On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs.

Yair Weiss and William T. Freeman

Abstract

Graphical models, such as Bayesian networks and Markov random fields, represent statistical dependencies of variables by a graph. The max-product "belief propagation" algorithm is a local-message passing algorithm on this graph that is known to converge to a unique fixed point when the graph is a tree. Furthermore, when the graph is a tree, the assignment based on the fixed-point yields the most probable a posteriori (MAP) values of the unobserved variables given the observed ones.

Recently, good empirical performance has been obtained by running the max-product algorithm (or the equivalent min-sum algorithm) on graphs with loops, for applications including the decoding of "turbo" codes. Except for two simple graphs (cycle codes and single loop graphs) there has been little theoretical understanding of the max-product algorithm on graphs with loops.

Here we prove a result on the fixed points of max-product on a graph with *arbitrary topology* and with *arbitrary probability distributions* (discrete or continuous valued nodes). We show that the assignment based on a fixed-point is a "neighborhood maximum" of the posterior probability: the posterior probability of the max-product assignment is guaranteed to be greater than all other assignments in a particular large region around that assignment. The region includes all assignments that differ from the max-product assignment in any subset of nodes that form no more than a single loop in the graph. In some graphs this neighborhood is exponentially large. We illustrate the analysis with examples.

Problems involving probabilistic belief propagation arise in a wide variety of applications, including error correcting codes, speech recognition and image understanding. Typically, a probability distribution is assumed over a set of variables and the task is to infer the values of the unobserved variables given the observed ones. The assumed probability distribution is described using a graphical model [13] — the qualitative aspects of the distribution are specified by a graph structure. The graph may either be directed as in a Bayesian network [17], [11] or undirected as in a Markov Random Field [17], [9].

Here we focus on the problem of finding an assignment for the unobserved variables that is most probable given the observed ones. In general, this problem is NP hard [18] but if the graph is singly connected (i.e. there is only one path between any two given nodes) then there exist efficient local message–passing schemes to perform this task. Pearl [17] derived such a scheme for singly connected Bayesian networks. The algorithm, which he called "belief revision", is identical to his algorithm for finding posterior marginals over nodes except that the summation operator is replaced with a maximization. Aji et al. [2] have shown that both of Pearl's algorithms can be seen as special cases of generalized distributive laws over particular semirings. In particular, Pearl's algorithm for finding maximum a posteriori (MAP) assignments can be seen as a generalized distributive law over the max-product semiring. We will henceforth refer to it as the "max-product" algorithm.

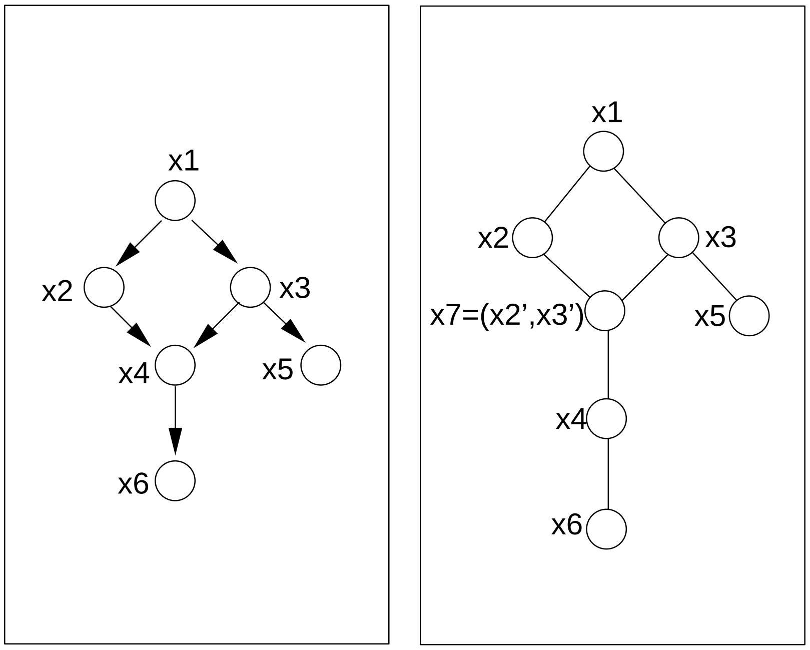
Pearl showed that for singly connected networks, the max-product algorithm is guaranteed to converge and that the assignment based on the messages at convergence is guaranteed to give the optimal assignment–values corresponding to the MAP solution.

Several groups have recently reported excellent experimental results by running the max-product algorithm on graphs with loops [22], [6], [3], [19], [6], [10]. Benedetto et al. used the max-product algorithm to decode "turbo" codes and obtained excellent results that were slightly inferior to the original turbo decoding algorithm (which is equivalent to the sum-product algorithm). Weiss [19] compared the performance of sum-product and max-product on a "toy" turbo code problem while distinguishing between converged and unconverged cases. He found that if one considers only the convergent cases, the performance of max-product decoding is significantly better than sum-product decoding. However, the max-product algorithm converges less often so its overall performance (including both convergent and nonconvergent cases) is inferior.

Progress in the analysis of the max-product algorithm has been made for two special topologies: single loop graphs, and "cycle codes". For graphs with a single loop [22], [19], [20], [5], [2], it can be shown that the algorithm converges to a stable fixed point or a periodic oscillation. If it converges to a stable fixed-point, then the assignment based on the fixed-point messages is the optimal assignment. For graphs that correspond to cycle codes (low density parity check codes in which each bit is checked by exactly two check nodes), Wiberg [22] gave sufficient conditions for max-product to converge to the transmitted codeword and Horn [10] gave sufficient conditions for convergence to the MAP assignment.

In this paper we analyze the max-product algorithm in graphs of *arbitrary topology.* We show that at a fixed-point of the algorithm, the assignment is a "neighborhood maximum" of the posterior probability: the posterior probability of the max-product assignment is guaranteed to be greater than all other assignments in a particular large region around that assignment. These results motivate using this powerful algorithm in a broader class of networks.

Y. Weiss is with Computer Science Division, 485 Soda Hall, UC Berkeley, Berkeley, CA 94720-1776. E-mail: yweiss@cs.berkeley.edu.
Support by MURI-ARO-DAAH04-96-1-0341, MURI N00014-00-1-0637 and NSF IIS-9988642 is gratefully acknowledged.
W. T. Freeman is with MERL, Mitsubishi Electric Research Labs., 201 Broadway, Cambridge, MA 02139. E-mail: freeman@merl.com.



a b

Fig. 1. Any Bayesian network can be converted into an undirected graph with pairwise cliques by adding cluster nodes for all parents that share a common child. **a.** A Bayesian network. **b.** The corresponding undirected graph with pairwise cliques. A cluster node for (x_2, x_3) has been added. The potentials can be set so that the joint probability in the undirected graph is identical to that in the Bayesian network. In this case the update rules presented in this paper reduce to Pearl's propagation rules in the original Bayesian network [20].

I. The max-product algorithm in pairwise Markov Random Fields

Pearl's original algorithm was described for directed graphs, but in this paper we focus on undirected graphs. Every directed graphical model can be transformed into an undirected graphical model before doing inference (see figure 1). An undirected graphical model (or a Markov Random Field) is a graph in which the nodes represent variables and arcs represents compatibility relations between them. Assuming all probabilities are nonzero, the Hammersley-Clifford theorem (e.g. [17]) guarantees that the probability distribution will factorize into a product of functions of the maximal cliques of the graph.

Denoting by x the values of all unobserved variables in the graph, the factorization has the form:

$$P(x) = \prod_c \Psi_c(x_c) \tag{1}$$

where x_c is a subset of x that form a clique in the graph and Ψ_c is the potential function for the clique.

We will assume, without loss of generality, that each x_i node has a corresponding y_i node that is connected only to x_i.

Thus:

$$P(x, y) = \prod_c \Psi_c(x_c) \prod_i \Psi_{ii}(x_i, y_i) \tag{2}$$

The restriction that all the y_i variables are observed and none of the x_i variables are is just to make the notation simple — $\Psi_{ii}(x_i, y_i)$ may be independent of y_i (equivalent to y_i being unobserved) or $\Psi_{ii}(x_i, y_i)$ may be $\delta(x_i - x_o)$ (equivalent to x_i being observed, with value x_o).

In describing and analyzing belief propagation, we assume the graphical model has been preprocessed so that all the cliques consist of pairs of units. Any graphical model can be converted into this form before doing inference through a suitable clustering of nodes into large nodes [20]. Figure 1 shows an example — a Bayesian network is converted into an MRF in which all the cliques are pairs of units.

Equation 2 becomes

$$P(x, y) = \prod_{i,j} \Psi_{ij}(x_i, x_j) \prod_i \Psi_{ii}(x_i, y_i) \tag{3}$$

where the first product is over connected pairs of nodes.

The important property of MRFs that we will use throughout this paper is the *Markov blanket* property. The probability of a subset of nodes S given all other nodes in the graph S^C depends only on the values of the nodes that immediately neighbor S. Furthermore, the probability of S given all other nodes is proportional to the product of all clique potentials within S and all clique potentials between S and its immediate neighbors.

$$P(x_S|x_{S^C}) = P(x_S|N(x_S)) \quad (4)$$
$$= \prod_{x_i,x_j \in S} \Psi_{ij}(x_i,x_j) \prod_{x_i \in S, x_j \in N(S)} \Psi_{ij}(x_i,x_j) \quad (5)$$

The advantage of preprocessing the graph into one with pairwise cliques is that the description and the analysis of belief propagation becomes simpler. For completeness, we review the belief propagation scheme used in [20]. As we discuss in the appendix, this belief propagation scheme is equivalent to Pearl's belief propagation algorithm in directed graphs, the Generalized Distributive Law algorithm of [1] and the factor graph propagation algorithm of [12]. These three algorithms correspond to the algorithm presented here with a particular way of preprocessing the graph in order to obtain pairwise potentials.

At every iteration, each node sends a (different) message to each of its neighbors and receives a message from each neighbor. Let x_i and x_j be two neighboring nodes in the graph. We denote by $m_{ij}(x_j)$ the message that node x_i sends to node x_j, by $m_{ii}(x_i)$ the message that y_i sends to x_i, and by $b_i(x_i)$ the belief at node x_i.

The max-product update rules are:

$$m_{ij}(x_j) \leftarrow \alpha \max_{x_i} \Psi_{ij}(x_i,x_j)\, m_{ii}(x_i) \prod_{x_k \in N(x_i) \backslash x_j} m_{ki}(x_i) \quad (6)$$

$$b_i(x_i) \leftarrow \alpha\, m_{ii}(x_i) \prod_{x_k \in N(x_i)} m_{ki}(x_i) \quad (7)$$

where α denotes a normalization constant and $N(x_i) \backslash x_j$ means all nodes neighboring x_i, except x_j.

The procedure is initialized with all message vectors set to constant functions. Observed nodes do not receive messages and they always transmit the same vector— if y_i is observed to have value y^* then $m_{ii}(x_i) = \Psi_{ii}(x_i, y^*)$. The normalization of m_{ij} in equation 6 is not necessary–whether or not the message are normalized, the belief b_i will be identical. However, normalizing the messages avoids numerical underflow and adds to the stability of the algorithm. We assume throughout this paper that all nodes simultaneously update their messages in parallel.

For singly connected graphs it is easy to show that:

- The algorithm converges to a unique fixed point regardless of initial conditions in a finite number of iterations.
- At convergence, the belief for any value x_i of a node i is the maximum of the posterior, conditioned on that node having the value x_i: $b_i(x_i) = \alpha \max_x P(x|y, x_i)$.
- Define the max-product assignment, x^* by $x_i^* = \arg\max_{x_i} b_i(x_i)$ (assuming a unique maximizing value exists). Then x^* is the MAP assignment.

The max product assignment assumes there are no "ties" — that a unique maximizing x_i exists for all $b(x_i)$. Ties can arise when the MAP assignment is not unique, e.g. when there are two assignments that have identical posterior and both maximize the posterior. For singly connected graphs, the converse is also true: if there are no ties in $b(x_i)$ then the MAP assignment is unique. In what follows, we assume a unique MAP assignment.

In particular applications, it might be easier to work in the log domain so that the product operation in equations 7 is replaced by a sum operations. Thus the max-product algorithm is sometimes referred to as the max-sum algorithm or the min-sum algorithm [22], [5]. If the graph is a chain, the max-product is a two-way version of the Viterbi algorithm in Hidden Markov Models and is closely related to concurrent dynamic programming [4]. Despite this connection to well studied algorithms, there has been very little analytical success in characterizing the solutions of the max-product algorithm on arbitrary graphs with loops.

II. What are the fixed points of the max-product algorithm?

Each iteration of the max-product algorithm can be thought of as an operator F that inputs a list of messages $m^{(t)}$ and outputs a list of messages $m^{(t+1)} = Fm^{(t)}$. Thus running belief propagation can be thought of as an iterative way of finding a solution to the fixed point equations $Fm = m$ with an initial guess $m^{(0)}$ in which all messages are constant functions.

Note that this is not the only way of finding fixed-points. Murphy et al. [16] describe an alternative method for finding fixed-points of F. They suggested iterating:

$$m^{(t+1)} = F\left(\alpha m^{(t)} + (1-\alpha) m^{(t-1)}\right) \quad (8)$$

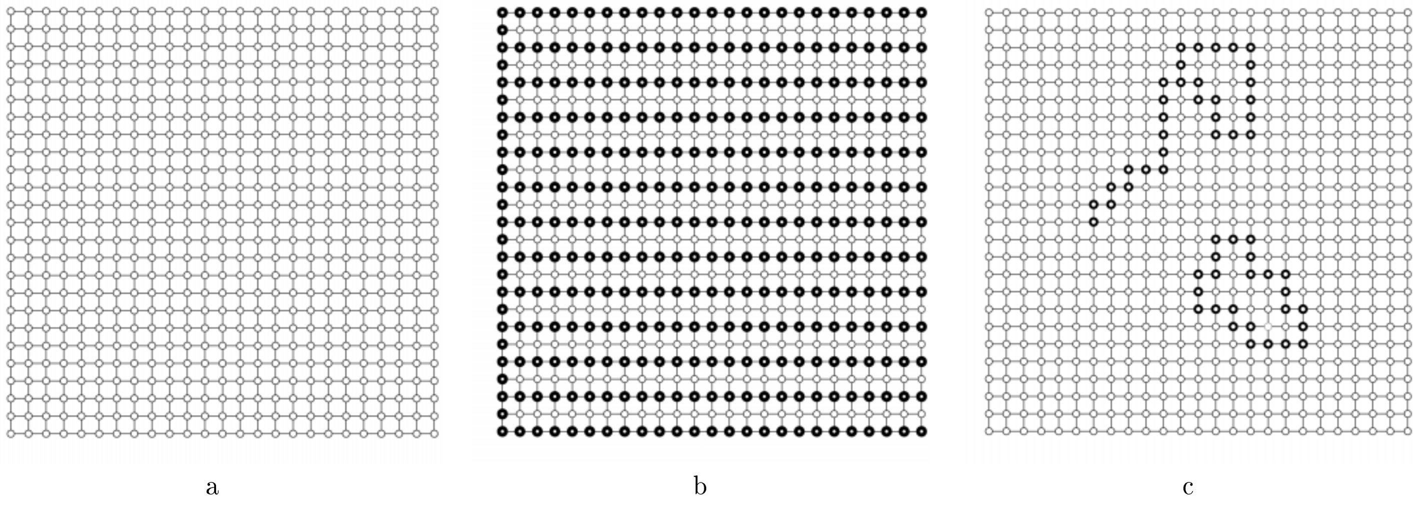


Fig. 2. **a.** A 25x25 grid of points. **b.-c.** Examples of subsets of nodes that form no more than a single loop. In this paper we prove that changing such a subset of nodes from the max-product will always decrease the posterior probability

Here again, if iterations of equation 8 converge to m^* then m^* satisfies $m^* = Fm^*$.

The equation $m^* = Fm^*$ is a highly nonlinear equation and it is not obvious how many solutions exist or how to characterize them. Horn [10] showed that in single-loop graphs, F can be considered as matrix multiplication over the max-product semiring and fixed points correspond to eigenvectors of that matrix. Thus even in single loop graphs, one can construct examples with any number of fixed points.

The main result of this paper is a characterization of how well the max-product assignment approximates the MAP assignment. We show that the assignment x^* must be a neighborhood maximum of $P(x|y)$: that is $P(x^*|y) > P(x|y)$ for all x in a particular large region around x^*. This condition is weaker than a global maximum but stronger than a local maximum.

To be more precise we define the *Single Loops and Trees (SLT)* neighborhood of an assignment x^* in a graphical model G to include all assignments x that can be obtained from x^* by:

- Choosing an arbitrary subset S of nodes in G that consists of disconnected combinations of trees and single loops.
- Assigning arbitrary values to x_S — the chosen subset of nodes. The other nodes have the same assignment as in x^*.

Claim 1: For an arbitrary graphical model with arbitrary potentials, if m^* is a fixed-point of the max-product algorithm and x^* is the assignment based on m^* then $P(x^*|y) > P(x|y)$ for all $x \neq x^*$ in the SLT neighborhood of x^*.

Figure 2 illustrates example configurations within the the SLT neighborhood of the max-product assignment. It shows examples of subsets of nodes that could be changed to arbitrary values and the posterior probability of the assignment is guaranteed to be worse than that of the max-product assignment.

To build intuition, we first describe the proof for a specific case, the diamond graph of figure 3. The general proof is given in section II-B.

A. Specific Example

We start by giving an overview of the proof for the diamond graph shown in figure 3a. The proof is based on the unwrapped tree — the graphical model that the loopy belief propagation is solving exactly when applying the belief propagation rules in a loopy network [8], [22], [20], [21]. In error-correcting codes, the unwrapped tree is referred to as the "computation tree" — it is based on the idea that the computation of a message sent by a node at time t depends on messages it received from its neighbors at time $t-1$ and those messages depend on the messages the neighbors received at time $t-2$ etc.

Figure 3 shows an unwrapped tree around node x_1 for the diamond shaped graph on the left. Each node has a shaded observed node attached to it that is not shown for simplicity.

To simplify notation, we assume that x^*, the assignment based on a fixed-point of the max-product algorithm is equal to zero, $x^* = 0$. The periodic assignment lemma from [21] guarantees that we can modify $\tilde{\Psi}_{ii}(x_i, y_i)$ for the leaf nodes so that the optimal assignment in the unwrapped tree is all zeros

$$P(\tilde{x} = 0|\tilde{y}) = \max_{\tilde{x}} P(\tilde{x}|\tilde{y}) \tag{9}$$

(The $\tilde{\Psi}_{ii}(x_i, y_i)$ are modified to include the messages from the nodes to be added at the next stage of the unwrapping).

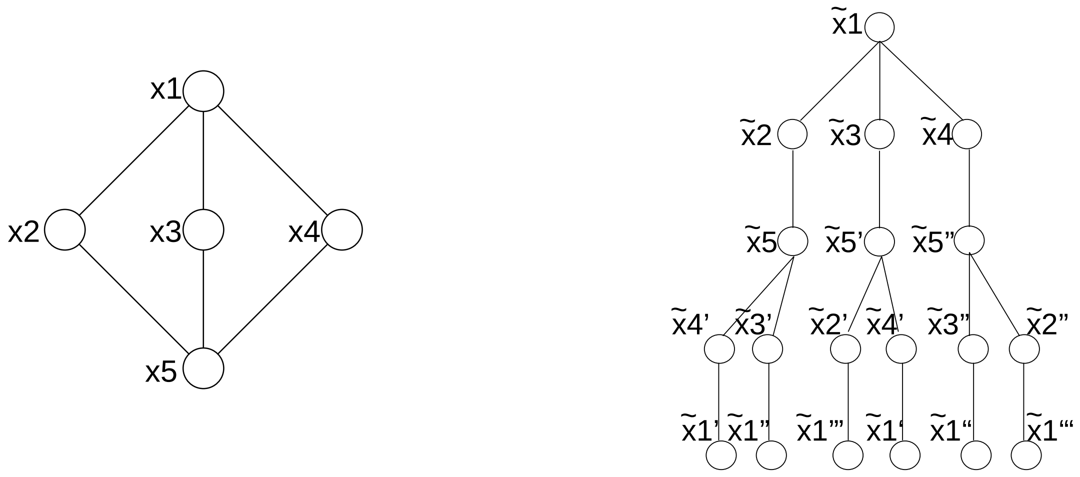


Fig. 3. **a:** A Markov Random Field with multiple loops. **b:** The unwrapped graph corresponding to this structure. The unwrapped graphs are constructed by replicating the potentials $\Psi(x_i, x_j)$ and observations y_i while preserving the local connectivity of the loopy graph. They are constructed so that the messages received by node x_1 after t iterations in the loopy graph are equivalent to those that would be received by x_1 in the unwrapped graph. An observed node, y_i, not shown, is connected to each depicted node.

We now show that the global optimality of $\tilde{x} = 0$ and the method of construction of the unwrapped tree guarantee that $P(x = 0|y) > P(x|y)$ for all x in the SLT neighborhood of 0.

Referring to figure 3a, suppose that $P(x = 10000|y) > P(x = 00000|y)$. By the Markov property of the diamond figure, this means that:

$$P(x_1 = 1|x_{2-4} = 0) > P(x_1 = 0|x_{2-4} = 0) \tag{10}$$

Note that node $\tilde{x}_1$ has exactly the same neighbors in the unwrapped graph as x_1 has in the loopy graph. Furthermore, by the method of construction, the potentials between x_1 and each of its neighbors is the same as the potentials between $\tilde{x}_1$ and its neighbors. Thus equation 10 implies that:

$$P(\tilde{x}_1 = 1|\tilde{x}_{2-4} = 0) > P(\tilde{x}_1 = 0|\tilde{x}_{2-4} = 0) \tag{11}$$

in contradiction to equation 9. Thus no change of a single x_i can improve the posterior probability.

What about changing two x_i at a time? If we change a pair that is not connected in the graph, say x_1 and x_5, then by the Markov property this is equivalent to changing one at a time. Thus suppose $P(10001) > P(0000)$ this again implies that $P(x_1 = 1|x_{2-4} = 0) > P(x_1 = 0|x_{2-4} = 0)$ and we have shown earlier that leads to a contradiction. Thus no change of assignment in two unconnected nodes can improve the posterior probability.

If the two are connected, say x_1, x_2 then the same argument holds with respect to the pair of nodes x_1, x_2. Note that the subgraph $\tilde{x}_{1-2}$ is isomorphic to the subgraph x_{1-2} and the two subgraphs have the same neighbors. Hence:

$$P(x_{1-2} = 1|x_{3-5} = 0) > P(x_{1-2} = 0|x_{3-5} = 0) \tag{12}$$

implies that:

$$P(\tilde{x}_{1-2} = 1|\tilde{x}_{3-5} = 0) > P(\tilde{x}_{1-2} = 0|\tilde{x}_{3-5} = 0) \tag{13}$$

and this is again in contradiction to equation 9. Thus no change of assignment in two connected nodes can improve the posterior probability.

Similar arguments show that no change of assignment in any subtree of the graph can improve the posterior probability (e.g. changing the values of x_{1-4} or changing the values of x_{1-2} and x_{4-5}).

These arguments no longer hold, however, when we change a subset of nodes that form a loopy subgraph of G. For example, the subgraph x_1, x_2, x_3, x_5 is *not* isomorphic to the subgraph $\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{x}_5$. Indeed since the unwrapped tree is a tree, it cannot have a loopy subgraph. Hence we cannot equate the probabilities of the two subgraphs given their neighbors.

If the subset forms a single loop, however, then there exists an arbitrarily long chain in the unwrapped tree that corresponds to the unwrapping of that loop. For example, note the chain $\tilde{x}_1, \tilde{x}_2, \tilde{x}_5, \tilde{x'}_3, \cdots$ in figure 3b. Using a similar argument to that used in proving optimality of max-product in a single loop graph [22], [19], [2], [5] we can show that if we can improve the posterior in the loopy graph by changing the value of x_1, x_2, x_3, x_5 then we can also improve the posterior in the unwrapped graph by changing the values of the arbitrarily long chain. This again leads to a contradiction.

B. Proof of Claim 1:

We denote by G the original graph and by $\tilde{G}$ the unwrapped graph. We use x_i for nodes in the original graph and $\tilde{x}_i$ for nodes in the unwrapped graph. We define a mapping C from nodes in $\tilde{G}$ to nodes in G. This mapping will say for each node in $\tilde{G}$ what is the corresponding node in G: $C(\tilde{x}_1) = x_1$.

The unwrapped tree is therefore a graph $\tilde{G}$ and a correspondence map. We now give the method of constructing both.

Pick an arbitrary node in G, say x_1. Set $C(\tilde{x}_1) = x_1$. Iterate t times.

- Find all leaves of $\tilde{G}$ (start with the root).
- For each leaf $\tilde{x}_i$ find all k nodes in G that neighbor $C(\tilde{x}_i)$.
- Add $k-1$ nodes as children to x_i, corresponding to all neighbors of $C(\tilde{x}_i)$ except $C(\tilde{x}_j)$, where $\tilde{x}_j$ is the parent of $\tilde{x}_i$.

The potential matrices and observations for each node in the unwrapped network are copied from the corresponding nodes in the loopy graph. That is, if $x_j = C(\tilde{x}_i)$ then $\tilde{y}_i = y_j$, and:

$$\tilde{\Psi}(\tilde{x}_i, \tilde{x}_j) = \Psi(C(\tilde{x}_i), C(\tilde{x}_j)) \tag{14}$$

Note that the unwrapped tree around x_i after t_1 iterations is a subtree of the unwrapped tree around x_j after $t_2 > t_1$ iterations. If we let the number of iterations $t \to \infty$ then the unwrapped tree $\tilde{G}$ becomes a well-studied object in topology: the universal covering of G [15]. Roughly speaking, it is a topology that preserves the *local* topology of the graph G but is singly connected. It is precisely this fact, that max-product gives the global optimum on a graph that has the same local topology as G, that makes sure that x^* is a neighborhood maximum of $P(x|y)$.

We now state some properties of $\tilde{G}$.

1. Equal neighbors property: Every non-leaf node in $\tilde{G}$ has the same number of neighbors as the corresponding node in G and the neighbors are in one-to-one correspondence. If $C(\tilde{x}_i) = x_i$ then for each $\tilde{x}_j \in N(\tilde{x}_i)$, $C(\tilde{x}_j) \in N(x_i)$ and for each $x_j \in N(x_i)$ there exists $\tilde{x}_k \in N(\tilde{x}_j)$ such that $C(\tilde{x}_k) = x_j$. This follows directly from the method of constructing $\tilde{G}$.

2. Equal conditional probability property: The probability of a nonleaf node $\tilde{x}_j$ given its neighbors in $\tilde{G}$ is equal to the probability of $C(\tilde{x}_j)$ given its neighbors. Formally, if $C(\tilde{x}_j) = x_i$ then

$$P\left(\tilde{x}_j | N(\tilde{x}_j) = z, y\right) = P\left(x_i | N(x_i) = z, y\right) \tag{15}$$

This follows from the Markov blanket property of MRFs (eq. 5) and the equal neighborhood property.

3. Isomoprhic subtree property: For any subtree $T \subset G$ then for sufficiently large unwrapping count t there exists an isomorphic subtree $\tilde{T} \subset \tilde{G}$. The nodes of the subtrees are in one-to-one correspondence: for each $x_i \in T$ there exists a $\tilde{x}_i \in \tilde{T}$ such that $C(\tilde{x}_i) = x_i$ and for each $\tilde{x}_j \in \tilde{T}$, $C(\tilde{x}_j) \in T$. To prove this we pick the root node of T, x_i as the initial node around which to expand the unwrapped tree. By the method of construction, the unwrapped tree after a number of iterations equal to the depth of T will be isomoprhic to T and in one-to-one correspondence. This gives us $\tilde{T}$. For any other choice of initial node for $\tilde{G}$, the unwrapped tree starting with x_i is a subtree of $\tilde{G}$.

4. Infinite chain property: For any loop $L \subset G$ there exists an arbitrarily long chain $\tilde{L} \subset \tilde{G}$ that is the unwrapping of L. Furthermore, if we denote by n the length of the chain divided by the length of the loop and by $\tilde{x}_1$ and $\tilde{x}_N$ the two endpoints of the chain then:

$$P(\tilde{x}_{\tilde{L}} | N(\tilde{x}_{\tilde{L}})) = P(x_L | N(X_L))^n \beta \tag{16}$$

with β the "boundary potentials"

$$\beta = \prod_{\tilde{x}_i \in N(\tilde{x}_1) \backslash \tilde{x}_2} \tilde{\Psi}(\tilde{x}_i, \tilde{x}_1) \prod_{\tilde{x}_i \in N(\tilde{x}_N) \backslash \tilde{x}_{N-1}} \tilde{\Psi}(\tilde{x}_i, \tilde{x}_N) \tag{17}$$

Note that β is independent of n. x_L and $\tilde{x}_{\tilde{L}}$ refer to all node variables in the sets L and $\tilde{L}$, respectively.

The existence of the arbitrarily long chain follows from the equal neighbor property, while equation 16 follows from the Markov blanket property for MRFs (equation 5).

Another property of the unwrapped tree that we will need was proven in [21]:

5. Periodic assignment lemma: Let m^* be a fixed-point of the max-product algorithm and x^* the max-product assignment in G. Let $\tilde{G}$ be the unwrapped tree. Suppose we modify the observation potentials $\tilde{\Psi}_{ii}$ at the leaf nodes to include the messages from the nodes to be added at the next stage of unwrapping. Then the MAP assignment $\tilde{x}^*$ in $\tilde{G}$ is a replication of x^*: if $x_i = C(\tilde{x}_j)$ then $\tilde{x}^*_j = x^*_i$.

Using these properties, we can prove the main claim. To simplify notation, we again assume that x^*, the assignment based on a fixed-point of the max-product algorithm is equal to zero, $x^* = 0$. The periodic assignment property guarantees that we can modify $\tilde{\Psi}_{ii}(x_i, y_i)$ for the leaf nodes so that the optimal assignment in the unwrapped tree is all zeros

$$P(\tilde{x} = 0 | \tilde{y}) = \max_{\tilde{x}} P(\tilde{x} | \tilde{y}) \tag{18}$$

Now, assume that we can choose a subtree of $T \subset G$ and change the assignment of these nodes x_T to another value and increase the posterior. Again, to simplify notation assume that maximizing value is $x_T = 1$. By the Markov property, this means that:

$$P(x_T = 1|N(x_T) = 0, y) > P(x_T = 0|N(x_T) = 0, y) \tag{19}$$

Now, by the isomorphic subtree property we know that there exists $\tilde{T} \subset \tilde{G}$ that is isomorphic to T. We also know that $\tilde{T}$ has the same conditional probability given its neighbors as T does. Thus equation 19 implies that:

$$P(\tilde{x}_{\tilde{T}} = 1|N(\tilde{x}_{\tilde{T}}) = 0, \tilde{y}) > P(\tilde{x}_{\tilde{T}} = 0|N(\tilde{x}_{\tilde{T}}) = 0, \tilde{y}) \tag{20}$$

in contradiction to equation 18. Hence changing the value of a subtree from the converged values of the max-product algorithm cannot increase the posterior probability.

Now, assume we change the value of a single loop $L \subset G$ and increase the posterior probability. This means that:

$$P(x_L = 1|N(x_L) = 0, y) > P(x_L = 0|N(x_L) = 0, y) \tag{21}$$

By the infinite chain property, we know that for arbitrarily large n:

$$P(\tilde{x}_{\tilde{L}} = 1|N(\tilde{x}_{\tilde{L}}) = 0) = P(x_L = 1|N(x_L) = 0)^n \beta_1 \tag{22}$$

Therefore equation 21 implies that:

$$P(\tilde{x}_{\tilde{L}} = 1|N(\tilde{x}_{\tilde{L}}) = 0) > P(\tilde{x}_{\tilde{L}} = 0|N(\tilde{x}_{\tilde{L}}) = 0) \tag{23}$$

in contradiction to the optimality of $\tilde{x}^* = 0$. Hence changing the value of a single loop cannot improve the posterior probability over x^*.

Now we take two disconnected components $C_1, C_2 \in G$ and assume that changing the values of x_{C_1}, x_{C_2} improves the posterior probability. Again, by the Markov property:

$$\begin{aligned} &P(x_{C_1} = 1, x_{C_2} = 1|N(x_{C_1}, x_{C_2}) = 0, y) > \\ &\quad P(x_{C_1} = 0, x_{C_2} = 0|N(x_{C_1}, x_{C_2}) = 0, y) \end{aligned} \tag{24}$$

but since C_1 and C_2 are not connected this implies:

$$\begin{aligned} &P(x_{C_1} = 1|N(x_{C_1}) = 0, y) P(x_{C_2} = 1|N(x_{C_2}) = 0, y) > \\ &\quad P(x_{C_1} = 0|N(x_{C_1}) = 0, y) P(x_{C_2} = 0|N(x_{C_2}) = 0, y) \end{aligned} \tag{25}$$

Which implies that either

$$P(x_{C_1} = 1|N(x_{C_1} = 0), y) > P(x_{C_1} = 0|N(x_{C_1} = 0), y) \tag{26}$$

or:

$$P(x_{C_2} = 1|N(x_{C_2} = 0), y) > P(x_{C_2} = 0|N(x_{C_2} = 0), y) \tag{27}$$

Thus if C_1 or C_2 are either a tree or a single loop this leads to a contradiction. Hence we cannot simultaneously change the values of two subtrees or of a subtree and a loop or of two disconnected loops and increase the posterior probability. Similarly, we can show that changing the value of any finite number of disconnected trees or single loops will not increase the posterior. This proves claim 1.

III. Examples

Claim 1 holds for arbitrary topologies and arbitrary potentials (both discrete and continuous nodes). We illustrate the implications of claim 1 for specific networks.

A. Gaussian graphical models

A Gaussian graphical model is one in which the joint distribution over x is Gaussian. Weiss and Freeman [21] have analyzed belief propagation on such graphs. One of the results given there can also be proved using our claim 1.

Corollary 1: For a Gaussian graphical model of arbitrary topology. If belief propagation converges, then the posterior marginal means calculated using belief-propagation are exact.

Proof: For Gaussians, max-product and sum-product are identical. The posterior means calculated by belief propagation are therefore identical to the max-product assignment. By claim 1, we know that this must be a neighborhood maximum of the posterior probability. But Gaussians are unimodal hence it must be a global maximum of the posterior probability. Thus the max-product assignment must equal the MAP assignment, and the posterior means calculated using belief propagation are exact.

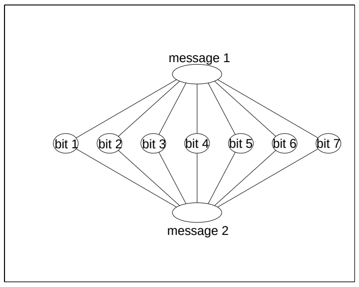


Fig. 4. Turbo code structure

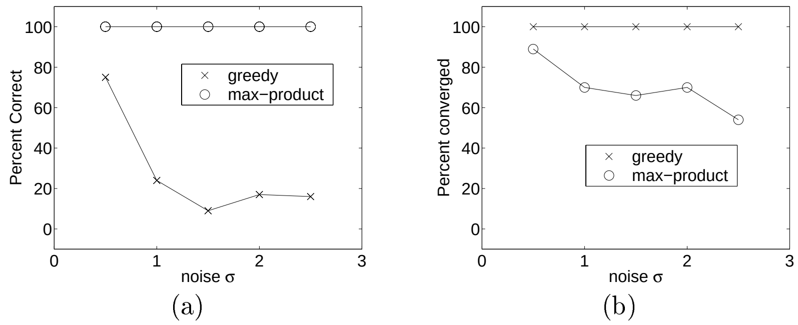


Fig. 5. Results of small turbo-code simulation. (a) Percent correct decodings for max-product algorithm, compared with greedy gradient ascent in the posterior probability. (b) Comparison of convergence results of the two algorithms.

B. Turbo-codes

Figure 4 shows the pairwise Markov network corresponding to the decoding of a turbo code with 7 unknown bits. The top and bottom nodes represent the two transmitted messages (one for each constituent code). Thus in this example, the top and bottom nodes can take on 128 possible values. The potentials between the message nodes and their observations give the posterior probability of the transmitted word given one message, and the potentials between the message nodes and the bit nodes impose consistency. For example $\Psi(\text{bit}_1, \text{message}_1) = 1$ if the first bit of message_1 is equal to bit_1 and zero otherwise. It is easy to show [20] that sum-product belief propagation on this graph gives the turbo decoding algorithm and max-product belief propagation gives the modified turbo decoding algorithm of [3].

Corollary 2: For a turbo code with arbitrary constituent codes. Let x^* be a fixed-point max-product decoding. Then $P(x^*|y) > P(x|y)$ for all x within Hamming distance 2 of x^*.

Proof: This follows from the main claim. Note that whenever we change any bits in the graph we also have to change the two message nodes so changing more than two bits will give two loops.

An obvious consequence of corollary 2 is that for a turbo code with arbitrary constituent codes, the max-product decoding is either the MAP decoding or at least Hamming distance 3 from the MAP decoding. In other words, the max-product algorithm cannot converge to a decoding that is “almost” right: if it is wrong it must be wrong in at least three bits. In order for max-product to converge to a wrong decoding, there must exist a decoding that is at least distance 3 from the MAP decoding, and that decoding must have higher posterior probability than anything in its neighborhood. If no such wrong decoding exists, the max-product algorithm must either converge to the MAP decoding or fail to converge to a fixed-point.

This behavior can be contrasted with the behavior of “greedy” iterative decoding which increases the posterior probability of the decoding at every iteration. Greedy iterative decoding checks all bits and compares the posterior probability with the current value of that bit versus flipping that bit. If the posterior probability improved with flipping, the algorithm flips it (this is equivalent to free energy decoding [14] at zero temperature). This greedy decoding algorithm is guaranteed to converge to a *local* maximum of the posterior probability.

To illustrate these properties we ran the following simulations. We simulated transmitting turbo-encoded codewords of length 7 bits over a Gaussian channel. We compared the max-product decoding and the greedy decoding to the MAP decoding (since we were dealing with such short block lengths we could calculate the MAP decoding using exhaustive search). We varied the noise σ of the Gaussian channel.

Figure 5 shows the results. Figure 5a shows the probability of convergence for both algorithms. Convergence was determined numerically for both algorithms: if the standard deviation of the messages over 10 successive iterations was less than 10^{-5} we declared convergence. If this criterion was not achieved in 100 iterations, we called that run a failure to converge.

Figure 5b shows the probability of a correct decoding (i.e. a decoding equal to the MAP decoding) for the two algorithms on the cases for which both converged. When max-product converges, it always finds the MAP decoding. In contrast, greedy decoding very frequently converges to a wrong decoding.

C. 2D grids

Figure 2 (a) shows a two dimensional grid. For two dimensional grids, it is easy to show the following corollaries:

Corollary 3: For two dimensional grids of arbitrary size and arbitrary potentials. Any configuration is either (1) in the SLT region of the max-product assignment or (2) in the SLT region of an assignment that is in the SLT region of the max-product assignment.

Corollary 4: For two dimensional grids of size n. The size of the SLT neighborhood increases exponentially with n.

Both corollaries follow from the fact that we can change the value of all the even (or odd) rows to an arbitrary value.

We compared greedy decoding to max-product decoding on the $3x3$ grid. Max-product found the MAP decoding in 99% of the runs and when it was wrong, its assignment was always the second best. In comparison, greedy decoding found the MAP assignment only on 44% of the runs and the ranking was anywhere between 4 and 61.

IV. Discussion

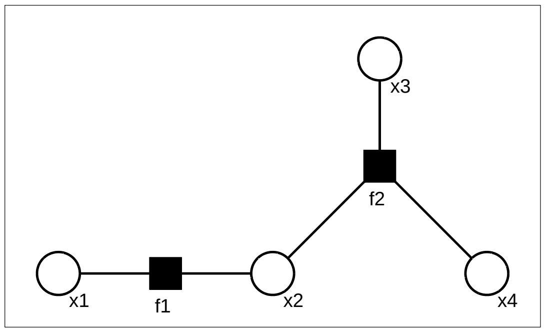
The idea of using the computation tree to prove properties of the max-product assignment was also used in [22], [19], [7], [10]. The main tool in those analyses was the fact that the max-product assignment was the *global* optimum in the unwrapped tree. There are two problems with generalizing this approach to arbitrary topologies.

First, the global optimum depends on the numerosity of node replicas in the unwrapped tree. That is, different nodes in G may have different number of replicas in $\tilde{G}$. This leads to a distinction between balanced (or nonskewed) graphs [19], [7] and unbalanced (or skewed) graphs. Balanced graphs are those for which all nodes in G have asymptotically the same number of replicas in $\tilde{G}$. For unbalanced graphs, it is much more difficult to relate global optimality in $\tilde{G}$ to optimality in G.

Second, the global optimum in the unwrapped tree contains contributions from the interior nodes (that have exactly the same neighbors in $\tilde{G}$ as do their corresponding nodes in G) and contributions from the leaf nodes (that are missing some of the neighbors in $\tilde{G}$). Unfortunately, for most graphs G the number of leaf nodes grows at the same rate as the non-leaf nodes and cannot be neglected from the analysis.

In this analysis, on the other hand, we used primarily the *local* properties of the computation tree. No matter what the topology of G is, it is always the case that the local structure of $\tilde{G}$ is the same as the local structure of G. Thus the numerosity of the nodes in $\tilde{G}$ and the ratio of leaf nodes to non-leaf nodes is irrelevant. In this way, we can analyze the max-product assignment in arbitrary topologies.

Although we exploited the local properties, we would like to extend our analysis using the global properties as well. Our simulation results indicate that the max-product assignment is better than our analytical results guarantee. For example, in the turbo code simulations we found that the posterior probability often contained two SLT maxima but for all these cases, max-product found the global maximum (and not the second SLT maximum). In current work, we are looking into using the global properties of the computation tree to extend our analysis.



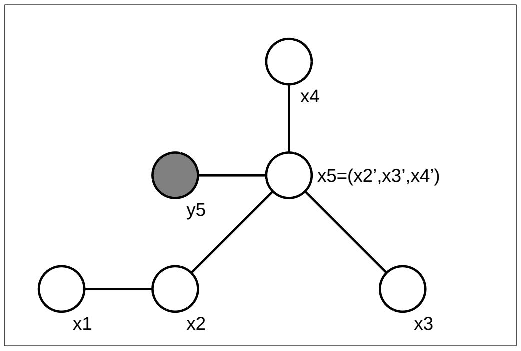


Fig. 6. Any factor graph can be converted into a Markov random field with pairwise potentials that represents exactly the same probability distribution over variables. When this conversion is done, the belief propagation algorithm for the pairwise Markov graph is equivalent to the belief propagation algorithm on the factor graph.

Appendix

Appendix — relationship between belief propagation schemes

A. Converting a factor graph to a pairwise Markov graph

A factor graph [7] is a bipartite graph with *function* nodes f_i denoted by filled squares and *variable* nodes x_i denoted by unfilled circles. The function nodes denote a decomposition of a "global" function $g(x)$ into a product of "local" functions $f_i(x)$. We will assume that g represents a joint distribution over the variable nodes. The method of converting a factor graph into a pairwise Markov graph is to remove the function nodes. Specifically:

- find all f_i of degree 2. For each such f_i remove it from the graph and directly connect the two variables nodes that were connected to those function nodes. That is, if f_1 is a degree 2 node connected to x_1, x_2 we directly connect x_1 and x_2 and set $\Psi(x_1, x_2) = f(x_1, x_2)$.
- for all f_i of degree $d > 2$, replace the node f_i with a new variable node x_{N+i}. The variable node x_{N+i} represents the joint configuration of all the x_i that were connected to f_i. That is if f_2 is connected to x_2, x_3, x_4 then the new variable x_6 is a vector with three components $x_5 = (x'_2, x'_3, x'_4)$. Set the potentials $\Psi(x_{N+i}, x_j) = \delta(x_j - x'_j)$. Finally, add an observation node y_{N+i} and set $\Psi(x_{N+i}, y_{N+i}) = f_i(x)$.

It is easy to show that (1) The joint distribution over the variables $x_1 \cdots x_N$ in the pairwise Markov graph is exactly $f(x)$ and (2) the belief propagation algorithm in the pairwise Markov graph is equivalent to the belief propagation algorithm in [7].

B. Converting a junction graph to a pairwise Markov graph

A junction graph [1] is a graph in which vertices s_i represent "local domains" of a global function. Thus if $g(x_1, x_2, x_3)$ factorizes so that:

$$g(x_1, x_2, x_3) = f_1(x_1, x_2) f_2(x_2, x_3) \tag{28}$$

then the two local domains are $s_1 = (x_1, x_2)$ and $s_2 = (x_2, x_3)$. Edges between these vertices correspond to "communication links" in a message passing scheme for calculating marginals of g.

Aji et al. showed that for such a message passing algorithm to exist, the junction graph must possess the "running intersection property" — the subset of nodes whose domains include x_i together with the edges containing these nodes must form a connected graph. We now show that junction graphs are equivalent to pairwise Markov graphs.

To show this we leave the graph between s_i unchanged and add "observation" nodes y_i such that $\Psi(s_i, y_i) = f(s_i)$. We set $\Psi(s_i, s_j) = 1$ if the two nodes agree on the value of any x_i that exists in both domains and zero otherwise. Note that the running intersection property guarantees that any two nodes (not necessarily neighboring) must agree on the value of a common x_i for the joint distribution to be nonzero. When the potentials are set in this way, it is easy to see that the joint distribution over x in the pairwise Markov graph is exactly $g(x)$ and that the belief propagation algorithm in the Markov graph is equivalent to the GDL algorithm in [1].

References

[1] S. M. Aji and R.J. McEliece. The generalized distributive law. IEEE Transactions on Information Theory, 2000. in press.

[2] S.M. Aji, G.B. Horn, and R.J. McEliece. On the convergence of iterative decoding on graphs with a single cycle. In *Proc. 1998 ISIT*, 1998.

[3] 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.

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

[5] G.D. Forney, F.R. Kschischang, and B. Marcus. Iterative decoding of tail-biting trellisses. preprint presented at 1998 Information Theory Workshop in San Diego, 1998.

[6] W.T. Freeman and E.C. Pasztor. Learning to estimate scenes from images. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Adv. Neural Information Processing Systems 11.* MIT Press, 1999.

[7] B J. Frey, R. Koetter, and A. Vardy. Skewness and pseudocodewords in iterative decoding. In *Proc. 1998 ISIT*, 1998.

[8] R.G. Gallager. *Low Density Parity Check Codes.* MIT Press, 1963.

[9] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. PAMI*, 6(6):721–741, November 1984.

[10] G.B. Horn. *Iterative decoding and pseudocodewords.* PhD thesis, Department of Electrical Engineering, California Institute of Technology, Pasadena, CA, May 1999.

[11] F.V. Jensen. *An Introduction to Bayesian Networks.* Springer, 1996.

[12] F. R. Kschischang, B. J. Frey, and H.A. Loeliger. factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 1998. submitted.

[13] S. Lauritzen. *Graphical Models.* Oxford University Press, 1996.

[14] D.J.C. Mackay and Radford M. Neal. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 1999.

[15] J.R. Munkres. *Topology: a first course.* Prentice Hall, 1975.

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

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

[18] S. E. Shimony. Finding the maps for belief networks is NP-hard. *Artificial Intelligence*, 68(2):399–410, 1994.

[19] Y. Weiss. Belief propagation and revision in networks with loops. Technical Report 1616, MIT AI lab, 1997.

[20] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, to appear, 2000.

[21] Y. Weiss and W. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. In S. Solla, T. K. Lean, and K. R. Muller, editors, *Advances in Neural Information Processing Systems 12*, 2000.

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