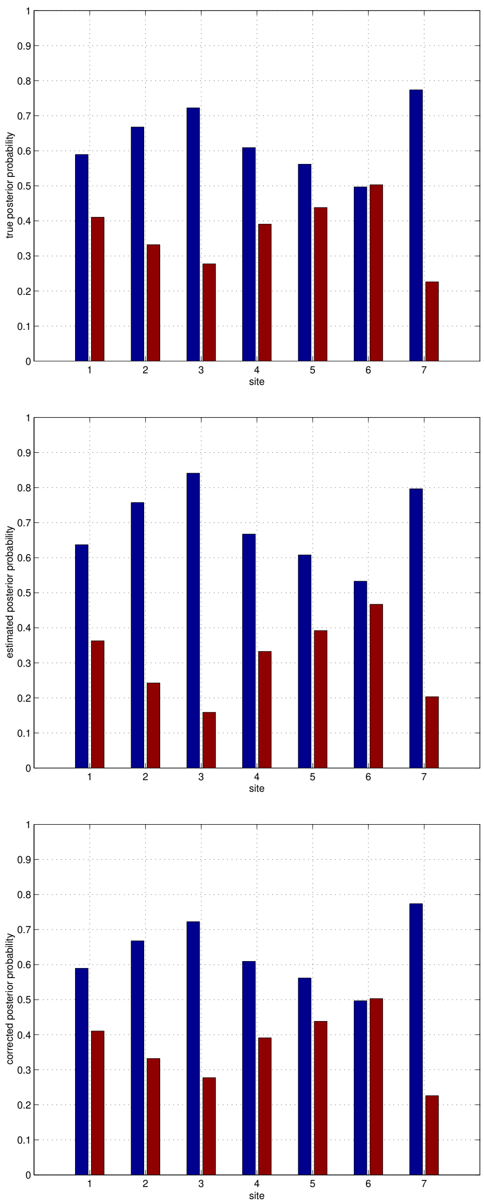
b

Figure 6: **a.** A network including a tree and a loop. **b.** The probability of each node given its local evidence used in the simulations.

defined as follows. We start with an arbitrary node in the loop X and create a chain that goes around the loop n times until it comes back to X. We then add the evidence nodes and the trees that were adjoined to the loop at the corresponding nodes. (see figure 8).

Using induction, it is easy to show that the unwrapped network of size n includes all nodes in the loopy network n times, except for node X which appears $n+1$ times. Furthermore, all connections between neighboring nodes in the loopy network appear n times in the unwrapped network. Finally, all nodes in the unwrapped network have the same neighbors as in the loopy network, except for the first and last copy of node X. For these two copies of X there is one missing neighbor. We denote by H_i the hidden nodes of the original network and by H_i^j the jth replica of this node in the unwrapped network. We refer to the first and last copies of X as the "endpoints" of the unwrapped network.

The essence of the proof is to relate the steady state beliefs in the loopy network to the steady state beliefs in the unwrapped network. Let us consider a concrete example. Table 1a shows the steady state beliefs for the loopy network in figure 8a with arbitrarily chosen local evidence probabilities. Table 1b shows the steady state beliefs in the corresponding unwrapped network which includes 9 replicas of the nodes in the loopy network. Note that except at the endpoints of the unwrapped network, all beliefs are identical to the corresponding ones in the loopy network. Below, we will prove that this is



a

b

c

Figure 7: **a.** The correct posterior probabilities calculated using exhaustive enumeration. **b.** The beliefs estimated using regular belief update on the loopy network. Note that the beliefs for the nodes inside the loop are on the correct side of 0.5 but overly confident. Note however that the belief of node 6 is *not* on the correct side of 0.5.**c.** The results of the corrected belief propagation algorithm by using equation 40 for nodes in the loop. The results are identical to the correct posteriors up to machine precision.

a

b

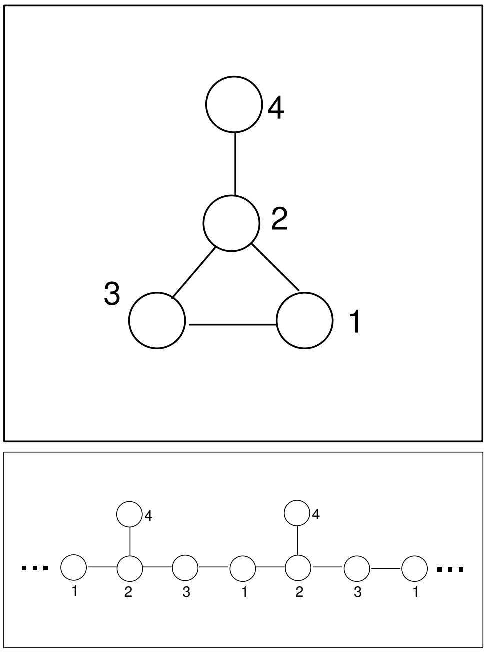


Figure 8: **a.** A simple structure with a loop (evidence nodes not shown for clarity). **b.** The unwrapped network corresponding to it. Note that the unwrapped network includes equal numbers of replicas of all the nodes and connections in the loopy network except for node 1 that appears once too many times.

a

node 1	node 2	node 3	node 4
0.6420	0.5575	0.0407	0.7943

b

replica	node 1	node 2	node 3	node 4
1	0.6811	0.5575	0.0407	0.7943
2	0.6420	0.5575	0.0407	0.7943
3	0.6420	0.5575	0.0407	0.7943
4	0.6420	0.5575	0.0407	0.7943
5	0.6420	0.5575	0.0407	0.7943
6	0.6420	0.5575	0.0407	0.7943
7	0.7290			

Table 1: **a.** The steady state beliefs for the loopy network in figure 8a with arbitrarily chosen local evidence probabilities. The belief of a given node being in state 0 is shown. **b.** The steady state beliefs in the corresponding unwrapped network of length 6 which includes 6 replicas of all nodes in the loopy network, except node 1 that appears once too many. Note that except at the endpoints of the unwrapped network, all beliefs are identical to the corresponding ones in the loopy network. In the text we prove that if belief revision converges, a subsequence of arbitrary length with this property can be found.

inward

leftward

rightward

outward

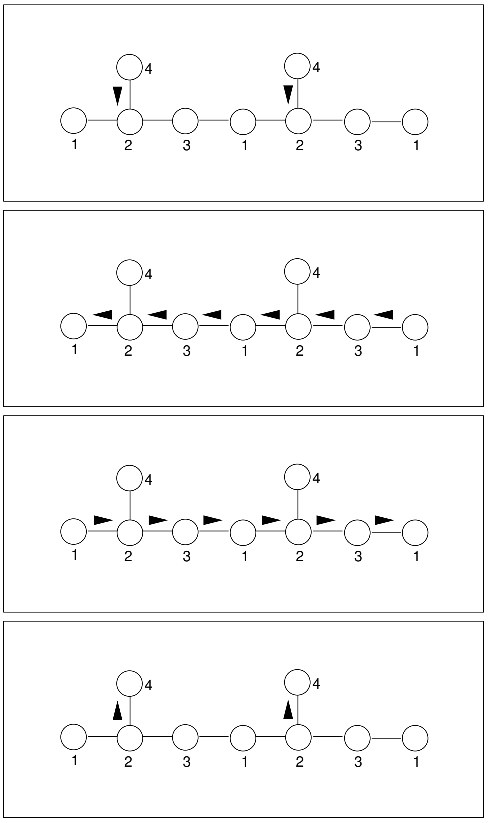


Figure 9: The four categories of messages in the unwrapped network. In the text we prove that if loopy propagation converges then all but a small number of messages in the unwrapped network are equal to the steady state messages in the loopy network.

the case for all networks with a single loop – all but a small number of beliefs in the unwrapped network are identical to those in the loopy network. Since the unwrapped network is a tree, the steady state beliefs are guaranteed to give rise to the most likely sequence for the unwrapped tree. We then show that the most likely sequence for the unwrapped network consists of replicas of the most likely sequence for the loopy network.

Claim: If the loopy propagation messages converge in k iterations, then all messages in the unwrapped network received by nodes whose distance from the endpoints is greater than k are identical to the steady state messages of the loopy propagation scheme.

Proof: The proof follows directly from the fact that all interior nodes in the unwrapped network have the same neighbors as those in the loopy network. In this way, it is similar to the proof given in the appendix which relates the messages received by the center node in the unwrapped network of size n to the messages received by the corresponding node in the loopy network after n iterations. However, when the message passing converges in the loopy network, we can actually relate almost all messages in the unwrapped network to a corresponding message in the loopy network – not just the messages at the center node.

To show this, we divide the nodes in the unwrapped network into two categories - "chain nodes" form part of the infinite chain (corresponding to nodes in the loop in the loopy network) and "nonchain" nodes (corresponding to nodes outside the loop in the loopy network). This gives four types of messages in the unwrapped network (see figure 9). "leftward" and "rightward" messages are the two directions of propagation inside the chain, "outward" are propagated in the nonchain nodes away from the chain nodes and "inward" messages go toward the chain nodes. We now show that for each category of messages, they are identical to the steady state messages in the loopy network provided the nodes are far enough from the endpoints.

The "inward" messages depend only on other inward messages and the evidence at the nonchain nodes. Thus they are identical at all replicas of the nonchain nodes. Similarly, in the loopy network, the corresponding messages do not depend on the iteration, since they are independent of messages passed inside the loop. Since the connectivity of nonchain nodes in the unwrapped network is the same as that of the nonloop nodes in the loopy network, the messages are identical.

The leftward messages depend only on leftward messages to the right, and the inward messages. Since the inward messages are identical to the corresponding steady state message, it is sufficient to show that one leftward message is identical for all leftward messages to the left of that node to be identical. Consider the leftward message transmitted by node X^{n-k}. It is easy to see that it is identical to the message transmitted after k iterations in the loopy network by node X to its neighbor. Thus all leftward messages to the left of node X^{n-k} are identical to the steady state messages in the loopy network.

The proof for the rightward message is analogous, and gives that all rightward messages to the right of X^k are identical to the steady state messages in the loopy network. Finally, the outward messages depend only the leftward and rightward messages, and hence are identical to the steady state messages for all nodes to the right of X^k and to the left of X^{n-k}.

The equivalence of the messages immediately gives an equivalence between the optimal sequence for the unwrapped network and the decoding obtained by maximizing the loopy beliefs.

Claim: Let a^* be the decoding obtained by choosing at an arbitrary node A in the loopy network the state that maximizes the steady state loopy beliefs. The optimal sequence *for the unwrapped problem* has a^* at all corresponding nodes whose distance from the endpoints is greater than k.

Proof: Since the steady state messages are identical, the beliefs at all unwrapped network nodes whose distance from the endpoints is greater than k are identical to the steady state beliefs in the loopy network. Since the unwrapped network is a polytree, choosing the sate that maximizes the beliefs is guaranteed to give the optimal sequence.

We can now prove the main result. Denoting by h_i^* the states that maximize the steady state beliefs in the loopy network, we show that these are the optimal states. First, the preceding discussion means that the optimal sequence for the unwrapped problem has a periodic subsequence consisting of $n-k$ replicas of h_i^*. We now have to show that being optimal in the unwrapped network means being optimal in the loopy network.

It is more convenient here to work with costs than probabilities. We denote by J the negative log probability of the hidden states given the data in the loopy network. J factorizes into a sum of terms corresponding to every link in the loopy network:

$$J(\{h_i\}) = \sum_{kl} J_{kl}(h_k, h_l) + \sum_i J_i(h_i, E_i) \tag{43}$$

where the first sum is taken over all pairs of connected nodes.

Likewise for the unwrapped network, a sequence $\{h_i^j\}$ that is optimal for the unwrapped problem minimizes:

$$J(\{h_i^j\}) = \sum_{ikl} J_{kl}^i(h_k^i, h_l^i) + \sum_{ij} J_i^j(h_i^j, E_i^j) \tag{44}$$

The preceding equation is for a general sequence of states. For a periodic sequence, $h_i^j = h_i$, the log probability of the long sequence is a function of the periodically repeated pattern. Furthermore, we can show that the log probability of a periodic sequence in the unwrapped network is closely related to the log probability in the original loopy network. Consider the subsequence of the unwrapped network starting from X^k and ending with X^{n-k}. Recall that a subnetwork of length n has n replicas of the data of all nodes and n connections between neighboring nodes. Thus if $\{h_i^j\}$ is an optimal *periodic* subsequence of states $h_i^j = h_i$, it must minimize:

$$\begin{aligned} J_n(\{h_i^j\}) &= (n-2k)\sum_{kl} J_{kl}(h_k, h_l) && (45)\\ &\quad +(n-2k)\sum_i J_i(h_i, E_i) + \tilde{J}(h_0) && (46) \end{aligned}$$

The last term captures the fact that the node X in the subsequence is neighbored by two additional nodes that constrain it, and and that node X appears one time too many as compared to the other nodes. If we assume that all states have positive probability, this last term is finite.

This leads to:

$$J_n(\{h_i^j\}) = (n-2k)J(\{h_i\}) + \tilde{J}(h_0) \tag{47}$$

Thus the function minimized by a periodic sequence in the unwrapped network is dominated, for large n, by a multiple of the function to be minimized in the loopy network. This implies that the periodically repeated pattern in the subsequence, must be an optimal sequence in the original loopy network. Formally, denoting again by h_i^* the decoding obtained by maximizing the steady state beliefs in the loopy network. Suppose this decoding was not optimal, i.e. there existed a sequence $\{h_i\}$ such that $J(\{h_i\}) - J(\{h_i^*\}) = d < 0$. Create the periodic sequence of length $n-2k$ that replicates the sequence $\{h_i\}$ $n-2k$ times. Then:

$$J_n(\{h_i\}) - J_n(\{h_i^*\}) = (n-2k)d + \tilde{J}(h_0) - \tilde{J}(h_0^*) \tag{48}$$

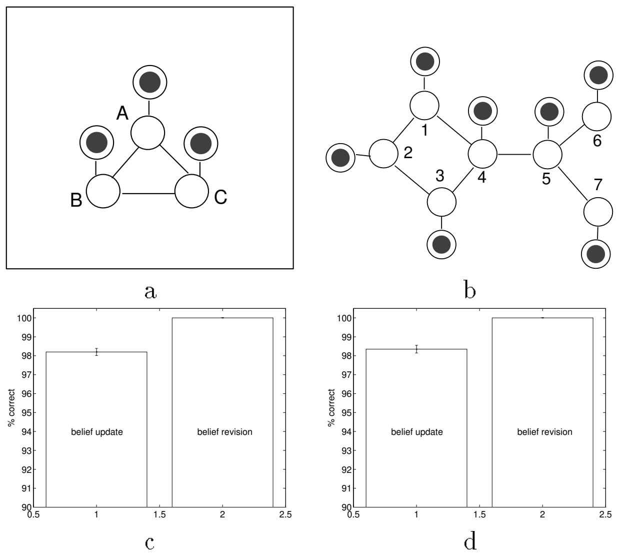


Figure 10: **a-b.** Two structures for which belief update need not give the optimal decoding. A simple loop structure with state spaces are of size $n = 3$, and a loop adjoined to a tree. In both these structures belief revision when it converges is guaranteed to give the correct decoding. **c-d.** The number of correct decodings using belief update and belief revision in 5000 networks of these structure with randomly generated local evidence probabilities. Consistent with the analysis, belief update may give an incorrect decoding, but an incorrect decoding based on belief revision is never observed.

Since this is true for all n and $\tilde{J}(h_0)$ is finite, this implies there exists an n such that $J_n(\{h_i\}) < J_n(\{h_i^*\}$ in contradiction to the fact that $\{h_i^*\}$ is the optimal subsequence for the unwrapped network. Therefore, h_i^* is the optimal decoding for the loopy network.

To illustrate this analysis, figure 10 shows two network structures for which there is no guarantee that belief update will give the right answer (the loop structure has state space of length three), but according to our proof, belief revision should always give the correct answer. We selected 5000 random connections and local evidences for each of the two structures, and calculated the correct beliefs using exhaustive enumeration. We then counted the number of correct decodings using belief update and belief revision (a decoding based on belief update was judged correct if the state at each node maximized the true posterior probability, and a decoding based on belief revision was judged correct if it gave the most likely sequence). If belief revision did not converge (i.e. entered a limit cycle) its decoding was not used in the results. A limit cycle was observed about 5% of the simulations.

The results are shown in figure 10. In both of these structure, belief update is not guaranteed to give the correct decoding and indeed the wrong decoding is obtained in a small number of cases. On the other hand, belief revision when it converges is guaranteed to give the optimal sequence and indeed a wrong decoding was never observed.

5.1 Discussion - single loop

To summarize, the analysis enables us to categorize a given network with a loop into one of two categories - (1) structures for which both belief update and belief revision are guaranteed to give a correct decoding, or (2) those for which only belief revision is guaranteed to give the correct decoding.

The crucial element determining the performance of belief update is the eigenspectrum of the matrix C. Note that this matrix depends both on the transition matrices and the observed data. Thus belief update may give very exact estimates for certain observed data even in structures for which it is not guaranteed to give the correct decoding.

We have primarily dealt here with discrete state spaces, but the analysis can also be carried out for continuous spaces. For the case of Gaussian posterior probabilities, it is easy to show that belief update will give the correct MAP estimate for all structures with a single loop (this is simply due to the fact that belief update and revision are identical for a Gaussian).

6 Loopy networks with multiple loops

The basic intuition that explains why belief propagation works in networks with a single loop is the notion of "equal double counting". Essentially, while evidence is double counted, all evidence is double counted equally as can be seen in the unwrapped network and hence belief revision is guaranteed to give the correct answer. Furthermore, by using recursion expressions for the messages we could quantify the "amount" of double counting, and derive a relationship between the estimated beliefs and the true ones.

In networks containing multiple loops, quantifying the amount of double counting is far less simple. However, using the unwrapped network we can prove for some structures that they lead to "equal double counting". Thus we would expect belief revision to give the optimal decoding in these structures. Again, the situation is more complex then in the single loop case — even in those structures belief revision may converge to an incorrect decoding. However, as we show in the simulations this happens far less frequently as compared to errors using belief update.

We define a loopy network as *balanced* if one can generate an unwrapped network of arbitrarily large size that will contain an equal number of replicas of all nodes and connections (excluding boundary nodes). For example, consider the network with multiple loops shown in figure 11a. The unwrapped network corresponding to it is shown in figure 11b. Using induction, we can show that there exist arbitrarily large unwrapped networks where all nodes and connections appear the same number of times, except for the boundary nodes that occur too many times.

Thus using a similar analysis to that used in the previous section, it can be shown that if $\{h_i^*\}$ are the states obtained by maximizing the steady state beliefs, they must maximize:

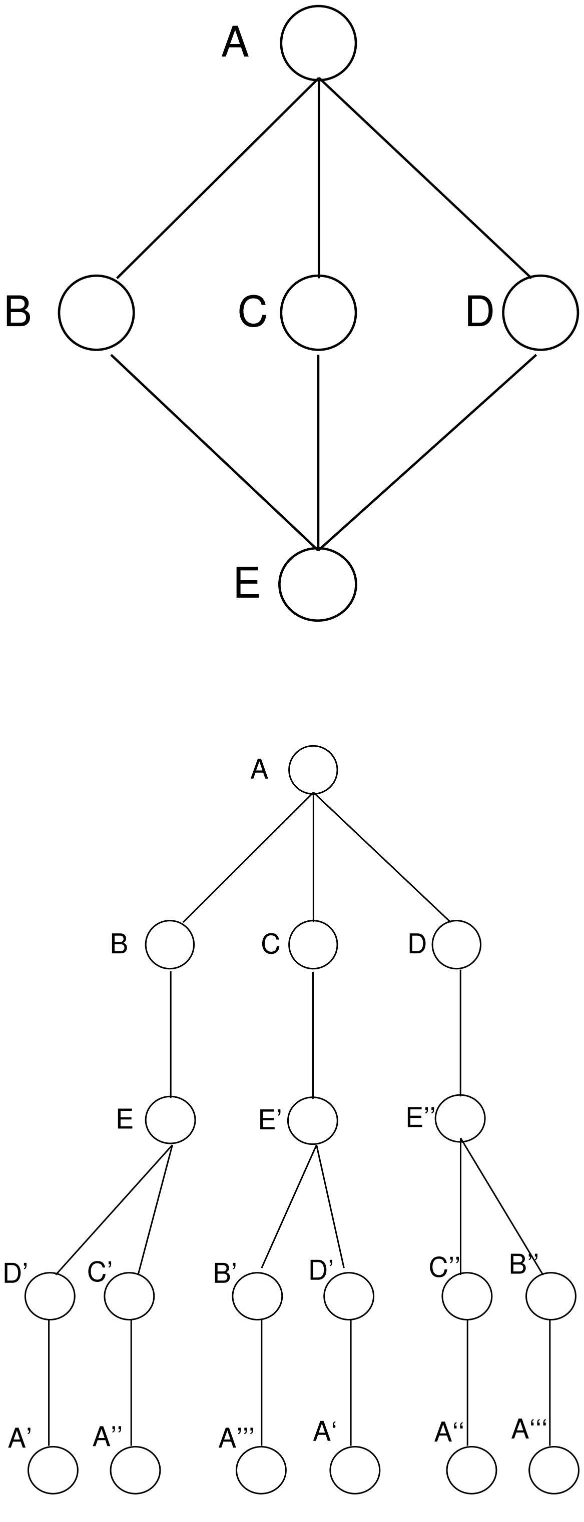


Figure 11: **Top:** A structure with multiple loops. Belief propagation in this structure is equivalent to the turbo decoding algorithm for a message of length 3. In a coding application nodes B, C, D are the unknown three bits, nodes A and B are the transmitted codewords. The compatibility constraints are nonprobabilistic. . **Bottom:** The unwrapped network corresponding to this structure. Note that all nodes and connections in the loopy structure appear equal numbers of time in the unwrapped network, except the boundary nodes A. Thus the Turbo code structure is balanced.

$$J_n(\{h_i\}) = nJ(\{h_i\}) + \tilde{J}_n(h_0) \qquad (49)$$

for arbitrarily large n. Here $\tilde{J}_n(h_0)$ is a boundary cost reflecting the constraints imposed on the endpoints of the unwrapped subnetwork. Note that unlike the single loop case, the boundary cost here may increase with increasing n. In other words the unwrapped network for a network with a single loop is essentially an infinite chain, and hence there are only two endpoint nodes for any finite n. For networks with multiple loops, the unwrapped network is an infinite tree, and the number of endpoints (i.e. leaf nodes) can grow exponentially with n.

Because the boundary cost may grow with n, we can *not* automatically assume that if a sequence maximizes J_n it also maximizes J, i.e. the sequence may be optimal only because of the boundary cost. We have not been able to analytically determine the conditions under which the steady state sequence is due to boundary effects as opposed to the cost functional J. Note that if belief propagation converges, one can choose unwrapped subnetworks with different boundary conditions that give the same optimal sequences. Thus a necessary (but *not* sufficient) condition for the state to be optimal is that loopy belief propagation converges to a steady state from multiple initial conditions.

To illustrate this calculation, we performed loopy belief revision on the structure shown in the figure using randomly selected transition matrices and local evidence probabilities. We repeated the experiment 10000 times and counted the number of correct decodings. As in the previous section, when belief revision yielded a limit cycle, we did not use its decoding. We never observed a convergence of belief revision to an incorrect decoding for this structure, while belief update gave 247 incorrect decodings (error rate 2.47%). Since these rates are based on 10,000 trials, the difference is highly significant.

6.1 Turbo codes

As mentioned in the introduction. The Turbo decoding algorithm has been shown to be equivalent to loopy belief propagation. Indeed, for a message of length 3 the Turbo code decoding algorithm reduces to belief propagation on the structure discussed in the previous section (figure 11) with the following properties:

- the states in the middle row are the hidden message bits. They are binary valued. They have local evidence corresponding to the uncoded transmitted message.
- the states in the top and bottom are the two coded message states. They have state space 2^n and each has local observations which imposes a local probability for any message of length n.
- the compatibility constraints between the hidden nodes of the network are nonprobabilistic. Given a state for one of the message nodes, the bitwise nodes are completely determined such that the kth bit node has the same value as the kth bit in the state of the message node.

McEliece et al. (1995) showed several examples where the turbo decoding algorithm converges to an incorrect

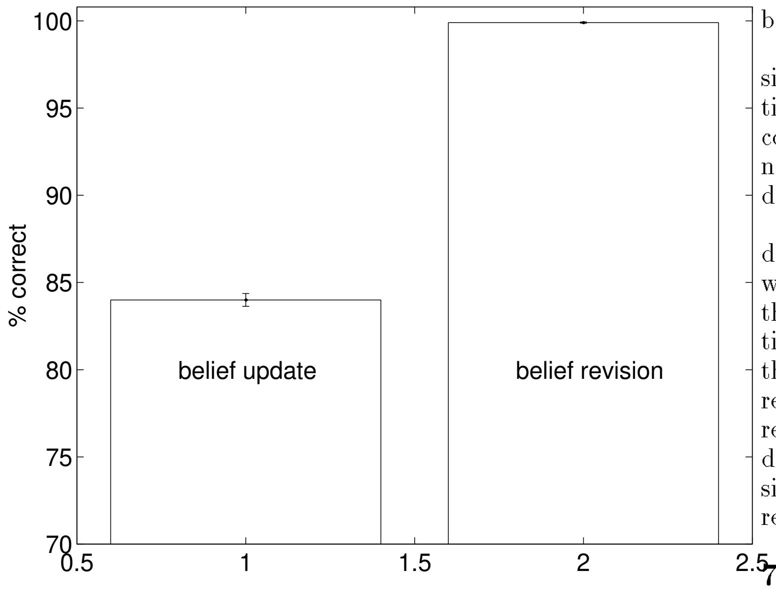


Figure 12: The results of belief revision and update on the turbo code problem.

decoding on this structure. Since the Turbo decoding algorithm is equivalent to belief update, we wanted to see how well a decoding algorithm equivalent to *belief revision* will perform. Since the Turbo code structure is *balanced*, we would expect such an algorithm to very rarely converge to an incorrect answer.

A variant of the Turbo decoding algorithm that is equivalent to belief revision was used by Benedetto et al. (1996). They implemented the decoding algorithm in circuits and this favors belief revision (the maximum operation is easier to implement than the sum operation). Benedetto et al. found that the bitwise error rate using belief revision was slightly worse than using belief update. This is however a slightly unfair comparison — even in singly connected networks belief revision will not minimize the bit error rate but rather the block error rate. Which of the two error rates should be used is, of course, application dependent. Traditionally block error rates are used for block codes and bit error rates are used for convolutional codes [13].

Here we are mostly interested in evaluating the error introduced by the suboptimal decoding algorithm. Thus we compare belief revision and belief update based on how often they agree with their corresponding optimal algorithms. We calculated the number of incorrect decodings for $10,000$ randomly selected probability distributions on the two received messages. The belief update decoding was judged as correct if it agreed with the optimal bitwise decoding, and the belief revision decoding was correct if it agreed with the maximum likelihood decoding. We only considered decodings where loopy belief propagation converged.

Results are shown in figure 12. As observed by McEliece et al. turbo decoding is quite likely to converge to an incorrect decoding (16% of the time). In contrast, belief revision converges to an incorrect answer less than 0.1% of the runs. Note that this difference is based on $10,000$ trials and is highly significant.

In the few incorrect answers obtained in belief revision, the incorrect answer depended on the initial conditions. Different initial conditions gave convergence to the correct decoding. This is consistent with equation 49 — nonoptimal decodings will arise only if the final answer depends on the boundary conditions.

Given the relatively high likelihood of an incorrect decoding with the standard Turbo decoding algorithm, why do Turbo codes work? Part of the answer may lie in the regime in which they are used in most coding applications. Typically, the probability distribution induced by the two coded messages is highly peaked around the correct message. In such a regime, belief update is closely related to belief revision, much more so than in the randomly selected probability distributions used in these simulations. Analyzing the behavior of belief update and revision in this regime is a topic of current research.

7 Conclusion

As Pearl has observed, in order for belief propagation to be exact, it must find a way to avoid double counting. Since double counting is unavoidable in networks with loops, belief propagation in such networks was widely believed to be a bad idea. The excellent performance of Turbo codes motivates a closer look at belief propagation in loopy networks. Here we have shown a class of networks for which, even though the evidence is double counted, all evidence is equally double counted. For networks with a single loop we have obtained an analytical expression relating the beliefs estimated by loopy belief propagation and the correct beliefs. This allows us to find structures in which belief update may give the wrong answer but is guaranteed to give the correct MAP estimate. We have also shown that for networks with a single loop belief revision is guaranteed to give the maximum likelihood decoding. For networks with multiple loops we have introduced the notion of a balanced network and given a necessary condition for the decoding using belief revision to be optimal. We have also shown that Turbo codes induce a balanced network and presented simulation results indicating belief revision is significantly more likely to give a correct decoding. These results are an encouraging first step towards understanding belief propagation in networks with loops.

Acknowledgments

I thank E. Adelson, M. Jordan, P. Dayan, M. Meila, Q. Morris and J. Tenenbaum for helpful discussions and comments.

References

[1] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara. Soft-output decoding algorithms in iterative decoding of turbo codes. Technical Report 42–124, JPL TDA, 1996.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo codes. In *Proc. IEEE International Communications Conference '93*, 1993.

[3] D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice Hall, 1987.

[4] B.J. Frey and F.R. Kschischang. Probability propagation and iterative decoding. In *Proc. 34th Allerton Conference on Communications, Control and Computing*, 1996.

[5] Brendan J. Frey. *Bayesian Networks for Pattern Classification, Data Compression and Channel Coding.* MIT Press, 1997.

[6] Arthur Gelb, editor. *Applied Optimal Estimation.* MIT Press, 1974.

[7] D. J. C. MacKay, R.J. McEliece, and J.F. Cheng. Turbo decoding as as an instance of pearl's 'belief propagation' algorithm. *IEEE Journal on Selected Areas in Communication*, page in press, 1997.

[8] D.J.C. Mackay and Radford M. Neal. Good error-correcting codes based on very sparse matrices. In *Cryptography and Coding - LNCS 1025.* 1995.

[9] R.J. McEliece, E. Rodemich, and J.F. Cheng. The turbo decision algorithm. In *Proc. 33rd Allerton Conference on Communications, Control and Computing*, 1995.

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

[11] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989.

[12] P. Smyth, D. Heckerman, and M. I. Jordan. Probabilistic independence networks for hidden markov probability models. *Neural Computation*, 1997.

[13] A.J. Viterbi, A.M. Viterbi, and N.T. Sinhushayana. Interleaved concatenated codes: new perspectives on approaching the Shannon limit. *Proc. Natl Acad. Sci. USA*, 94:9525–9531, 1997.

[14] Y. Weiss. Interpreting images by propagating bayesian beliefs. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, 1996.

Generating the unwrapped network

The formal algorithm for generating the unwrapped network is to generate an infinite tree. With each node in the tree we associate a pointer to the original loopy network to which it is identical. We pick an arbitrary node in the loopy network as the root of the tree. We then iterate the following procedure:

- find all leafs of the tree (nodes that have no children).
- For each leaf, find all k nodes in the loopy graph that neighbor the node corresponding to this leaf.
- Add $k-1$ nodes as children to each leaf, corresponding to all neighbors except the parent node.

We will refer to the unwrapped tree generated after n iterations with root node corresponding to node X as the nth unwrapped tree with root X. Note that if Z and Y are the neighbors of X the nth unwrapped tree with root X consists of connecting two subtrees of the $n-1$ unwrapped trees with root nodes Z and Y. The subtrees are obtained from the $n-1$ unwrapped tree by deleting node X and its children from the top.

Claim: The messages that a node X receives after n parallel updates of belief revision in the loopy network are the exact same messages that the root node receives at steady state in the nth unwrapped tree with root X.

Proof: The proof for the case $n = 1$ is trivial. Node X will received a vector of ones from all non evidence neighbors, and the local evidence message from its evidence neighbor. The unwrapped tree will consist of the immediate neighbors of X and the steady state messages are identical. We proceed by induction. The messages that node Y sends to node X at time n is given by combining all the messages Y received at time $n-1$ from all of its neighbors excluding X. The message root X receives from a node corresponding to Y in the nth order unwrapped tree, is obtained by combining all messages arriving at Y in the $n-1$ unwrapped tree whose root is Y. These messages, by the induction assumption, will be identical to the messages received by node Y in the loopy network at time $n-1$.