Appeared as MIT-LIDS technical report P-2554
Laboratory for Information and Decision Systems, MIT

Tree consistency and bounds on the performance of the max-product algorithm and its generalizations

Martin Wainwright* martinw@eecs.berkeley.edu
Tommi Jaakkola† tommi@ai.mit.edu
Alan Willsky† willsky@mit.edu

Electrical Engineering & CS*
UC Berkeley, Berkeley, CA

Electrical Engineering & CS†
MIT, Cambridge, MA

October 28, 2002

Abstract

Finding the *maximum a posteriori* (MAP) assignment of a discrete-state distribution specified by a graphical model requires solving an integer program. The max-product algorithm, also known as the max-plus or min-sum algorithm, is an iterative method for (approximately) solving such a problem on graphs with cycles. We provide a novel perspective on the algorithm, which is based on the idea of reparameterizing the distribution in terms of so-called *pseudo-max-marginals* on nodes and edges of the graph. This viewpoint provides conceptual insight into the max-product algorithm in application to graphs with cycles. First, we prove the existence of max-product fixed points for positive distributions on arbitrary graphs. Next, we show that the approximate max-marginals computed by max-product are guaranteed to be consistent, in a suitable sense to be defined, over every tree of the graph. We then turn to characterizing the nature of the approximation to the MAP assignment computed by max-product. We generalize previous work by showing that for any graph, the max-product assignment satisfies a particular optimality condition with respect to any subgraph containing at most one cycle per connected component. We use this optimality condition to derive upper bounds on the difference between the log probability of the true MAP assignment, and the log probability of a max-product assignment. Finally, we consider extensions of the max-product algorithm that operate over higher-order cliques, and show how our reparameterization analysis extends in a natural manner.

Keywords: MAP estimation; integer programming; max-product; min-sum; max-plus; graphical models; max-marginals; inference; iterative decoding; Viterbi algorithm; dynamic programming

1 Introduction

Integer programming problems are important in a variety of fields, including artificial intelligence and statistics (e.g., Pearl, 1988; Dawid, 1992; Cowell et al., 1999), statistical physics (e.g., Barahona, 1982), error-correcting coding (e.g., Aji et al., 1998; Gallager, 1963), and statistical image processing (e.g., Besag, 1986; Geman and Geman, 1984). Such problems, which entail optimizing a cost function over some subset of the integers, can in many cases be formulated in terms of *graphical models* (e.g., Cowell et al., 1999; Jordan, 1999). In the formalism of such models, the cost function to be maximized corresponds to a distribution that factorizes as a product of terms over cliques of the graph, and the associated optimization problem is to find the *maximum a posteriori* (MAP) configuration.

The complexity of this MAP estimation problem turns out to depend critically on the structure of the underlying graph. On one hand, for a graph without cycles (i.e., a tree), the MAP configuration can be computed efficiently by algorithms in which nodes convey information to one another by passing messages. Updates of this type are known under various names, including belief revision (Pearl, 1988), as well as the max-product or min-sum algorithm (e.g., Aji and McEliece, 2000). One way to view max-product updates is as a parallel implementation of standard dynamic programming updates (Bertsekas, 1995); in this sense, they represent a generalization of the Viterbi algorithm (Viterbi, 1967; Forney, 1973) from a chain-structured graph to an arbitrary tree. On such tree-structured graphs, it is well-known that the message-passing updates converge, and yield the correct MAP assignment after a finite number of steps (e.g. Pearl, 1988).

Dawid (1992) described how to compute the MAP configuration for more general graphs by first converting to a junction tree representation, and then performing dynamic programming in the junction tree. Nilsson (1998)

Work supported in part by ODDR&E MURI Grant DAAD19-00-1-0466 through the Army Research Office; by the Office of Naval Research through Grant N00014-00-1-0089; and by the Air Force Office of Scientific Research through Grant F49620-00-1-0362.

analyzed this algorithm in more detail, and showed how a partitioning scheme can be used to compute the M most probable configurations. The complexity of such exact methods scales exponentially in the size of the largest clique in the junction tree, a quantity closely related to the graph *treewidth* (see Bodlaender (1993)). Unfortunately, for many graphs with cycles, this treewidth becomes so large that the junction tree approach is no longer feasible. This intractability motivates the development and use of approximate methods for computing MAP assignments on graphs with cycles.

The focus of this paper is a "loopy" version of the max-product algorithm, in which the usual message-passing updates are applied in parallel to a graph with cycles. This use of the max-product algorithm for computing approximate MAP assignments on graphs with cycles has been the focus of considerable research in recent years (e.g., Aji et al., 1998; Horn, 1999; Forney et al., 2001; Frey et al., 1998; Weiss, 2000; Freeman and Weiss, 2001). As a consequence of the presence of cycles, the max-product updates are no longer guaranteed to converge. Nonetheless, the loopy max-product algorithm has been used with good results in various applications, including decoding of graphical codes (e.g., Benedetto et al., 1996; Aji et al., 1998; Wiberg, 1996) as well as computer vision (e.g., Freeman and Pasztor, 1999). Most previous analysis of the max-product algorithm for graphs with cycles is based on an idea that dates back to the work of Gallager (1963) — namely, that of studying the *computation tree* associated with the parallel message-passing updates. Given some node specified as the root, the corresponding computation tree (at iteration n) represents the set of all possible paths (of length n) that end at this root node; any such path corresponds to the propagation of information in a message from some initial node to the root. More details on such computation trees can be found in (e.g., Gallager, 1963; Wiberg, 1996; Frey et al., 1998; Weiss, 2000; Freeman and Weiss, 2001). A number of researchers have studied max-product algorithm in application to a single cycle (e.g., Weiss, 2000; Horn, 1999; Aji et al., 1998; Frey et al., 1998; Forney et al., 2001), for which the computation tree is especially simple — namely, a chain-structured graph. For a single cycle, the message passing updates converge to either a stable fixed point, or a periodic oscillation. When it converges to stable fixed point, the max-product assignment is guaranteed to be exact, as long as a certain uniqueness condition (to be specified) holds. Freeman and Weiss (2001) analyzed the max-product assignment for positive compatibility functions on arbitrary graphs, and showed that the cost of the max-product assignment cannot be improved by changing any subset of the variables that form no more than a single cycle in the underlying graph. In the context of graphs defined by codes, Frey and Koetter (2000) showed that if max-product messages are suitably attenuated as they are passed up the computation tree, then the algorithm (if it converges) computes the MAP assignment. For the special case of cycle codes,[1] Wiberg (1996) gave sufficient conditions for max-product to fail to converge to the transmitted codeword; see also Horn (1999) for related results on cycle codes. For the ferromagnetic Ising model,[2] Fridman (2002) showed how to assess the correctness of the max-product assignment in an *a priori* manner (i.e., before even running the algorithm).

The analysis of this paper, in contrast to previous work using computation trees, is based on a different perspective. Any distribution defined by a graphical model is represented as a product of so-called compatibility functions over cliques of the underlying graph. Since the factorization is not unique, it is tempting to seek alternative representations that have desirable properties for solving inference problems. In previous work (Wainwright et al., 2002), we have shown that the sum-product or belief propagation algorithm for graphs with cycles can be viewed as seeking a particular *reparameterization* of the distribution. In this paper, we show that the conceptual framework of reparameterization can be applied fruitfully to the max-product algorithm on graphs with cycles as well.

More specifically, the complexity of the MAP estimation problem stems from the specification of the overall distribution as a (normalized) product of compatibility functions on the cliques of the graph. These functions couple together the optimization of variables at all the nodes. However, the factorization of the overall distribution is not unique, which suggests the idea of finding a new factorization — that is, *reparameterizing* the distribution — in terms of functions that reduce or remove this coupling. For a distribution on a decomposable graph, a variant of the junction tree theorem (Dawid, 1992; Cowell et al., 1999) yields a factorization in terms of so-called *max-marginals* on the maximal cliques and separator sets of the junction tree. As we elucidate in Section 3 for trees, the message-passing of the max-product algorithm can be understood as computing this max-marginal factorization.

At one level, then, solving the MAP estimation problem on a decomposable graph can be viewed as converting from the original representation into a form that makes extracting the MAP estimate computationally simple.

[1] Cycle codes correspond to low-density parity check codes (Gallager, 1963) for which each bit is connected to exactly two parity checks.

[2] This model corresponds to a collection of binary variables with attractive couplings between adjacent nodes.

Indeed, if each single-node max-marginal has a unique maximum, then the overall MAP estimate can be computed by performing *decoupled* maximizations of the max-marginals at each node. As we discuss in Section 3, the violation of this assumption is only a minor issue for tree-structured distributions. In fact, since the max-product algorithm also computes pairwise max-marginals, dynamic programming methods can be applied to the tree in order to compute one of the MAP estimates. (See also Dawid (1992) for description of a similar sampling procedure for junction trees). However, as we illustrate by example in Section 4, when the uniqueness assumption is violated for a graph with cycles, the max-marginals are at best problematic, and at worst very misleading. Thus, a minor contribution of our paper is to identify the importance of the uniqueness assumption both for our analysis, and for the utility of max-product in application to graphs with cycles.

The reparameterization perspective motivates the idea of considering a sequence of updates on trees embedded within a graph with cycles. Each such update consists of reparameterizing the subset of factors defining the distribution that correspond to (nodes and edges in) the tree, while leaving untouched the remaining factors. More specifically, the reparameterization for the tree terms is specified by what we refer to as a set of *pseudo-max-marginals*. The main results of this paper, which are presented in Section 4, are based on a detailed analysis of such sequences of reparameterization updates. In this section, we prove that fixed points of such updates are equivalent to those of the usual message-passing updates. This equivalence allows us to analyze max-product in terms of reparameterization as opposed to the usual message-passing updates. For a graph with cycles, it is not obvious that fixed points of the max-product algorithm necessarily exist. A number of researchers (e.g., Aji et al., 1998; Weiss, 2000) have shown the existence of fixed points for positive compatibility functions on a single cycle graph. We generalize this result by proving that max-product fixed points exist for any positive distribution on an arbitrary graph with cycles. An interesting implication is that any positive distribution can be factorized in terms of a set of pseudo-max-marginals that are consistent (in a sense that we make precise) on every spanning tree of the graph. We also show in Section 4 that the uniqueness of the single-node maximizers is critical if the pseudo-max-marginals are to be of any value for MAP estimation. Furthermore, under the assumption of uniqueness, we prove that the max-product assignment on a graph G with cycles is guaranteed to be "optimal" on *any* subgraph of G containing at most one cycle, where "optimality" here refers to the inability to obtain a better assignment for a modified cost function formed by the product of terms corresponding to the subgraph. This theorem generalizes earlier results by Weiss (2000), and Freeman and Weiss (2001) mentioned previously. Using this optimality theorem, we then derive a set of computable upper bounds on the difference between the log probability of the MAP assignment, and the log probability of the max-product assignment on an arbitrary graph with cycles, and we illustrate the consequences of these bounds. In Section 5 and in analogy to recent generalizations of the sum-product algorithm (e.g., Yedidia et al., 2001), we consider generalizations of the max-product algorithm that operate over higher order cliques. An attractive feature of our reparameterization analysis is that it extends very naturally to these generalizations.

2 Background

In this section, we first provide some basic background on graph theory, giving only those concepts necessary for subsequent developments; we refer the reader to the books by Bollobás (1998) and Berge (1976) for further background. We then discuss the formalism of graphical models — in particular, Markov random fields — as well as the associated MAP estimation problem. More details about graphical models can be found in various sources (e.g., Cowell et al., 1999; Lauritzen, 1996; Jordan, 1999).

2.1 Graph-theoretic basics

An undirected graph $G = (V, E)$ consists of a set of nodes or vertices $V = \{1, \ldots, N\}$ that are joined by a set of edges E. Throughout this paper, we focus on *simple* graphs, for which multiple edges between the same pair of vertices as well as self-loops (i.e., an edge from a node back to itself) are forbidden. For each $s \in V$, let $\Gamma(s) = \{\, t \in V \mid (s,t) \in E \,\}$ denote the set of *neighbors* of s. A *clique* of the graph G is any subset of the vertex set V for which each node is connected to every other. A clique is *maximal* if it is not properly contained within any other clique. Note that any single node is itself a clique, but not a maximal clique unless it has no neighbors.

A subgraph $H = (V(H), E(H))$ of the graph G is a graph formed by subsets $V(H)$ and $E(H)$ of the vertex and edge sets (V and E respectively). A *spanning subgraph* is one that includes every vertex of the graph (i.e., $V(H) = V$). In the sequel, it will be important to consider *node-induced subgraphs*. In particular, given a subset $S \subset V$, the corresponding node-induced subgraph $H[S]$ is formed by the vertex set $V(S) = S$

and the edge set $E(S) = \{ (s,t) \in E \mid s,t \in S \}$. Figure 1 provides an illustration of a node-induced subgraph. A *path* is a graph P consisting of a set of vertices $V(P) = \{s_0, s_1, \ldots, s_k\}$ and a collection of distinct edges

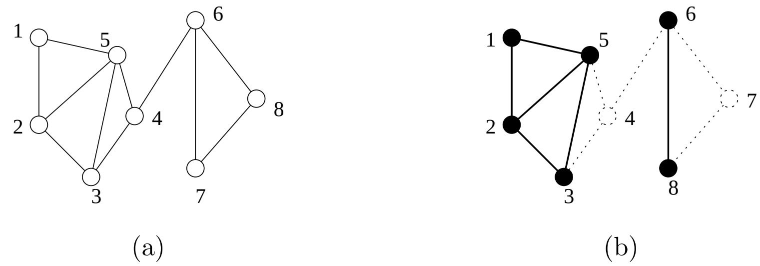


Figure 1. Illustration of a node-induced subgraph. (a) Original graph G. (b) Subgraph $H[S]$ induced by the subset of nodes $S = \{1, 2, 3, 5, 6, 8\}$.

$E(P) = \{(s_0, s_1), \ldots (s_{k-1}, s_k) \}$. We say that P is a path from s_0 to s_k. A graph is *connected* if for each pair $\{s, t\}$ of distinct vertices, there is a path from s to t. A *component* of a graph is a maximal connected subgraph.

Graphs without cycles play an important role in our analysis. A *tree* $\mathcal{T}$ is a cycle-free graph consisting of a single connected component; a *spanning tree* is an acyclic subgraph whose vertex set is all of V. See Figure 2 for an illustration of a spanning tree. A *leaf node* of a tree is a vertex with a single neighbor; any tree with $N \geq 2$ nodes is guaranteed to have at least two leaves (Bollobás, 1998).

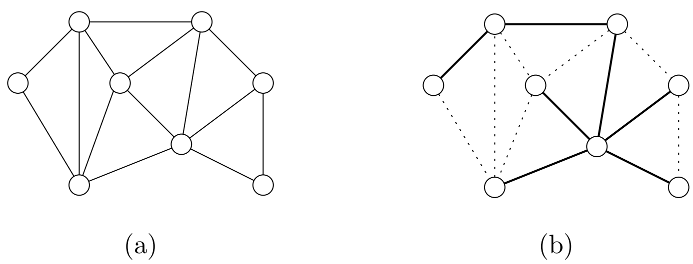


Figure 2: (a) Graph G with cycles. (b) Embedded spanning tree that reaches each vertex of G.

2.2 Markov random fields and MAP estimation

Given an undirected graph G, we associate with each node s in the vertex set a discrete random variable x_s taking values in the discrete space $\mathcal{X} = \{0, \ldots, m-1\}$. The full vector $\mathbf{x} = \{ x_s \mid s \in V\}$ takes values in the Cartesian product space $\mathcal{X}^N$. Of interest to us are random vectors $\mathbf{x}$ that respect a set of *Markov properties* associated with the graph. In particular, let A, B, and C be three disjoint subsets of the vertex set, and suppose that B separates A from C (as illustrated in Figure 3). We then require that the collection of random variables

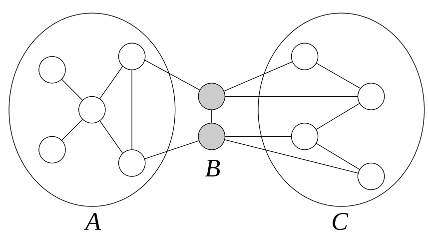


Figure 3. Illustration of graph separation. With all paths that pass through subset B blocked, the two sets A and C are separated. Thus, B forms a vertex cutset for the graph.

in A (i.e., $\mathbf{x}_A = \{x_s \mid s \in A\}$) is conditionally independent of those in C (i.e., $\mathbf{x}_C$), given the variables $\mathbf{x}_B$. The random vector $\mathbf{x}$ is a *Markov random field* if it respects all such properties associated with G.

Given a clique $\mathcal{C}$ of the graph G, a *compatibility function* is a mapping $\psi_{\mathcal{C}} : \mathcal{X}^N \to \mathbb{R}^+$ that depends only on the subvector $\mathbf{x}_{\mathcal{C}} = \{ x_s \mid s \in \mathcal{C} \}$. The Hammersley-Clifford theorem (e.g., Brémaud, 1991) guarantees that a random vector $\mathbf{x}$ with strictly positive distribution $p(\mathbf{x})$ is a Markov random field (MRF) with respect to G if and only if the distribution factorizes as a product of compatibility functions over cliques:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{\mathcal{C} \in \mathbf{C}} \psi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}}) \tag{1}$$

Here $\mathbf{C}$ is the set of all cliques of G, and $Z = \sum_{\mathbf{x}} \prod_{\mathcal{C} \in \mathbf{C}} \psi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}})$ is the partition function that normalizes the distribution.

As is standard in most treatments of the sum-product or max-product algorithms (e.g., Pearl, 1988; Weiss, 2000), the bulk of this paper treats pairwise Markov random fields, for which G has only singleton and pairwise cliques. In Section 5, we show how the ordinary max-product algorithm can be extended to accommodate higher-order cliques, and how our analysis applies to these generalizations as well. For a pairwise Markov random field, the set of cliques is given by $V \cup E$, so that we have a compatibility function ψ_s for each node, and a function ψ_{st} for each edge $(s,t) \in E$. In this case, equation (1) assumes the following simpler form:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{s \in V} \psi_s(x_s) \prod_{(s,t) \in E} \psi_{st}(x_s, x_t) \tag{2}$$

The effect of including independent noisy observation y_s of x_s at each node is to modify the singleton compatibility functions. Since the addition of observations can be modeled by equation (2), we suppress any explicit reference to measurements throughout this paper.

The problem of *maximum a posteriori* (MAP) estimation corresponds to finding the configuration that is most likely under the distribution $p(\mathbf{x})$ defined in equation (1). In more formal terms, the MAP configuration $\widehat{\mathbf{x}}_{MAP} = \{\widehat{x}_s \mid s \in V\}$ is given by

$$\widehat{\mathbf{x}}_{MAP} \triangleq \arg \max_{\mathbf{x} \in \mathcal{X}^N} p(\mathbf{x}). \tag{3}$$

Note that the MAP configuration need not be unique; that is, there may be multiple configurations that attain the maximum in equation (3). In this case, we refer to each one as a possible MAP configuration.

Equivalently, we can formulate the MAP problem as finding the configuration $\mathbf{x} \in \mathcal{X}^N$ that maximizes the function:

$$J(\mathbf{x}; G) \triangleq \sum_{s \in V} \log \psi_s(x_s) + \sum_{(s,t) \in E} \log \psi_{st}(x_s, x_t) \tag{4}$$

Note that any unconstrained integer programming problem in which variables interact in a pairwise manner can be expressed in the form of equation (4). Examples include the minimum $s-t$ cut and multiway-cut problems in combinatorial optimization (e.g., Grötschel et al., 1993), image segmentation on a grid (e.g., Geman and Geman, 1984), and calculating ground states in the Ising model of statistical physics (e.g., Barahona, 1982).

3 MAP estimation on trees

The focus of this section is the problem of MAP estimation for tree-structured graphs, in which context we introduce max-marginals and the max-product algorithm. This section is, in part, continued background, but it also provides several important observations that are critical for our main development in Section 4. In Subsection 3.1, we review the factorization of any tree-structured distribution in terms of its max-marginals, as guaranteed by the junction tree theorem (e.g., Cowell et al., 1999; Dawid, 1992). We also show that the use of the max-marginals for MAP estimation requires some care if the the single-node maximizers fail to be unique. In Subsection 3.2, we introduce the max-product algorithm in the context of tree-structured distributions, and prove that this algorithm exactly computes the reparameterization of the overall distribution in terms of max-marginals. This interpretation is key to our analysis in Section 4 of max-product on graphs with cycles, in which case the quantities computed by the max-product algorithm represent only approximations to the true max-marginals.

3.1 Role of max-marginals

We begin by defining the max-marginals associated with any distribution $p(\mathbf{x})$. For each $x_s \in \mathcal{X}$, the value of the max-marginal $P_s(x_s)$ associated with node $s \in V$ is given by maximizing $p(\mathbf{x})$ over the subset of configurations

$\{ \mathbf{x}' \in \mathcal{X}^N \mid x'_s = x_s \}$ — viz.:

$$P_s(x_s) \quad = \quad \kappa \max_{\{\mathbf{x}' \mid x'_s = x_s\}} p(\mathbf{x}') \tag{5}$$

Here κ denotes a positive but otherwise arbitrary normalization constant; for example, it can be chosen so that $\max_{x'_s} P_s(x'_s) = 1$. We use this notation throughout the paper, where the value of κ may differ from node to node, and from line to line.

Note that $P_s(x_s)$ is proportional to the probability of the most likely configuration $\mathbf{x}'$ with x'_s fixed to the value of x_s. In an analogous fashion, we can define pairwise max-marginals for pairs of nodes $(s,t) \in E$:

$$P_{st}(x_s, x_t) \quad \triangleq \quad \kappa \max_{\{\mathbf{x}' \mid (x'_s, x'_t) = (x_s, x_t)\}} p(\mathbf{x}') \tag{6}$$

Again, $P_{st}(x_s, x_t)$ is proportional to the probability of the most likely configuration $\mathbf{x}' \in \mathcal{X}$ subject to the constraint that $x'_s = x_s$ and $x'_t = x_t$.

The original representation of $p(\mathbf{x})$ given in equation (2) is in terms of compatibility functions on the nodes and edges of the graph. One remarkable property of any tree-structured distribution is that it can also be factorized in terms of its max-marginals:

Theorem 1 (Tree factorization). *Any tree-structured distribution $p(\mathbf{x})$ has the following alternative factorization in terms of its max-marginals:*

$$p(\mathbf{x}) \quad \propto \quad \prod_{s \in V} P_s(x_s) \prod_{(s,t) \in E} \frac{P_{st}(x_s, x_t)}{P_s(x_s) P_t(x_t)} \tag{7}$$

where the max-marginals $\mathbf{P} = \{P_s, P_{st}\}$ are defined in equations (5) *and* (6) *respectively.*

Equation (7) is a special case of the more general junction tree representation (Cowell et al., 1999; Dawid, 1992). In Section 3.2, we provide, as a by-product of our presentation of the max-product algorithm, a constructive proof of the factorization in Theorem 1.

For the time being, suppose that we have computed the complete set of max-marginals $\mathbf{P} = \{P_s, P_{st}\}$. We now consider the problem of using the max-marginals to obtain an MAP configuration. If, for each node, the maximum of P_s over $x'_s \in \mathcal{X}$ is attained at a unique value, then it can be seen that the MAP configuration $\widehat{\mathbf{x}}_{MAP} = \{ \widehat{x}_s \mid s \in V\}$ is unique, with elements given by $\widehat{x}_s = \arg\max_{x'_s \in \mathcal{X}} P_s(x'_s)$. Note the benefit of converting $p(\mathbf{x})$ to the alternative max-marginal representation: instead of globally optimizing the distribution in equation (2), a local optimization at each node — i.e., maximizing $P_s(x_s)$ — suffices to obtain the MAP configuration.

Suppose, on the other hand, that for at least one node $s \in V$, there exist multiple states x_s that attain the maximum of P_s. In this case, there must be multiple MAP configurations (at least one for each state x_s that achieves the maximum); moreover, the single node max-marginals P_s, on their own, do *not* contain enough information to determine a valid MAP assignment. The problem is illustrated by the following example:

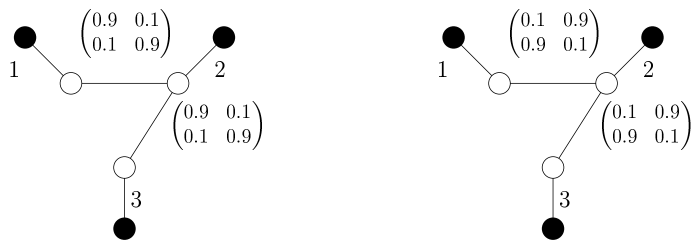
Example 1 (Insufficiency of P_s). Consider a binary random vector $\mathbf{x} \in \{0,1\}^3$ on a Markov chain with three nodes, as illustrated in Figure 4. Suppose that the distribution $p(\mathbf{x})$ is formed as in equation (2) by the following compatibility functions:[3]

$$\psi_s(x_s) \quad = \quad [\,1 \quad 1\,]' \qquad \text{for all } s = 1,2,3 \in V \tag{8a}$$

$$\psi_{st}(x_s, x_t) \quad = \quad \begin{pmatrix} \beta & (1-\beta) \\ (1-\beta) & \beta \end{pmatrix} \qquad \text{for all } (s,t) \in E = \{ (1,2), \ (2,3) \} \tag{8b}$$

Here $\beta \in (0,1)$ is a parameter to be specified. Panels (a) and (b) of Figure 4 illustrate the cases $\beta = 0.9$ and $\beta = 0.1$ respectively. Note the symmetry inherent to this problem: if a configuration $\mathbf{x} \in \{0,1\}^3$ achieves the maximum, then so will $\mathbf{x} \oplus [1 \quad 1 \quad 1]$, where $\oplus$ denotes component-wise addition in mod two arithmetic. More

[3]Here we adopt a notational shorthand for describing compatibility functions. In particular, the function $\psi_s(x_s)$ consists of m numbers $\{\psi_s(j) \mid j \in \mathcal{X}\}$. For a binary process, this collection can be represented as a 2-vector, as in equation (8a). Similarly, the function $\psi_{st}(x_s, x_t)$ consists of m^2 numbers $\{\psi_{st}(j,k) \mid j,k \in \mathcal{X}\}$, which we collect into a $m \times m$ matrix. For a binary process, this matrix is 2×2, as in equation (8b).

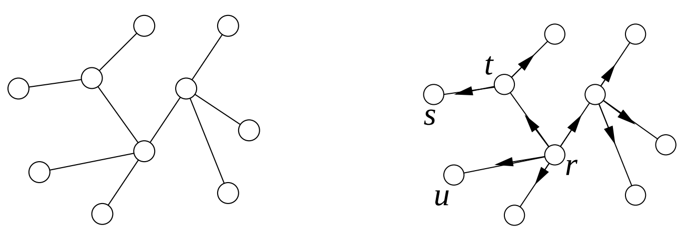


(a) $\beta = 0.9$ (b) $\beta = 0.1$

Figure 4. Illustration of symmetric problems on a simple tree, for which multiple configurations attain the MAP value. In this case, the single node max-marginals $\{P_s\}$ do not provide sufficient information to determine a MAP configuration. Nonetheless, for a tree, a recursive procedure can be used to sample the set of possible MAP assignments.

specifically, if $\beta > 0.5$ (as in panel (a)), then it can be seen that the MAP value is achieved by both $[1\ \ 1\ \ 1]$ and $[0\ \ 0\ \ 0]$. Conversely, if $\beta < 0.5$ (as in panel (b)), then both $[1\ \ 0\ \ 1]$ and $[0\ \ 1\ \ 0]$ are MAP configurations.

For any $\beta \in (0, 1)$, suppose that we use the compatibility functions to define another set of functions via $P_s(x_s) = \psi_s(x_s)$ and $P_{st}(x_s, x_t) = \psi_{st}(x_s, x_t)\psi_s(x_s)\psi_t(x_t)$. As can be seen by direct computation, the set of $\mathbf{P} = \{P_s, P_{st}\}$ defined this way are equivalent, up to normalization constants, to the exact max-marginals of $p(\mathbf{x})$. Since $P_s(x_s) = [1 \quad 1]'$ for each $s \in V$ and for all $\beta \in (0, 1)$, each individual max-marginal indicates that either choice of x_s (0 or 1) is equally good independent of the other variables, and the value of β. However, as we have seen the true optimal MAP assignments depend heavily on the choice of β. A random independent sampling from $\{0, 1\}$ (as implied by the single node max-marginals) will always yield an overall optimal estimate only if $\beta = 0.5$. $\square$



(a) (b)

Figure 5. (a) A tree-structured graph (undirected). (b) A directed version of the same tree, where node r is designated as the root. With the exception of the root, each node has a unique parent; for example, node t is the parent of node s. Nodes s and u are leaf nodes.

Therefore, if there are multiple values of node variables that maximize one or more of the max-marginals, then the single node max-marginals P_s do not provide unambiguous information for MAP estimation. However, the fact that we are dealing with a tree-structured distribution permits the application of ideas from dynamic programming (Bertsekas, 1995). As described in Dawid (1992), similar ideas apply more generally to junction trees.

Using the full set of single *and* pairwise max-marginals, it is possible to draw samples from the set of all MAP assignments. We begin by designating an arbitrary node $r \in V$ as the root of the tree, and then assign directions to the edges of the tree, running from the root to the leaves, so as to ensure that each node (except the root) has a unique parent. This procedure of assigning directions to edges is illustrated in Figure 5. For each $s \in V \backslash \{r\}$, let $t \equiv \text{pa}(s)$ denote the unique parent of s. In the context of this directed tree, for each child-parent pair $\{s, t\}$, the quantity $\log[P_{st}(x_s, x_t)/P_t(x_t)]$ is a transition function along the directed (t, s) edge.

For each child-parent pair $\{s, t\}$, we observe that a parent-to-child version of Bellman's equation (Bertsekas,

1995) holds:

$$\log P_s(x_s) \;=\; \kappa \max_{x'_t \in \mathcal{X}} \left\{ \log \frac{P_{st}(x_s, x'_t)}{P_t(x'_t)} + \log P_t(x'_t) \right\}$$

so that $\{\log P_s(x_s) \mid s \in V\}$ can be viewed as a collection of *cost-to-go* functions. As a result, we can apply the following procedure to draw a random sample from the set of configurations that attain the MAP cost:

1. Make a random choice of $\widehat{x}_r$ from the uniform distribution over all $x_r \in \mathcal{X}$ that maximize $P_r(x_r)$.
2. Proceeding down the tree from the root to leaves, for each child-parent pair $\{s, \mathrm{pa}(s)\} \equiv \{s, t\}$, choose $\widehat{x}_s$ uniformly from the set of $x_s \in \mathcal{X}$ that maximize $P_{st}(x_s, \widehat{x}_t)$. Note that the tree ordering ensures that the parent variable $\widehat{x}_t \equiv \widehat{x}_{\mathrm{pa}(s)}$ is always specified prior to any of its children.

3.2 Computing max-marginals via max-product

In this section, we introduce the max-product algorithm in application to trees. The perspective that we take differs from other treatments (e.g., Pearl, 1988), in that we interpret max-product as a particular method for computing the max-marginal reparameterization of Theorem 1. A by-product of this interpretation is a constructive proof of the existence of this max-marginal factorization.

For an arbitrary vertex s, consider the set of its neighbors $\Gamma(s) = \{\, t \mid (s,t) \in E \,\}$. For each $t \in \Gamma(s)$, let $\mathcal{T}(t)$ be the subgraph formed by the set of nodes that can be reached from t by paths that *do not* pass through node s. The key property of a tree is that each such subgraph $\mathcal{T}(t)$ is again a tree, and $\mathcal{T}(t)$ and $\mathcal{T}(u)$ are disjoint for $t \neq u$. In this way, each vertex t in the neighbor set $\Gamma(s)$ can be viewed as the root node of a subtree $\mathcal{T}(t)$, as illustrated in Figure 6.

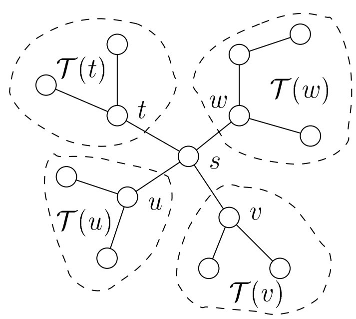


Figure 6. Decomposition of a tree, rooted at node s, into subtrees. Each neighbor (e.g., t) of node s is the root of a subtree (e.g., $\mathcal{T}(t)$). Subtrees $\mathcal{T}(t)$ and $\mathcal{T}(u)$ (for $t \neq u$) are disconnected when node s is blocked.

For each subtree $\mathcal{T}(t) = (V(\mathcal{T}(t)), E(\mathcal{T}(t)))$, let $\mathbf{x}_{\mathcal{T}(t)} = \{\, x_u \mid u \in \mathcal{T}(t)\}$. Moreover, let

$$p(\mathbf{x}_{\mathcal{T}(t)}; \boldsymbol{\psi}_{\mathcal{T}(t)}) \;\propto\; \prod_{u \,\in\, V(\mathcal{T}(t))} \psi_u(x_u) \prod_{(u,v) \in E(\mathcal{T}(t))} \psi_{uv}(x_u, x_v) \tag{9}$$

denote the subset of terms in equation (1) associated with vertices or edges in $\mathcal{T}(t)$. With this notation, the computation of P_s can be broken down into a set of subproblems, one for each subtree $\mathcal{T}(t)$:

$$P_s(x_s) \;=\; \kappa\, \psi_s(x_s) \prod_{t \in \Gamma(s)} \max_{\mathbf{x}'_{\mathcal{T}(t)}} \left\{ \psi_{st}(x_s, x'_t)\;\; p(\mathbf{x}'_{\mathcal{T}(t)}; \boldsymbol{\psi}_{\mathcal{T}(t)}) \right\} \tag{10}$$

Each of the subproblems $F(x_s; \boldsymbol{\psi}_{\mathcal{T}(t)}) = \max_{\mathbf{x}'_{\mathcal{T}(t)}} \psi_{st}(x_s, x'_t)\;\; p(\mathbf{x}'_{\mathcal{T}(t)}; \boldsymbol{\psi}_{\mathcal{T}(t)})$ is again a tree-structured maximization, and so can be then broken down recursively in a similar fashion. Similarly, the computation of P_{st} can be expressed in terms of such subproblems:

$$P_{st}(x_s, x_t) \;=\; \kappa\, \psi_s(x_s)\psi_t(x_t)\psi_{st}(x_s, x_t) \prod_{u \in \Gamma(s)/t} F(x_s; \boldsymbol{\psi}_{\mathcal{T}(u)}) \prod_{v \in \Gamma(t)/s} F(x_t; \boldsymbol{\psi}_{\mathcal{T}(v)}) \tag{11}$$

Therefore, in order to compute its max-marginal, each node s needs to receive the quantity $F(x_s; \boldsymbol{\psi}_{\mathcal{T}(t)})$ from each of its neighbors $t \in \Gamma(s)$. The *max-product* algorithm computes these quantities efficiently by a parallel set of message-passing operations. At each iteration $n = 0, 1, 2 \ldots$, every node $t \in V$ passes a message, denoted by $M^n_{ts}(x_s)$, to each of its neighbors $s \in \Gamma(s)$. Observe that the messages passed to node s are, in fact, functions of x_s. Messages are then updated according to the following recursion:

$$M^{n+1}_{ts}(x_s) \quad = \quad \kappa \max_{x'_t} \Big\{ \psi_{st}(x_s, x'_t)\, \psi_t(x'_t) \prod_{u \in \Gamma(t)/s} M^n_{ut}(x'_t) \Big\} \tag{12}$$

It can be shown (e.g., Pearl, 1988) that for any tree-structured graph, the message update equation (12) converges to a unique fixed point $\mathbf{M}^* = \{M^*_{st}\}$ after a finite number of iterations. The converged values of the messages $\mathbf{M}^*$ define functions T^*_s and T^*_{st} on the nodes and edges as follows:

$$T^*_s(x_s) \quad = \quad \kappa\, \psi_s(x_s) \prod_{u \in \Gamma(s)} M^*_{us}(x_s) \tag{13a}$$

$$T^*_{st}(x_s, x_t) \quad = \quad \kappa\, \psi_s(x_s)\, \psi_t(x_t)\;\; \psi_{st}(x_s, x_t) \prod_{u \in \Gamma(s)/t} M^*_{us}(x_s) \prod_{u \in \Gamma(t)/s} M^*_{ut}(x_t) \tag{13b}$$

Note the parallel between the structure of equations (13a) and (13b), and that of equations (10) and (11).

We now prove the following important properties of these functions:

(a) The functions $\mathbf{T}^*$ define an alternative parameterization of the distribution $p(\mathbf{x})$.

(b) For a tree-structured graph, the functions $\mathbf{T}^*$ are, in fact, equivalent to the max-marginals defined in equations (5) and (6) respectively.

To establish property (a), we consider the distribution defined by the collection $\mathbf{T}^* = \{T^*_s, T^*_{st}\}$ as follows:

$$p(\mathbf{x}; \mathbf{T}^*) \quad \propto \quad \prod_{s \in V} T^*_s(x_s) \prod_{(s,t) \in E} \frac{T^*_{st}(x_s, x_t)}{T^*_s(x_s) T^*_t(x_t)} \tag{14}$$

Lemma 1 (Equivalence of distributions). *The distribution defined in equation* (14) *is an alternative factorization of the original distribution* $p(\mathbf{x})$.

Proof. From the representation of T^*_s and T^*_{st} given in equations (13a) and (13b) respectively, we have

$$\frac{T^*_{st}(x_s, x_t)}{T^*_s(x_s) T^*_t(x_t)} \quad = \quad \kappa\, \frac{\psi_{st}(x_s, x_t)}{M^*_{st}(x_t) M^*_{ts}(x_s)}$$

Substituting this relation, as well as equation (13a), into the definition of $p(\mathbf{x}; \mathbf{T}^*)$ yields:

$$p(\mathbf{x}; \mathbf{T}^*) \quad \propto \quad \prod_{s \in V} \Big[\psi_s(x_s) \prod_{u \in \Gamma(s)} M^*_{us}(x_s) \Big] \prod_{(s,t) \in E} \frac{\psi_{st}(x_s, x_t)}{M^*_{st}(x_t) M^*_{ts}(x_s)}$$

which can be seen, after cancelling out the messages, to be equivalent to the original distribution $p(\mathbf{x})$. □

Note that from the message-passing updates in equation (12), the functions T^*_{st} and T^*_s defined in equations (13b) and (13a) satisfy the relation:

$$\kappa \max_{x'_t \in \mathcal{X}} T^*_{st}(x_s, x'_t) = T^*_s(x_s) \quad \text{for all} \quad x_s \in \mathcal{X} \tag{15}$$

where κ is a positive normalization factor that may depend on (s,t). Of course, the max-marginals P_s and P_{st} themselves also satisfy the local optimality condition of equation (15). Indeed, it turns out that for trees, this local optimality is sufficient to guarantee global optimality on the entire tree.

Proposition 1 (Local to global optimality). *For any tree-structured graph, the quantities T^*_s and T^*_{st} defined in equations* (13a) *and* (13b) *are equivalent to the max-marginals P_s and P_{st} defined in equations* (5) *and* (6).

Proof. There are various proofs of this result (e.g., Cowell et al., 1999; Dawid, 1992; Pearl, 1988). The proof that we present in Appendix A entails exploiting the local edgewise optimality of equation (15) to remove edges from the graph one at a time. In the end, we are left with a single edge (s,t), for which it is clear that T^*_{st} is the max-marginal. □

4 Max-product on graphs with cycles

This section presents a reparameterization framework for understanding and analyzing the max-product algorithm on arbitrary graphs. For graphs with cycles, the usual formulation of the max-product algorithm corresponds to applying the message updates of equation (12), while ignoring the presence of cycles. Unlike the case of trees, the algorithm is no longer guaranteed to converge; moreover, even if it does, the quantities T_s^* and T_{st}^* defined in equations (13a) and (13b) will not generally be equivalent to the true max-marginals. As a result, the max-product assignment — obtained by maximizing the individual T_s^* at each node — will not, in general, be an MAP assignment. Most previous work has focused on the dynamics of the message-passing updates themselves. The analysis presented here, in contrast, follows in the spirit of Section 3; we show that the max-product algorithm (and variations thereof) can be interpreted as reparameterizing the original distribution $p(\mathbf{x})$ in terms of a set of *pseudo-max-marginals* — namely, the set of T_s^* and T_{st}^* at every node and edge of the graph with cycles.

Focusing on the pseudo-max-marginals allows a great deal of the intuition for trees, as developed in Section 3, to be extended in a natural manner to graphs with cycles. We show that at a fixed point of max-product message-passing (or reparameterization) updates, the pseudo-max-marginals are required to satisfy the same edgewise consistency as the tree case. Based on our understanding of tree-structured problems, it then follows that the collection $\mathbf{T}^* = \{T_s^*, T_{st}^*\}$, while not equivalent to the max-marginals of the original distribution $p(\mathbf{x})$, must correspond to a set of max-marginals for modified problems on *every* spanning tree of the graph. As would be expected, this spanning tree consistency strongly constrains the associated max-product assignment.

This section begins with an introduction to the reparameterization view of max-product on graphs with cycles. We then establish that the max-product algorithm has at least one fixed point for positive compatibility functions on an arbitrary graph with cycles. In the context of reparameterization, this existence result implies that any distribution with positive compatibility functions can be refactored in terms of a set of pseudo-max-marginals that are consistent on *every* tree of the graph. We next turn to the question of specifying the max-product assignment. We show that for a graph with cycles, the consequences of non-unique maximizers at even one of the nodes are more severe than for a tree-structured distribution. For a graph with cycles, the full set of single node and pairwise pseudo max-marginals may fail to provide useful information concerning the MAP estimate. As a result, the use of the max-product algorithm, in either message-passing or reparameterization form, requires unique single-node maximizers, an assumption that we state at the start of Subsection 4.3. Under this assumption, we prove via a simple argument that the max-product assignment is guaranteed to be a global optimum of any portion of the cost function formed by terms from a subgraph with at most one cycle (per connected component). Based on this characterization, we also develop upper bounds on the error in the max-product algorithm — that is, the difference between the cost of the (optimal) MAP assignment, and the cost of the max-product assignment.

4.1 Reparameterization view of max-product

In this section, we present the reparameterization viewpoint of the max-product algorithm on a graph with cycles. Our approach is to build on Section 3, where we showed how the MAP problem on a tree can be solved by reparameterizing the distribution in terms of its max-marginals. To demonstrate how reparameterization can be applied to graphs with cycles, it is convenient to define and analyze a sequence of updates on spanning trees of the graph. Overall, a wide class of algorithms, including the ordinary message-passing form of max-product, can be interpreted as performing reparameterization. We make use of the tree-based updates presented in this section as an exemplar for illustrating the properties of this class of algorithms.

4.1.1 Tree-based reparameterization updates

We begin with the simple observation that embedded within any graph with cycles are a number of spanning trees, as illustrated in Figure 7. Our strategy, then, is to formulate and study a sequence of modified problems on these spanning trees. In particular, we consider a collection $\{\mathcal{T}^i\}$ of spanning trees, specified by edge sets $\{E^i\}$.

Our starting point, as illustrated in Figure 8(a), is a distribution specified as a product of compatibility functions over the full graph with cycles G (as in equation (2)). Given some spanning tree $\mathcal{T}^i$ with edge set E^i, we isolate those components of $p(\mathbf{x})$ that correspond to nodes and edges in the spanning tree:

$$p^i(\mathbf{x}) \quad \propto \quad \prod_{s \in V} \psi_s(x_s) \prod_{(s,t) \in E^i} \psi_{st}(x_s, x_t)$$

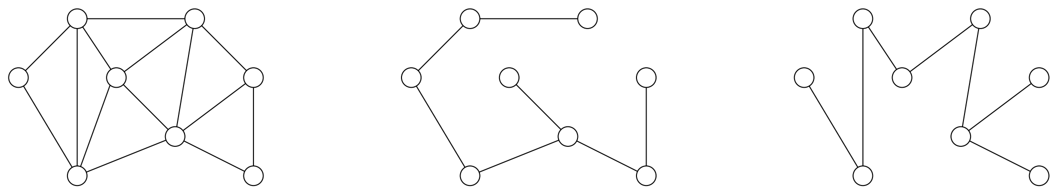


Figure 7. Any graph with cycles has several embedded spanning trees. (a) Original graph G. (b) One embedded spanning tree $\mathcal{T}^1$. (c) Another embedded spanning tree $\mathcal{T}^2$.

The term $r^i(\mathbf{x})$ denotes the residual set of terms corresponding to edges not in the spanning tree $\mathcal{T}^i$. Overall, we have used the spanning tree to decompose the original distribution $p(\mathbf{x})$ into the product $p^i(\mathbf{x})\, r^i(\mathbf{x})$ of the spanning tree component and the residual component.

As illustrated in Figure 8(b), the distribution $p^i(\mathbf{x})$ is a tree-structured, meaning that it can be reparameterized in terms of its max-marginals, denoted by $\{T_s, T_{st}\}$. Accordingly, we compute these max-marginals, and then use

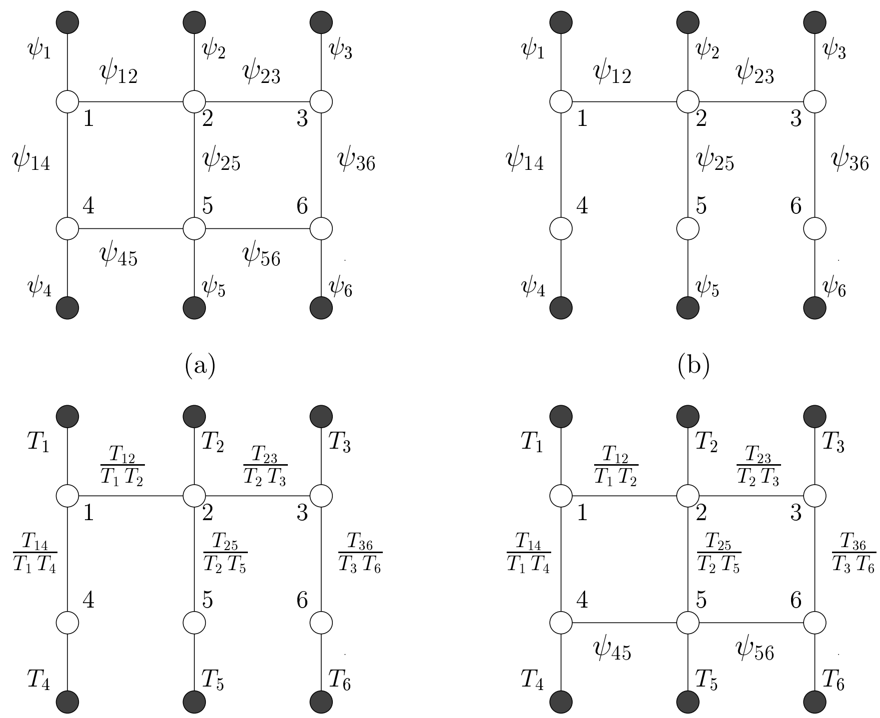


Figure 8. Illustration of reparameterization form of max-product updates on spanning trees. (a) Original parameterization of $p(\mathbf{x})$ in terms of compatibility functions. (b) Tree distribution $p^i(\mathbf{x})$ formed of components on spanning tree $\mathcal{T}^i$. (c) Spanning tree after reparameterization update has been performed. (d) Full graph after one iteration, with original functions re-instated on edges not in the tree.

them to specify the canonical tree factorization of $p^i(\mathbf{x})$ of equation (7). This reparameterized form of the tree distribution is illustrated in in Figure 8(c). We conclude the update by reinstating component $r^i(\mathbf{x})$ (without any modification), leaving the overall distribution in the form shown in Figure 8(d).

Note that this update simply amounts to specifying an alternative factorization, or reparameterization, of the original distribution $p(\mathbf{x})$. An important property of this reparameterization is that the modified compatibility functions on nodes and edges in the spanning tree $\mathcal{T}^i$ are, in fact, a consistent set of max-marginals on that particular tree. In other words, the parameterization is now *tree-consistent* with respect to the tree $\mathcal{T}^i$.

Of course, there are no guarantees that tree-consistency will hold for some other tree distinct from $\mathcal{T}^i$. However, the ultimate goal is to obtain a reparameterization for which tree-consistency does hold for every tree simultaneously. In order to move towards this goal, a subsequent iteration entails isolating a different spanning tree $\mathcal{T}^j$, and following the same sequence of steps. That is, starting from the factorization obtained after the preceding reparameterization step, we again perform the decomposition $p(\mathbf{x}) \propto p^j(\mathbf{x})r^j(\mathbf{x})$; reparameterize the tree distribution $p^j(\mathbf{x})$; and then reinstate the residual term.

More formally, each of these updates can be expressed as a functional mapping $\mathbf{T}^n \mapsto \mathbf{T}^{n+1}$, where $\mathbf{T}^n = \{T_s^n, T_{st}^n\}$. At each iteration $n = 0, 1, 2, \ldots$, the collection $\mathbf{T}^n$ specifies a particular parameterization of the original distribution $p(\mathbf{x})$:

$$p(\mathbf{x}; \mathbf{T}^n) \quad \propto \quad \prod_{s \in V} T_s^n(x_s) \prod_{(s,t) \in E} \frac{T_{st}^n(x_s, x_t)}{\{[\max_{x_s'} T_{st}^n(x_s', x_t)][\max_{x_t'} T_{st}^n(x_s, x_t')]\}} \tag{16}$$

We let $\mathcal{T}^0, \ldots, \mathcal{T}^{L-1}$ denote a collection of spanning trees, with associated edge sets $E^1, \ldots, E^L$. The only restriction placed on the choice of these spanning trees is that each edge in the graph appear in at least one spanning tree. At each iteration, we choose a spanning tree index $i(n) \in \{1, \ldots, L-1\}$. To be concrete, although a variety of other orderings are possible (e.g., Censor and Zenios, 1988), we focus on the cyclic ordering $i(n) \equiv n \pmod{L}$. With this set-up and notation, Algorithm 1 gives a formal specification of the reparameterization updates:

Algorithm 1 (Tree reparameterization max-product).

1. At iteration $n = 0$, initialize $\mathbf{T}^0$ as:

$$T_{st}^0(x_s, x_t) \quad = \quad \kappa\, \psi_s(x_s)\psi_t(x_t)\psi_{st}(x_s, x_t) \tag{17a}$$

$$T_s^0(x_s) \quad = \quad \kappa\, \psi_s(x_s) \prod_{t \in \Gamma(s)} [\max_{x_t'} \psi_{st}(x_s, x_t')\psi_t(x_t')] \tag{17b}$$

2. At iterations $n = 1, 2, \ldots$, use spanning tree $\mathcal{T}^{i(n)}$ with edge set $E^{i(n)}$. For each edge $(s,t) \in E^{i(n)}$ and for each node $s \in V$, update pseudo-max-marginals as follows:

$$T_{st}^{n+1}(x_s, x_t) \quad = \quad \max_{\{\mathbf{x}' \mid (x_s', x_t') = (x_s, x_t)\}} p^{i(n)}(\mathbf{x}'; \mathbf{T}^n) \tag{18a}$$

$$T_s^{n+1}(x_s) \quad = \quad \max_{\{\mathbf{x}' \mid x_s' = x_s\}} p^{i(n)}(\mathbf{x}'; \mathbf{T}^n) \tag{18b}$$

   For each edge $(s,t) \in E \backslash E^{i(n)}$, set $T_{st}^{n+1} = T_{st}^n$.

Remarks: At one level, Algorithm 1 can be viewed as a particular tree-based schedule for message passing. In this message-passing view, each iteration entails fixing all those messages on edges not in the spanning tree, and then updating the remaining messages on tree edges until convergence. Conversely, the parallel message-passing updates in equation (12) can be reformulated as a local reparameterization algorithm, in which exact computations are performed over very simple trees consisting of single edges. The details of this equivalence for the closely related sum-product or belief propagation algorithm can be found in the thesis (Wainwright, 2002).

4.1.2 Properties of fixed points

Highlighted by the reparameterization formulation of Algorithm 1 is the fact that the original distribution is not changed, a property that we formalize below. The parallel message-passing form of max-product (12) also does not change the distribution, since (as noted above) it can be reformulated as reparameterization over single edges. More generally, other procedures for scheduling messages for max-product correspond to reparameterization procedures using more complex (but perhaps not spanning) acyclic subgraphs. From this equivalence, we can immediately conclude that each fixed point $\mathbf{M}^*$ of any max-product algorithm (i.e., using any particular message-passing schedule) corresponds to a fixed point of tree-reparameterization (Algorithm 1), where that correspondence is specified by equations (13a) and (13b). As a result, we can exploit the insights afforded by tree-reparameterization to analyze the fixed points of a rich class of max-product algorithms. We summarize our results in the following:

Proposition 2 (Properties of max-product fixed points).
(a) Any fixed point $\mathbf{T}^*$ of tree reparameterization updates or max-product message-passing is a reparameterization of the original distribution. That is, the distribution

$$p(\mathbf{x}; \mathbf{T}^*) \quad \propto \quad \prod_{s \in V} T_s^*(x_s) \prod_{(s,t) \in E} \frac{T_{st}^*(x_s, x_t)}{T_s^*(x_s) T_t^*(x_t)}$$

is equivalent to the original one defined in equation (2).

(b) The elements of $\mathbf{T}^*$ are tree-consistent for any tree embedded within the graph with cycles. More precisely, given an arbitrary tree $\mathcal{T}$ with edge set $E(\mathcal{T}) \subset E$, form the tree-structured distribution:

$$p^{\mathcal{T}}(\mathbf{x}; \mathbf{T}^*) \quad \propto \quad \prod_{s \in V} T_s^*(x_s) \prod_{(s,t) \in E(\mathcal{T})} \frac{T_{st}^*(x_s, x_t)}{T_s^*(x_s) T_t^*(x_t)} \tag{19}$$

Then the elements $\{T_s \mid s \in V\}$ and $\{T_{st} \mid (s,t) \in E(\mathcal{T})\}$ are the correct max-marginals for $p^{\mathcal{T}}(\mathbf{x}; \mathbf{T}^*)$.

Proof. (a) In setting up Algorithm 1, we established that the initialized distribution $p(\mathbf{x}; \mathbf{T}^0)$ is equivalent to the original distribution, and moreover, that the reparameterization updates do not alter the distribution. I.e., $p(\mathbf{x}; \mathbf{T}^n) = p(\mathbf{x}; \mathbf{T}^0)$ for all iterations. Since the mapping $\mathbf{T}^n \mapsto p(\mathbf{x}; \mathbf{T}^n)$ is continuous, the set $\{\mathbf{U} \mid p(\mathbf{x}; \mathbf{U}) \equiv p(\mathbf{x}; \mathbf{T}^0)\}$ is closed. Therefore, the invariance property will also hold for any fixed point $\mathbf{T}^*$ of Algorithm 1. The invariance can be established for explicit message-passing updates by the argument of Lemma 1. (The proof of this lemma is equally applicable to graphs with cycles).
(b) If Algorithm 1 converges, then by definition of the updates, the set of pseudo-max-marginals $\mathbf{T}^* = \{T_s^*, T_{st}^*\}$ must be tree-consistent, in the sense of Proposition 2(b), with respect to each of the trees $\mathcal{T}^0, \ldots, \mathcal{T}^{L-1}$ used in the algorithm. By assumption, each edge (s,t) belongs to at least one of these spanning trees, meaning that each T_{st}^* and T_t^* are joint pairwise and single node max-marginals for at least one tree-structured distribution $p^i(\mathbf{x}; \mathbf{T}^*)$. As max-marginals, they satisfy the local consistency condition $\max_{x'_s} T_{st}^*(x'_s, x_t) = \kappa\, T_t^*(x_t)$; that is, the elements of $\mathbf{T}^*$ are locally consistent on every edge of the graph. From this point, we follow the proof of Proposition 1 to establish that $\mathbf{T}^*$ must be tree-consistent for an arbitrary tree embedded within the graph. □

4.2 Existence of max-product fixed points

Proposition 2 characterizes the fixed points of tree-reparameterization max-product (Algorithm 1), synchronous max-product message-passing updates (equation (12)), as well as other variants. However, for general graphs, it has been an open question of whether or not the max-product algorithm has fixed points. Previous work (e.g., Weiss, 2000; Horn, 1999; Aji et al., 1998) has established the existence of max-product fixed points for positive compatibility functions on graphs consisting of a single cycle. In this section, we generalize this results by proving that fixed points exist for a positive distribution on an arbitrary graph with cycles:

Theorem 2 (Existence of max-product fixed points). *Any positive distribution $p(\mathbf{x})$ on a graph with pairwise cliques can be reparameterized in terms of a tree-consistent set $\mathbf{T}^*$ of pseudo-max-marginals.*

Proof. We prove the theorem by showing that a suitably modified form of the message-passing updates (12) satisfies the hypotheses of Brouwer's fixed point theorem (e.g., Ortega and Rheinboldt, 2000). We begin by reformulating the updates in terms of log messages. The message M_{st} from node s to node t is a vector of length m, where the k^{th} element denotes its value when $x_t = k$. For this proof, ζ_{st} denotes a m-vector that corresponds to the logs of the elements of M_{st}; the notation $\zeta_{st;k}$ denotes the k^{th} element of this log message. For each edge $(s,t) \in E$, note that the log message from s to t, denoted ζ_{st}, and its counterpart from t to s, denoted ζ_{ts}, are distinct quantities. Accordingly, we let $\boldsymbol{\zeta} = \{\zeta_{st},\ \zeta_{ts} \mid (s,t) \in E\ \}$ be a vector in $\mathbb{R}^D$, where $D = 2\,|E|\; m$.

We now define a mapping that is equivalent to the message updates of equation (12). For each edge $(s,t) \in E$ and each element $k \in \mathcal{X}$, we begin by defining a mapping $\mathcal{L}_{st}(\cdot\,; k) : \mathbb{R}^D \to \mathbb{R}$ as follows:

$$\mathcal{L}_{st}(\boldsymbol{\zeta}; k) \;=\; \max_{j \in \mathcal{X}} \Big[\log \psi_{st}(j,k) + \log \psi_s(j) + \sum_{u \in \Gamma(s)/t} \zeta_{us}(j) \;\Big] \tag{20}$$

Equation (20) is simply a re-statement of the message update equation (12) in terms of log messages.

Note that the message update equation (12) permits the messages to be renormalized by some factor κ. In the log domain, this rescaling is equivalent to adding a constant (independent of k) to all elements of the vector $\{ \mathcal{L}_{st}(\boldsymbol{\zeta}; k) \mid k \in \mathcal{X} \}$. For analytic purposes, it is convenient to exploit this freedom and define another collection of mappings $\mathcal{F}_{st}(\cdot\,; k) : \mathbb{R}^D \to \mathbb{R}$ as follows:

$$\mathcal{F}_{st}(\boldsymbol{\zeta}; k) \quad = \quad \mathcal{L}_{st}(\boldsymbol{\zeta}; k) - \frac{1}{m} \sum_{j \in \mathcal{X}} \mathcal{L}_{st}(\boldsymbol{\zeta}; j) \tag{21}$$

There are a total of $D = 2\,|E|\,m$ of these real-valued mappings ($2\,m$ for each edge $(s,t) \in E$). Thus, the overall map $\mathcal{F} : \mathbb{R}^D \to \mathbb{R}^D$ is defined by this full collection of mappings:

$$\mathcal{F}(\boldsymbol{\zeta}) = \{ \mathcal{F}_{st}(\boldsymbol{\zeta}; k),\;\; \mathcal{F}_{ts}(\boldsymbol{\zeta}; k) \;\big|\; k \in \mathcal{X},\; (s,t) \in E \} \tag{22}$$

With this set-up, the theorem itself is proved by applying Brouwer's fixed point theorem (Ortega and Rheinboldt, 2000). In order to apply this theorem, we require the following two lemmas:

Lemma 2 (Continuity). *The mapping $\mathcal{F}$ is continuous.*

Lemma 3 (Uniformly bounded). *For any distribution $p(\mathbf{x})$ formed of positive compatibility functions $\{\psi_s, \psi_{st}\}$, there is a finite constant c such that $\|\mathcal{F}(\boldsymbol{\zeta})\|_\infty \;\leq\; c$ for all $\boldsymbol{\zeta} \in \mathbb{R}^D$.*

Proofs of these lemmas can be found in Appendix B. We now complete the proof of the theorem. Define $A \;\triangleq\; \{ \boldsymbol{\zeta} \in \mathbb{R}^D \mid \|\boldsymbol{\zeta}\|_\infty \;\leq\; c \}$. Then A is a compact and convex set, and by Lemma 3, the image $\mathcal{F}(A)$ is contained within A. Moreover, $\mathcal{F}$ is continuous by Lemma 2, so that by the Brouwer fixed-point theorem (Ortega and Rheinboldt, 2000), we conclude that $\mathcal{F}$ has a fixed point in A. $\square$

The implication of Theorem 2, when stated in the language of Proposition 2, is rather striking: *any* positive distribution $p(\mathbf{x})$ on an arbitrary graph with cycles can be reparameterized in terms of a set of functions $\mathbf{T}^*$ that are consistent single node and pairwise max-marginals for every spanning tree of the graph. For a tree, this assertion is a well-known fact (see Theorem 1). In contrast, the existence of such a parameterization for an arbitrary graph with cycles is by no means obvious.

4.3 Specification of the max-product assignment

Our analysis up to this point has focused on properties of the pseudo-max-marginals $\mathbf{T}^*$ that are computed by any max-product/reparameterization algorithm. In the remaining analysis, our goal is to illuminate properties of the *max-product assignment*. Before doing so, however, we need to address the following question: given a fixed point $\mathbf{T}^* = \{T^*_s, T^*_{st}\}$ of the max-product updates, how should the max-product assignment $\mathbf{x}^*$ be defined? First of all, suppose that the following condition holds:

Asumption 1 (Unique maximum). For all nodes $s \in V$, the maximum $\max_{x's \in \mathcal{X}} T^*_s(x'_s)$ is attained at a unique point x^*_s.

In this case, the max-product assignment $\mathbf{x}^*$ can be defined unambiguously via $x^*_s = \arg\max_{x'_s \in \mathcal{X}} T^*_s(x'_s)$ for all nodes $s \in V$.

Suppose, on the other hand, that Assumption 1 does not hold for some (or all) of the nodes in the graph. In Section 3.1, we demonstrated how if Assumption 1 fails for a tree-structured problem, then interpreting the quantities $\log T^*_s(x_s)$ as cost-to-go functions, as in dynamic programming (Bertsekas, 1995), leads to a recursive procedure for drawing random samples from the set of all configurations achieving the MAP cost. Whether or not this same type of interpretation and sampling procedure applies to graph with cycles is an issue that does not appear to have been addressed in the literature. In this section, we demonstrate via some simple examples that when cycles are present in the graph, the cost-to-go interpretation no longer applies.

An important aspect of the tree sampling procedure of Section 3.1 is that any sampled configuration $\mathbf{x}^*$ is guaranteed to satisfy the following properties:

Node optimality: For each node $s \in V$, x^*_s achieves the optimum $\max_{x'_s \in \mathcal{X}} T^*_s(x'_s)$.

Edgewise optimality: For each edge $(s,t) \in E$, (x^*_s, x^*_t) achieves the optimum $\max_{x'_s, x'_t \in \mathcal{X}} T^*_{st}(x'_s, x'_t)$.

As the examples below demonstrate, it is possible, when Assumption 1 fails for a graph with cycles, that there are no configurations that satisfy these local optimality properties for all nodes and edges.

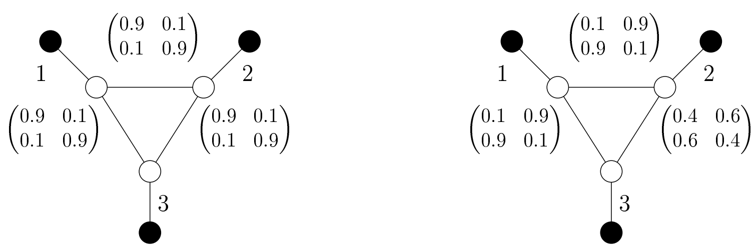
Example 2. We begin with an extension of Example 1. Consider the graph consisting of a single cycle on three nodes, as illustrated in Figure 9, and a binary vector $\mathbf{x} \in \{0,1\}^3$ with compatibility functions of the form:

$$\begin{aligned} \psi_s(x_s) &= [\,1 \quad 1\,]' \qquad \text{for all } s \in V \\ \psi_{st}(x_s, x_t) &= \begin{pmatrix} \beta_{st} & (1-\beta_{st}) \\ (1-\beta_{st}) & \beta_{st} \end{pmatrix} \qquad \text{for all } (s,t) \in E = \{\,(1,2),\ (1,3),\ (2,3)\,\} \end{aligned}$$

where $\beta_{st} \in (0,1)$ are parameters to be specified. Observe that for any values of β_{st} in $(0,1)$, the following local consistency condition holds for each edge $(s,t) \in E$:

$$\max_{x'_s \in \mathcal{X}} \psi_{st}(x'_s, x_t)\psi_s(x'_s)\psi_t(x_t) \;=\; \kappa \;\psi_t(x_t) \tag{23}$$

for all $x_t \in \mathcal{X}$. As a consequence, if we define $T^*_s(x_s) = \psi_s(x_s)$ and $T^*_{st}(x_s, x_t) = \psi_{st}(x_s, x_t)\ \psi_s(x_s)\ \psi_t(x_t)$, then $\mathbf{T}^* = \{T^*_s, T^*_{st}\}$ is already a fixed point of the max-product algorithm, when considered in terms of reparameterization. Moreover, note that since $T^*_s(x_s) = [1 \;\; 1]'$, Assumption 1 fails for every node $s \in V$.



(a) $\beta_{st} = 0.9$ (b) $\beta_{12} = \beta_{13} = 0.1$ and $\beta_{23} = 0.4$

Figure 9. Illustration of symmetric problems for which the MAP value is attained by multiple configurations. The single-node pseudo-max-marginals T^*_s are uniform, and therefore do not yield any information. In panel (a), there are distributions consistent with these pseudo-max-marginals; this is not true for panel (b).

(a) First of all, suppose $\beta_{st} = 0.9$ for all edges (s,t), as illustrated in panel (a). In this case, the MAP cost is achieved by both $\mathbf{0} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ and $\mathbf{1} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$. Suppose that we try to mimic the cost-to-go sampling procedure described for a tree in Section 3.1. In particular, let us start by drawing a random sample from the pseudo-max-marginal $T^*_1(x_1) \equiv \begin{bmatrix} 1 & 1 \end{bmatrix}'$ — that is, choosing x^*_1 equal to 0 or 1 with equal probability. We then proceed around the cycle in the direction $1 \to 2 \to 3$, first drawing x^*_2 uniformly from those configurations that maximize $T^*_{12}(x^*_1, x_2)$, and then x^*_3 uniformly from the configurations that maximize $T^*_{23}(x^*_2, x_3)$. It can be seen that the output of this sampling procedure is either $\mathbf{x}^* = \mathbf{0}$ or $\mathbf{x}^* = \mathbf{1}$, depending on whether x^*_1 was chosen to be 0 or 1 at the first step. Either configuration achieves the MAP cost. For this particular sampling procedure, the pseudo-max-marginal T^*_{13} on edge $(1,3)$ played no role whatsoever. Nonetheless, the constraint that it imposes — namely, that the subvector (x_1, x_3) either be equal to [0 0] or [1 1] — turns out to be satisfied by every possible output of the sampler. However, this consistency is simply fortuitous, as part (b) will demonstrate.

(b) Now suppose that $\beta_{12} = \beta_{13} = 0.1$ and $\beta_{23} = 0.4$, as illustrated in panel (b). In this case, it can be seen that the MAP cost is achieved by both [1 0 0] and [0 1 1]. Consider the sampling procedure $(1 \to 2 \to 3)$ described in part (a). Depending on whether x^*_1 is chosen to be 0 or 1 respectively at the first step, this sampler outputs either [0 1 0] or [1 0 1] respectively. Unlike part (a), *neither* of these outputs is a MAP configuration.

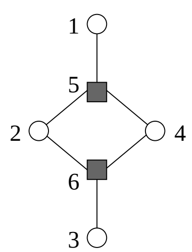
Of course, for this special case, we could obtain one of the two MAP configurations by sampling in a different order — say $2 \to 1 \to 3$. But no matter which order is used to draw samples, there is a key difference with the example of part (a). For any order, there will be some pseudo-max-marginal T^*_{st} that plays no role in the sampling. (For the $2 \to 1 \to 3$ order, T^*_{23} plays no role). Unlike part (a), the constraint imposed by this pseudo-max-marginal — namely, that (x_2, x_3) either be equal to [1 0] or [0 1] — is never satisfied by *any* sampled configuration over the entire graph with cycles.

This inconsistency arises because the pseudo-max-marginals $\mathbf{T}^*$ impose a set of consistency constraints that cannot be simultaneously satisfied. In particular, for a configuration $\mathbf{x}$ to be edgewise optimal, each pair of random variables (x_s, x_t) linked by T^*_{st} must take opposite values for all three edges $(s,t) \in E$. There is no configuration $\mathbf{x} \in \{0,1\}^3$ that satisfies all three conditions simultaneously.

$\square$

For the problems considered in Example 2, the uniform pseudo-max-marginals $T^*_s(x_s) = [1 \;\; 1]'$ were, at worst, unhelpful. The following example shows that when Assumption 1 is not satisfied at all nodes, the pseudo-max-marginals can be quite misleading (as opposed to simply unhelpful), even for a graph containing only a single cycle.

Example 3 (Misleading pseudo-max-marginals). Consider the 6-node graph shown in Figure 10(a). The random variables x_5 and x_6 are special in that they take four possible values, whereas all the other variables $\{x_1, x_2, x_3, x_4\}$ are binary-valued. A set of single node and pairwise pseudo-max-marginals are given in panels (b) and (c) respectively of Figure 10, in which vectors and matrices are used to denote the values of the pseudo max-marginals T^*_s and T^*_{st} respectively. Observe that quantities T^*_5 and T^*_6 are represented by 4-vectors, since the random variables x_5 and x_6 each take four possible values; the remaining compatibility functions $T^*_s, s \neq 5, 6$ are represented by 2-vectors. Each of the edges (s,t) in the graph connects either node 5 or 6 to one of the other binary nodes, so that each compatibility function T^*_{st} is represented by a 2×4 matrix. The quantity $\epsilon \geq 0$ in



(a)

$$
\begin{aligned}
T^*_1(x_1) &= [\,0.9 \;\; 0.1\,]' \\
T^*_3(x_3) &= [\,0.4 \;\; 0.6\,]' \\
T^*_s(x_s) &= [\,1 \;\; 1\,]' \qquad \text{for } s \in \{2,4\} \\
T^*_5(x_5) &= [0.9 \;\; 0.9 \;\; 0.1 \;\; 0.1]' \\
T^*_6(x_6) &= [0.4 \;\; 0.4 \;\; 0.6 \;\; 0.6]'
\end{aligned}
$$

(b)

$$
\begin{aligned}
T^*_{15}(x_1, x_5) &= \begin{pmatrix} 0.9 & 0.9 & \epsilon & \epsilon \\ \epsilon & \epsilon & 0.1 & 0.1 \end{pmatrix} \\
T^*_{25}(x_2, x_5) &= \begin{pmatrix} 0.9 & \epsilon & 0.1 & \epsilon \\ \epsilon & 0.9 & \epsilon & 0.1 \end{pmatrix} \\
T^*_{45}(x_4, x_5) &= \begin{pmatrix} 0.9 & \epsilon & \epsilon & 0.1 \\ \epsilon & 0.9 & 0.1 & \epsilon \end{pmatrix} \\
T^*_{36}(x_3, x_6) &= \begin{pmatrix} 0.4 & 0.4 & \epsilon & \epsilon \\ \epsilon & \epsilon & 0.6 & 0.6 \end{pmatrix} \\
T^*_{26}(x_2, x_6) &= \begin{pmatrix} 0.4 & \epsilon & 0.6 & \epsilon \\ \epsilon & 0.4 & \epsilon & 0.6 \end{pmatrix} \\
T^*_{46}(x_4, x_6) &= \begin{pmatrix} 0.6 & \epsilon & \epsilon & 0.4 \\ \epsilon & 0.6 & 0.4 & \epsilon \end{pmatrix}
\end{aligned}
$$

(c)

Figure 10. A problem for which the pseudo-max-marginals are misleading. (a) Structure of graphical model. (b) Single node pseudo-max-marginals. (c) Joint pairwise pseudo-max-marginals.

panel (c) is a parameter to be specified. By construction, as long as $\epsilon < 0.1$, the full collection $\mathbf{T}^* = \{T^*_s, T^*_{st}\}$ of pseudo-max-marginals specified in this way satisfies local edgewise consistency; that is, for each edge (s,t), we have $\max_{x'_s \in \mathcal{X}_s} T^*_{st}(x'_s, x_t) = \kappa \, T^*_t(x_t)$. Therefore, $\mathbf{T}^*$ is a fixed point of max-product reparameterization for $0 \leq \epsilon < 0.1$.

We use these pseudo-max-marginals to form the distribution $p(\mathbf{x}) \propto \prod_{s \in V} T^*_s(x_s) \prod_{(s,t) \in E} \frac{T^*_{st}(x_s, x_t)}{T^*_s(x_s) T^*_t(x_t)}$. The nature of this distribution is best understood by considering first the extreme $\epsilon = 0$, in which case the set of functions T^*_{st}, each relating one of the binary nodes with one of the 4-value nodes (nodes 5 and 6) act, in essence, to enforce a set of parity checks for the set of nodes $\{1,2,4\}$ and the set $\{2,3,4\}$. In particular, when $\epsilon = 0$, any configuration $\mathbf{x}$ for which either of the subvectors (x_1, x_2, x_4) or (x_2, x_3, x_4) have odd parity are forbidden (i.e., they are given zero probability). As a consequence, for $\epsilon = 0$, all configurations with positive probability have $x_1 = x_3$.

When ϵ is strictly larger than zero (but still very small), then the model places positive probability on all configurations, but very low ($\mathcal{O}(\epsilon)$) probability on any configuration for which $x_1 \neq x_3$; as a consequence, any configuration $\widehat{\mathbf{x}}$ achieving the MAP value will satisfy $\widehat{x}_1 = \widehat{x}_3$. In contrast, the pseudo-max-marginals at nodes 3 and node 1 suggest that we should set $x_3 = 1$ and $x_1 = 0$ respectively. With these choices, there is no assignment for the remaining variables that is edgewise optimal with respect to all the joint $\{T^*_{st}\}$; that is, at least one edge will contribute an ϵ-term. Thus, any configuration that is optimal with respect to the single node pseudo-max-marginals will have $\mathcal{O}(\epsilon)$ probability in the overall model. The single node and pairwise pseudo max-marginals have given conflicting, and in this case, quite misleading information. □

4.4 Characterization of max-product assignment

Based on the examples in the previous section, it is impossible to define and/or interpret a max-product assignment when Assumption 1 is not satisfied. Accordingly, for the analysis in the remainder of Section 4, we assume that Assumption 1 holds. We let $\mathbf{x}^* = \{x^*_s \mid s \in V\}$ denote the max-product assignment, with elements $x^*_s = \arg\max_{x'_s} T^*_s(x'_s)$. For the purposes of analysis, it is convenient to use the pseudo-max-marginals to define a cost function as follows:

$$F(\mathbf{x}; \mathbf{T}^*; G) = \sum_{s \in V} \log T^*_s(x_s) + \sum_{(s,t) \in E} \log \frac{T^*_{st}(x_s, x_t)}{T^*_s(x_s) T^*_t(x_t)} \tag{24}$$

From Proposition 2(a), it can be seen that $F(\mathbf{x}; \mathbf{T}^*; G)$ is equivalent (up to an additive constant) to the original cost function $J(\mathbf{x}; G)$ defined in equation (4). Therefore, statements about the optimality of $\mathbf{x}^*$ with respect to $F(\mathbf{x}; \mathbf{T}^*; G)$ can be translated directly to optimality statements of $\mathbf{x}^*$ with respect to $J(\mathbf{x}; G)$.

It is also useful to isolate those components of the cost function (24) that correspond to a given subgraph $H = (V(H), E(H))$ of G:

$$F(\mathbf{x}; \mathbf{T}^*; H) = \sum_{s \in V(H)} \log T^*_s(x_s) + \sum_{(s,t) \in E(H)} \log \frac{T^*_{st}(x_s, x_t)}{T^*_s(x_s) T^*_t(x_t)} \tag{25}$$

In the analysis to follow, we often make statements of the form *"the max-product assignment* $\mathbf{x}^*$ *for* G *is optimal with respect to the subgraph* H*"*, by which we mean $\mathbf{x}^*$ maximizes $F(\mathbf{x}; \mathbf{T}^*; H)$.

In previous work, Weiss (2000) showed that the max-product assignment is correct for any graph with at most one cycle. Freeman and Weiss (2001) showed that in an arbitrary graph, the max-product assignment is a local optimum over any subset of nodes that form a *node-induced* subgraph with at most one cycle per connected component. (See Figure 1 for an illustration of a node-induced subgraph.) The main result in this subsection generalizes these previous results by showing that the max-product assignment is optimal with respect to *any* subgraph that contains at most one cycle per connected component.

The following lemma is central to our analysis:

Lemma 4. *For all* $x_s, x_t \in \mathcal{X}$*, we have:*

$$\log \frac{T^*_{st}(x^*_s, x^*_t)}{T^*_t(x^*_t)} \quad \geq \quad \log \frac{T^*_{st}(x_s, x_t)}{T^*_t(x_t)} \tag{26}$$

Proof. From the tree consistency stated in Proposition 2(b), we certainly have $\max_{x'_s} T^*_{st}(x'_s, x_t) = \kappa \;\, T^*_t(x_t)$ for all $x_t \in \mathcal{X}$. As a result, the quantity $\max_{x'_s} T^*_{st}(x'_s, x_t)/T^*_t(x_t)$ is a constant κ for all x_t. Moreover, when $x_t = x^*_t$, this maximum is attained at x^*_s. Therefore:

$$\begin{aligned}
\log \frac{T^*_{st}(x^*_s, x^*_t)}{T^*_t(x^*_t)} \quad &= \quad \max_{x'_s} \log \frac{T^*_{st}(x'_s, x^*_t)}{T^*_t(x^*_t)} \\
&= \quad \max_{x'_s} \log \frac{T^*_{st}(x'_s, x_t)}{T^*_t(x_t)} \\
&\geq \quad \log \frac{T^*_{st}(x_s, x_t)}{T^*_t(x_t)}
\end{aligned}$$

which holds for all $x_s, x_t \in \mathcal{X}$. □

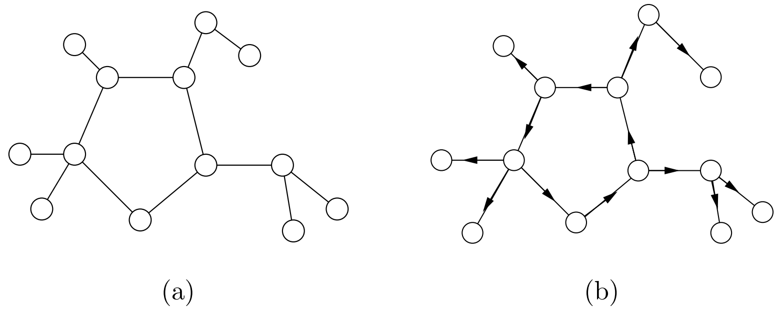


Figure 11. (a) A single cycle with trees dangling from some nodes. (b) An orientation (assignment of directions to the edges) of the graph in (a), such that each node has exactly one incoming edge.

Theorem 3 (Tree-plus-cycle optimality). *Let $\mathbf{T}^*$ be a fixed point of the max-product algorithm, with corresponding max-product assignment $\mathbf{x}^*$ that satisfies Assumption 1. Then $\mathbf{x}^*$ is a global optimum of $F(\mathbf{x}; \mathbf{T}^*; H)$ for all subgraphs H of G containing at most one cycle per connected component. That is, for all $\mathbf{x} \in \mathcal{X}^N$, we have:*

$$F(\mathbf{x}^*; \mathbf{T}^*; H) \quad \geq \quad F(\mathbf{x}; \mathbf{T}^*; H) \tag{27}$$

Proof. From Proposition 2(b), we know that $\mathbf{x}^*$ is optimal with respect to every tree $\mathcal{T}$ of G. We now extend this optimality to all subgraphs H with at most a single cycle per connected component. We assume that the subgraph has one connected component; if it has multiple components, we can apply the same argument separately to each component. As illustrated in Figure 11(a), any such subgraph H consists of a single cycle, with (possibly) a tree dangling from each node in the cycle. For such a graph, it is always possible to direct the edges such that each node has exactly one incoming edge (Figure 11(b)). Using this direction of the edges, we can write the cost function in the following way:

$$F(\mathbf{x}; \mathbf{T}^*; H) \quad = \quad \sum_{s \in V(H)} \log \frac{T^*_{st}(x_s, x_t)}{T^*_t(x_t)} \tag{28}$$

In this statement, for every node $s \in V(H)$, t is the unique node that sends a directed edge to s. Applying Lemma 4 to each term in equation (28) yields the statement of the theorem. □

Theorem 3 has a number of interesting corollaries. In particular, a special type of subgraph are *node-induced subgraphs*; see Figures 1 and 12 for illustrations. Thus, we can show that Theorem 3 implies the result of Freeman and Weiss (2001) mentioned above, which applies to those node-induced subgraphs with at most one cycle per connected component:

Corollary 1 (Optimality on node-induced subgraphs). *Let $\mathbf{x}^*$ be a max-product assignment satisfying Assumption 1. For a subset $S \subset V$, let $H = H[S]$ be the corresponding node-induced subgraph. If H has at most one cycle per connected component, then:*

$$J[\mathbf{x}^*; G] \quad \geq \quad J\big[(\mathbf{x}_S; \mathbf{x}^*_{S^c}); G \big] \tag{29}$$

where S^c denotes the complement $V \backslash S$; and $\mathbf{x}_S \triangleq \{ \, x_s \mid s \in S \, \}$ ranges over all possible configurations in $\mathcal{X}^{|S|}$.

Proof. See Appendix C. □

Figure 12 illustrates the difference between the set of node-induced subgraphs to which Corollary 1 applies, and the larger set of subgraphs to which Theorem 3 applies. Shown in Figure 12(a) is a graph with cycles, in this case a 5×5 grid. Panel (b) shows a particular node-induced subgraph; nodes in gray correspond to those *not* in the subgraph. Both Corollary 1 and Theorem 3 apply to this subgraph. Panels (c) and (d) show two subgraphs to which Theorem 3 applies but Corollary 1 does not. Each subgraph consists of a spanning tree plus a single cycle. Note that the vertex sets of each of these subgraphs are the full set V, but their edge sets do not include every edge. Therefore, they are not node-induced subgraphs, meaning that Corollary 1 no longer applies.

Secondly, on the basis of Theorem 3, we can establish directly a result that was first proved, using alternative methods, by Weiss (2000); this result is also a corollary of the Freeman and Weiss (2001) result from above.

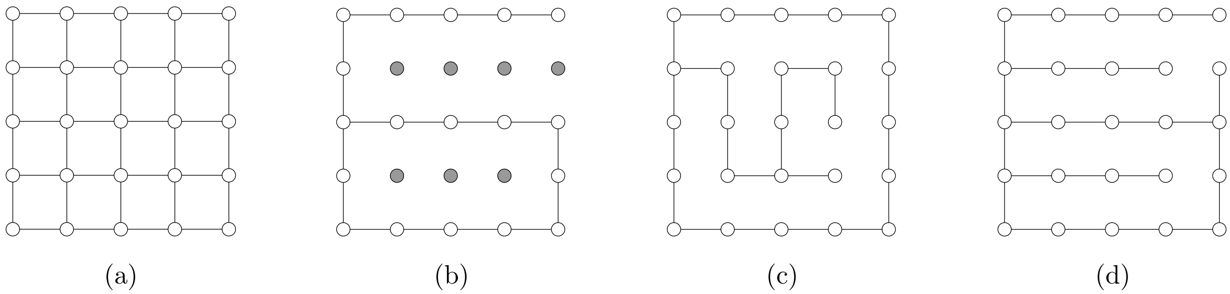


Figure 12. Difference between node-induced subgraphs and general subgraphs. (a) Original graph is a 5×5 grid. (b) A node-induced subgraph with a single cycle; nodes in gray are *not* part of the subgraph. (c) A spanning tree plus a single cycle. (d) A second spanning tree plus a single cycle. Neither of the subgraphs in (c) or (d) are node-induced.

Corollary 2 (Exactness on single cycles). *Under Assumption 1, the max-product assignment is exact for a positive distribution on a graph containing at most one cycle.*

4.5 Error bounds and consequences

In an integer programming problem, one way to measure the error between the optimal assignment $\widehat{\mathbf{x}}_{\text{MAP}}$ and a trial assignment $\mathbf{x}^*$ is in terms of the difference between the optimal cost $J(\widehat{\mathbf{x}}_{\text{MAP}}; G)$ and the cost $J(\mathbf{x}^*; G)$ associated with the approximation. In this section, we show that our analysis enables us to give bounds on this error in the max-product assignment in a straight-forward manner. In addition, these bounds allow us to prove that the max-product assignment $\mathbf{x}^*$ either agrees with a MAP configuration, or differs from any MAP configuration by a Hamming distance larger than a linear function of the graph girth.

Let $\widehat{\mathbf{x}}_{\text{MAP}} = \{ \widehat{x}_s \mid s \in V\}$ denote a MAP assignment, and let $\mathbf{x}^* = \{ x_s^* \mid s \in V\}$ denote the max-product assignment. We define the following quantity to assess the discrepancy between $\widehat{\mathbf{x}}_{\text{MAP}}$ and $\mathbf{x}^*$ on edge $(s,t) \in E$:

$$\Delta_{st}(\widehat{\mathbf{x}}_{\text{MAP}}, \mathbf{x}^*) \quad \triangleq \quad \log \frac{T^*_{st}(\widehat{x}_s, \widehat{x}_t)}{T^*_s(\widehat{x}_s)\, T^*_t(\widehat{x}_t)} - \log \frac{T^*_{st}(x^*_s, x^*_t)}{T^*_s(x^*_s)\, T^*_t(x^*_t)}$$

With this notation, we have:

Theorem 4 (Error bounds). *Let $H = (V, E(H))$ be a spanning subgraph of G that has at most one cycle per connected component. Then the difference between the cost of the optimal MAP assignment $\widehat{\mathbf{x}}_{\text{MAP}}$ and the max-product assignment $\mathbf{x}^*$ on the full graph G is bounded above as follows:*

$$\begin{aligned} J(\widehat{\mathbf{x}}_{\text{MAP}}; G) - J(\mathbf{x}^*; G) \quad &\leq \quad \sum_{(s,t) \in E \backslash E(H)} \Delta_{st}(\widehat{\mathbf{x}}_{\text{MAP}}, \mathbf{x}^*) && \text{(30a)} \\ &\leq \quad \max_{\mathbf{x}' \in \mathcal{X}^N} \sum_{(s,t) \in E \backslash E(H)} \Delta_{st}(\mathbf{x}', \mathbf{x}^*) && \text{(30b)} \end{aligned}$$

Proof. To obtain an upper bound on $J(\widehat{\mathbf{x}}_{\text{MAP}}; G) - J(\mathbf{x}^*; G)$, it is equivalent to upper bound the difference $F(\widehat{\mathbf{x}}_{\text{MAP}}; \mathbf{T}^*; G) - F(\mathbf{x}^*; \mathbf{T}^*; G)$. Any constants, which are independent of $\mathbf{x}$, vanish after taking the difference. Let H be a spanning subgraph of G with at most one cycle. Then we write:

$$\begin{aligned} &F(\widehat{\mathbf{x}}_{\text{MAP}}; \mathbf{T}^*; G) - F(\mathbf{x}^*; \mathbf{T}^*; G) \quad = \\ &\quad \Big[F(\widehat{\mathbf{x}}_{\text{MAP}}; \mathbf{T}^*; H) - F(\mathbf{x}^*; \mathbf{T}^*; H)\Big] + \sum_{(s,t) \in E \backslash E(H)} \left[\log \frac{T^*_{st}(\widehat{x}_s, \widehat{x}_t)}{T^*_s(\widehat{x}_s)\, T^*_t(\widehat{x}_t)} - \log \frac{T^*_{st}(x^*_s, x^*_t)}{T^*_s(x^*_s)\, T^*_t(x^*_t)} \right] \\ &\qquad \leq \sum_{(s,t) \in E \backslash E(H)} \left[\log \frac{T^*_{st}(\widehat{x}_s, \widehat{x}_t)}{T^*_s(\widehat{x}_s)\, T^*_t(\widehat{x}_t)} - \log \frac{T^*_{st}(x^*_s, x^*_t)}{T^*_s(x^*_s)\, T^*_t(x^*_t)} \right] \end{aligned}$$

where we have used Theorem 3 to apply the bound $\Big[F(\widehat{\mathbf{x}}_{\text{MAP}}; \mathbf{T}^*; H) - F(\mathbf{x}^*; \mathbf{T}^*; H)\Big] \leq 0$. This establishes equation (30a), from which equation (30b) follows immediately. $\square$

4.5.1 Practical considerations

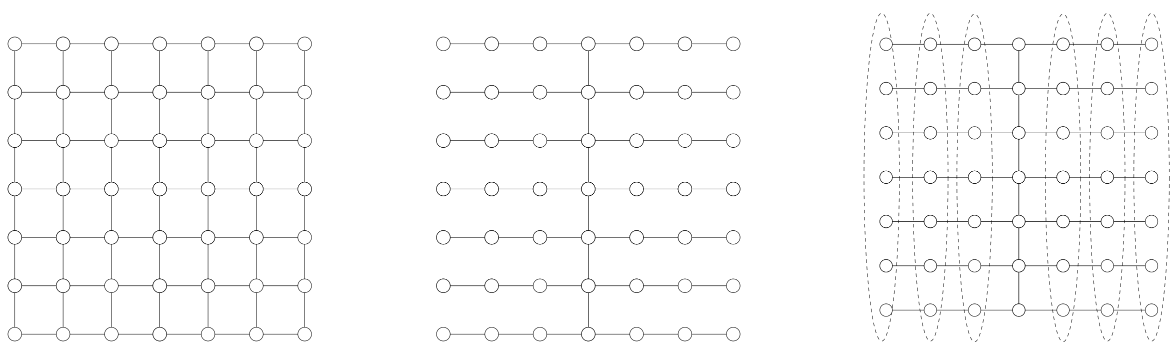
For Theorem 4 to be practically useful, we need to be able to compute the upper bounds. Computing the bound in equation (30a) is as difficult as solving the original integer program (IP), since it involves the unknown MAP configuration $\widehat{\mathbf{x}}_{\text{MAP}}$ itself. Although the bound of equation (30b) does not involve the unknown MAP configuration, computing it requires solving a different IP. The complexity of this IP depends on both the graph structure, and the choice of spanning subgraph H. For a very dense graph, a subgraph H with at most one cycle can cover only a small fraction of the edges, in which case it is likely that equation (30b) will be expensive to compute. However, it is always possible to compute a weaker upper bound by splitting the optimization in equation (30b) into several smaller and hence tractable pieces. To be more specific, for any disjoint partition $\{E^\beta\}$ of the edges in $E \backslash E(H)$, the following weaker upper bound is valid:

$$J(\widehat{\mathbf{x}}_{\text{MAP}}; G) - J(\mathbf{x}^*; G) \quad \leq \quad \sum_{\beta} \Big\{ \max_{\mathbf{x}' \in \mathcal{X}^N} \Big[\sum_{(s,t) \in E^\beta} \Delta_{st}(\mathbf{x}', \mathbf{x}^*) \Big] \Big\}$$

In the worst case, we could take the finest partition of $E \backslash E(H)$ possible — namely into individual edges.

There are other types of less densely-connected graphs for which the general MAP problem is still NP-hard, but yet a single spanning tree covers a significant fraction of the edges. In such cases, making a judicious choice of the subgraph H in Theorem 4 can ensure that the upper bound of equation (30b) is computable, as illustrated in the following example.

Example 4. Consider a $\sqrt{N} \times \sqrt{N}$ nearest-neighbor grid in 2D; it has N nodes and $2\sqrt{N}\,(\sqrt{N} - 1)$ edges. The case $N = 49$ is illustrated in Figure 13(a); one spanning tree for this graph is shown in panel (b). If the spanning



(a) (b) (c)

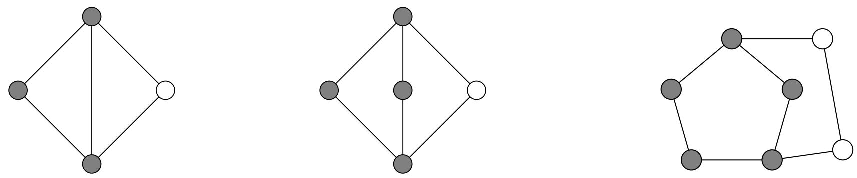
Figure 13. Example of an intractable graph where upper bound of Theorem 4 is efficiently computable. (a) Original graph is a 2D nearest-neighbor grid. (b) Spanning tree. (c) Upper bound requires separate optimizations over each chain of nodes contained within a dotted ellipse.

tree shown in (b) is used as the subgraph H in Theorem 4, then the optimization in equation (30b) naturally decouples into a set of separate optimization problems, one for each column of the grid, as shown in Figure 13(c). Each of these subproblems is equivalent to a MAP estimation problem on a chain-structured graph of length $\sqrt{N}$, and therefore can be solved exactly and efficiently in $\mathcal{O}(\sqrt{N})$ time. There are $\sqrt{N}$ such chain-structured problems to solve, one for each column of the grid, meaning that the overall complexity of computing the upper bound in equation (30b) is $\mathcal{O}(N)$. □

4.5.2 Theoretical consequences

From a theoretical perspective, the useful part of Theorem 4 is equation (30a), which links the max-product assignment $\mathbf{x}^*$ to a MAP assignment $\widehat{\mathbf{x}}_{\text{MAP}}$. In this section, we exploit this result to show that max-product assignment either achieves the MAP cost, or differs from the MAP assignment in very structured ways. For two strings $\mathbf{a}, \mathbf{b} \in \mathcal{X}^N$, we use $d_H(\mathbf{a}, \mathbf{b})$ to denote the Hamming distance:

$$d_H(\mathbf{a}, \mathbf{b}) \;=\; \sum_{i=1}^{N} \delta(a_i \neq b_i) \tag{31}$$



(a) (b) (c)

Figure 14. Graphs with two minimal cycles, and girths g from 3 to 5. (a) In the case $g = 3$, adding a single node is sufficient to generate multiple cycles of length 3. Note that in accordance with Lemma 6, $l(3) = 6 - (1 + 1) = 4$. (b) For a girth $g = 4$, we have $l(4) = 5 = 4 + 1$, so that one node is again sufficient. (c) When $g = 5$, then $l(5) = 10 - 2 - 1 = 7$, so that 2 additional nodes are required to form additional cycles of length 5.

I.e., the Hamming distance is a count of the number of elements in which $\mathbf{a}$ and $\mathbf{b}$ differ. Given a graph G, its *girth* is the number of edges in its shortest cycle. The main result of this section is to show that the max-product assignment is either equivalent to a MAP assignment, or differs from it by a Hamming distance that depends linearly on the girth. This result generalizes similar results of the same flavor by Freeman and Weiss (2001).

Before proving this result, we require the following lemmas:

Lemma 5. *Whenever* $d_H\big\{(\widehat{x}_s, \widehat{x}_t), (x^*_s, x^*_t)\big\} \leq 1$, *then* $\Delta_{st}(\widehat{\mathbf{x}}_{\text{MAP}}, \mathbf{x}^*) \leq 0$.

Proof. If $d_H\big\{(\widehat{x}_s, \widehat{x}_t), (x^*_s, x^*_t)\big\} = 0$, the statement is trivial. Otherwise, suppose without loss of generality that $\widehat{x}_s = x^*_s$ and $\widehat{x}_t \neq x^*_t$. Then we write:

$$\begin{aligned}
\Delta_{st}(\widehat{\mathbf{x}}_{\text{MAP}}, \mathbf{x}^*) &= \log \frac{T^*_{st}(x^*_s, \widehat{x}_t)}{T^*_s(x^*_s)\, T^*_t(\widehat{x}_t)} - \log \frac{T^*_{st}(x^*_s, x^*_t)}{T^*_s(x^*_s)\, T^*_t(x^*_t)} \\
&= \log \frac{T^*_{st}(x^*_s, \widehat{x}_t)}{T^*_t(\widehat{x}_t)} - \log \frac{T^*_{st}(x^*_s, x^*_t)}{T^*_t(x^*_t)} \\
&\leq 0
\end{aligned}$$

where we have used Lemma 4 to obtain the inequality. Thus, we have established that $\Delta_{st}(\widehat{\mathbf{x}}_{\text{MAP}}, \mathbf{x}^*) \leq 0$ for all $\mathbf{x}^*$ within Hamming distance one of $\widehat{\mathbf{x}}_{\text{MAP}}$. □

Lemma 6. *Given a graph G of girth g, define $l(g) = 2g - (\lfloor g/2 \rfloor + 1)$. Here $\lfloor g/2 \rfloor$ denotes the floor of $g/2$ (i.e., its integral part). Then any node induced subgraph $G[S]$ with a number of nodes $|S| \leq l(g) - 1$ has at most one cycle.*

Proof. See Appendix D. □

We now proceed to the main result of this section:

Corollary 3. *For a graph G with girth g, the max-product assignment $\mathbf{x}^*$ is either a MAP assignment (i.e., achieves the MAP value), or differs from any MAP assignment by a Hamming distance of at least $l(g) = 2g - \lfloor g/2 \rfloor - 1$.*

Proof. Suppose that the max-product assignment $\mathbf{x}^*$ is not equal to a MAP assignment; moreover, say it differs from a MAP assignment $\widehat{\mathbf{x}}_{\text{MAP}}$ in a subset of vertices S with $|S| \leq l(g) - 1$. By Lemma 6, the corresponding node-induced subgraph $G[S]$ can have at most one cycle. Therefore, we can augment $G[S]$ to a spanning subgraph $H = (V, E(H))$ that also contains at most one cycle, and such that $E(H)$ contains the edge subset $E[S] = \{(u_i, u_j) \in E \mid u_i, u_j \in S\}$. Applying equation (30a) from Theorem 4 to this subgraph, we obtain:

$$0 \;\leq\; J(\widehat{\mathbf{x}}_{\text{MAP}}; G) - J(\mathbf{x}^*; G) \;\leq\; \sum_{(s,t) \in E \backslash E(H)} \Delta_{st}(\widehat{\mathbf{x}}_{\text{MAP}}, \mathbf{x}^*) \tag{32}$$

By construction, no edge (s, t) in $E \backslash E(H)$ is of the form (u_i, u_j) for some $u_i, u_j \in S$. Therefore, if $\mathbf{x}^*$ differs from $\widehat{\mathbf{x}}_{\text{MAP}}$ only in elements within the set $\{x^*_u \mid u \in S\}$, then $d_H\big\{(\widehat{x}_s, \widehat{x}_t), (x^*_s, x^*_t)\big\} \leq 1$ for all $(s, t) \in E \backslash E(H)$. In this case, applying Lemma 5 to equation (32) yields that $J(\widehat{\mathbf{x}}_{\text{MAP}}; G) = J(\mathbf{x}^*; G)$, so that $\mathbf{x}^*$ achieves the

MAP value. Otherwise, there must be at least one vertex $s \in V \backslash S$ (i.e., in addition to those in S) for which $x_s^* \neq \widehat{x}_s$. Therefore, $\mathbf{x}^*$ must differ from $\widehat{\mathbf{x}}_{\mathrm{MAP}}$ by Hamming distance at least $|S|+1$. If $|S| = l(g)-1$, then we are done; otherwise, we simply augment S by adding the additional vertex s for which $x_s^* \neq \widehat{x}_s$, and apply the same argument to this augmented set (using the fact that $|S \cup \{s\}| = |S|+1 \leq l(g)-1$.) This same procedure can be applied until $|S| = l(g)-1$, at which point we will have established that $\mathbf{x}^*$ is either equal to a MAP assignment, or differs from any MAP assignment by Hamming distance at least $l(g)$. $\square$

The Hamming distance bound of Corollary 3 is tight for $g = 3$, in the sense that there exist graphs for which the bound is met with equality. For instance, consider the 4-node graph shown in Figure 14(a); it has girth $g = 3$, and $l(3) = 4$. There exist potentials for this 4-node graph for which the MAP assignment $\widehat{\mathbf{x}}_{\mathrm{MAP}} = [1 \;\; 1 \;\; 1 \;\; 1]$, whereas the max-product assignment is $\mathbf{x}^* = [0 \;\; 0 \;\; 0 \;\; 0]$, so that $d_H\{\widehat{\mathbf{x}}_{\mathrm{MAP}}, \mathbf{x}^*\} = l(3)$.

5 Generalizations of max-product

Several researchers have proposed extensions to the sum-product or belief propagation algorithm that entail operating over higher order cliques. Such extensions include generalized belief propagation (GBP) and Kikuchi approximations (Yedidia et al., 2001), as well as expectation-propagation updates (Minka, 2001). It is natural, then, to consider the analogs of such generalizations for the max-product algorithm. As in the case of belief propagation, there are at least two reasons for doing so.

First of all, the usual max-product algorithm applies to a Markov random field (MRF) with pairwise maximal cliques. Of course, any MRF with higher order cliques can, in principle, be converted to a pairwise MRF defined on an augmented graph (see Freeman and Weiss, 2001). Although the usual max-product algorithm can be applied directly to this modified graph, an alternative approach (described here) is to modify the max-product algorithm itself to handle higher-order cliques in a direct manner.

Secondly, even for an MRF with pairwise maximal cliques, there may be good reasons to consider larger groups of nodes over which to update — in particular, to obtain better approximations to the true MAP estimate. One way to construct such groupings is by adding edges to the graph, thereby forming an augmented graph with higher-order cliques. In fact, in the limit of adding enough edges to the graph to triangulate it, performing reparameterization over the maximal cliques of the triangulated graph is equivalent to applying the junction tree algorithm (Cowell et al., 1999; Dawid, 1992), which yields the exact max-marginals and MAP estimate. However, for many graphs, the process of triangulation leads to an explosion in the clique size, which means that the practical application of this junction tree algorithm is limited. Given the prohibitive cost of the junction tree algorithm, it is reasonable to consider partial triangulations of graphs. Such structures arise in certain extensions to BP, such as the GBP updates of Yedidia et al. (2001), which can be formulated as minimizing a cost function (e.g., Kikuchi free energy) that is defined by a partially triangulated graph.

To date, the counterparts of these extensions have not been explored in the context of MAP estimation. Accordingly, this section is devoted to the development and analysis of generalized max-product updates. These updates involve reparameterizing distributions that are defined by compatibility functions over higher-order cliques. One way to discuss such updates is in terms of aggregated graphs, where new vertices are defined by aggregating sets of nodes of the original graph into clusters (e.g., as in Kikuchi approximations (Yedidia et al., 2001)). The route that we pursue here is different, in that we formulate generalized max-product updates within the formalism of hypergraphs. Of particular importance to our development are acyclic hypergraphs, otherwise known as hypertrees, which are essentially equivalent to junction trees (Cowell et al., 1999). We refer the reader to Berge (1989) and Bodlaender (1993) for more details on hypergraphs and hypertrees. Based on this set-up, we then introduce generalizations of the max-product algorithm that entail performing reparameterization over hypertrees. Finally, we discuss how our results on the ordinary max-product algorithm have analogs for these generalizations.

5.1 Hypergraphs

Hypergraphs represent a natural generalization of graphs. In particular, a hypergraph $G_{\mathrm{HYP}} = (V, E)$ consists of a vertex set $V = \{1, \ldots, N\}$, and a set of hyperedges E, where each *hyperedge* h is a particular subset of V (i.e., an element of the power set of V). The set of hyperedges can be viewed naturally as a partially-ordered set, where the partial ordering is specified by inclusion. More details on hypergraphs can be found in Berge (1989), whereas Stanley (1997) provides more information on partially-ordered sets (also known as posets).

Given two hyperedges g and h, one of three possibilities can hold:

(a) the hyperedge g is contained within h, in which case we write $g < h$.

(b) if h is contained within g, then we write $h < g$.

(c) finally, if neither containment relation holds, then g and h are incomparable.

We say that a hyperedge is *maximal* if it is not contained within any other hyperedge.

With these definitions, an ordinary graph is a special case of a hypergraph, in which each maximal hyperedge consists of a pair of vertices (i.e., an ordinary edge of the graph). Note that for hypergraphs (unlike graphs), the set of hyperedges may include (a subset of) the individual vertices.

A convenient graphical representation of a hypergraph is in terms of a diagram of its hyperedges, with (directed) edges representing the inclusion relations. Diagrams of this nature have been used by Yedidia et al. (2001), who refer to them as *region graphs*; other researchers (McEliece and Yildirim, 2002; Pakzad and Anantharam, 2002) have used the term *Hasse diagram* from poset terminology (Stanley, 1997). Figure 15 provides some simple

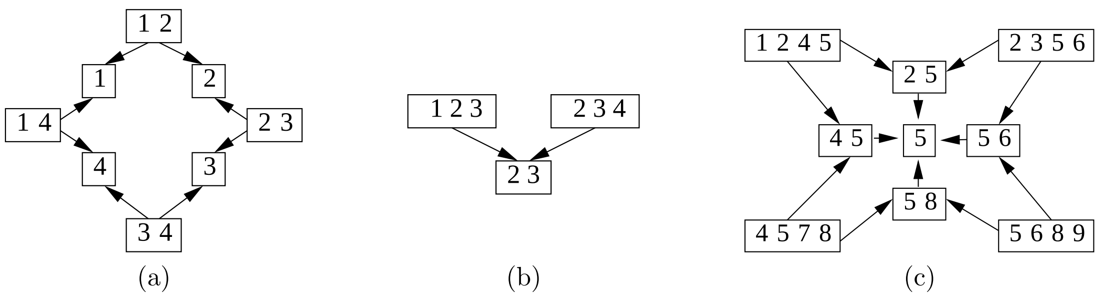


Figure 15. Graphical representations of hypergraphs. Subsets of nodes corresponding to hyperedges are shown in rectangles, whereas the arrows represent inclusion relations among hyperedges. (a) An ordinary single cycle graph represented as a hypergraph. (b) A simple hypergraph. (c) A more complex hypergraph.

graphical illustrations of hypergraphs. As a special case, any ordinary graph can be drawn as a hypergraph; in particular, panel (a) shows the hypergraph representation of a single cycle on four nodes. Panel (b) shows a hypergraph that is not equivalent to an ordinary graph, consisting of two hyperedges of size three joined by their intersection of size two. Shown in panel (c) is a more complex hypergraph, to which we will return in the sequel.

Given any hyperedge h, we define the sets of its *descendants* and *ancestors* in the following way:

$$\mathcal{D}(h) \quad = \quad \{g \in E \mid g < h\} \tag{33a}$$

$$\mathcal{A}(h) \quad = \quad \{g \in E \mid g > h\} \tag{33b}$$

For example, given the hyperedge $h = (1245)$ in the hypergraph in Figure 15(c), we have $\mathcal{A}(h) = \emptyset$ and $\mathcal{D}(h) = \{(25), (45), (5)\}$. We use the notation $\mathcal{D}^+(h)$ and $\mathcal{A}^+(h)$ as shorthand for the sets $\mathcal{D}(h) \cup h$ and $\mathcal{A}(h) \cup h$ respectively.

5.2 Junction trees and hypertrees

Of particular importance are acyclic hypergraphs, which are also known as hypertrees. In order to define these objects, we require the notions of tree decomposition and running intersection, which are well-known in the context of junction trees. More background on junction trees and running intersection can be found in Lauritzen (1996) or Cowell et al. (1999). Given a hypergraph G_{HYP}, a *tree decomposition* is an acyclic graph in which the nodes are formed by the maximal hyperedges of G_{HYP}. Any intersection $g \cap h$ of two maximal hyperedges that are adjacent in the tree is known as a *separator set*. The tree decomposition has the *running intersection property* if for any two nodes g and h in the tree, all nodes on the unique path joining them contain the intersection $g \cap h$. A hypergraph is *acyclic* if it possesses a tree decomposition with the running intersection property. The *width* of an acyclic hypergraph is the size of the largest hyperedge minus one; we use the term *k-hypertree* to mean a singly-connected acyclic hypergraph of width k.

As a simple illustration, a spanning tree of an ordinary graph is a 1-hypertree, because its maximal hyperedges (i.e., ordinary edges) all have size two. As a second example, the hypergraph of Figure 15(b) has maximal hyperedges of size three. This hypergraph is acyclic with width two, since it is in direct correspondence with the junction tree on the two maximal hyperedges and the single separator set (23). Figure 16 shows a more subtle

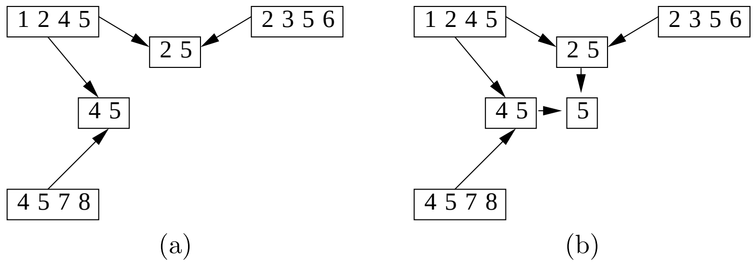


Figure 16. Two different graphical representations of the same underlying hypertree. (a) This diagram clearly corresponds to an acyclic hypergraph. (b) This representation seems different, but in fact corresponds to the same hypertree. Hence hypertrees cannot be identified simply by the absence (or presence) of cycles in poset diagrams.

example. Again, it is easy to see that the hypergraph in panel (a) is in direct correspondence with a junction tree, and hence is a width three hypertree. At first glance, the hypergraph illustrated in panel (b) might appear different, but in fact, it corresponds to the same hypertree. From the junction tree view, the addition of the extra hyperedge (5) and links to (25) and (45) is superfluous. This issue will be clarified in Example 5(c).

5.3 Hypergraph-structured distributions

We now consider the role of hypergraphs and hypertrees in specifying particular factorizations of distributions. Given a hypergraph G_{HYP}, we consider a distribution specified as a product compatibility functions over hyperedges:

$$p(\mathbf{x}) \quad = \quad \frac{1}{Z} \prod_{h \in E} \psi_h(\mathbf{x}_h) \tag{34}$$

If this hypergraph is acyclic (and hence can be associated with a junction tree), then the well-known junction tree theorem (Cowell et al., 1999) guarantees that p can be factored as a product of max-marginals on maximal cliques divided by a product of max-marginals on separator sets in the junction tree.

Here we describe an alternative but equivalent factorization in terms of the hypertree itself, which will be more convenient for our purposes. First of all, given a hyperedge $h \in E$, we define the associated max-marginal over the subset of variables $\mathbf{x}_h = \{x_s \mid s \in h\}$:

$$P_h(\mathbf{x}_h) \quad = \quad \max_{\{\mathbf{x}' \mid \mathbf{x}'_h = \mathbf{x}_h\}} p(\mathbf{x}') \tag{35}$$

Next, we use these max-marginals to define a function φ_h as follows:

$$\varphi_h(\mathbf{x}) \quad \triangleq \quad \frac{P_h(\mathbf{x}_h)}{\prod_{g \in \mathcal{D}(h)} \varphi_g(\mathbf{x}_g)} \tag{36}$$

where $\mathcal{D}(h)$, as defined in equation (33a), is the set of descendants of h. In terms of these quantities, the hypertree factorization itself is very simple:

$$p(\mathbf{x}) \quad = \quad \frac{1}{Z} \prod_{h \in E} \varphi_h(\mathbf{x}_h) \tag{37}$$

In this formulation, we have explicitly included a normalization constant Z, which may or may not be necessary, depending on how the max-marginals are normalized individually.

This factorization is, in fact, equivalent to the familiar junction tree representation:

$$p(\mathbf{x}) \quad = \quad \frac{1}{Z} \frac{\prod_{h \in E_{\max}} P_h(\mathbf{x}_h)}{\prod_{g \in E_{\mathrm{sep}}} \left[P_g(\mathbf{x}_g)\right]^{d(g)-1}} \tag{38}$$

In this equation, $E_{\max}$ denotes the set of maximal hyperedges, E_{sep} denotes the set of separator sets in a tree decomposition, and $d(g)$ denotes the number of maximal hyperedges that contain the separator set g.

We illustrate the hypertree factorization and its equivalence to the junction tree representation with a few examples:

Example 5 (Hypertree factorization).
(a) First suppose that the hypertree is an ordinary tree, in which case the hyperedge set consists of the union of the vertex set with the (ordinary) edge set. For any vertex s, we have $\varphi_s(x_s) = P_s(x_s)$, whereas for any edge (s, t) we have $\varphi_{st}(x_s, x_t) = P_{st}(x_s, x_t)/[P_s(x_s)\, P_t(x_t)]$. Therefore, in this special case, equation (37) reduces to the tree factorization in equation (7).

(b) Consider the hypergraph illustrated in Figure 15(b), specified by the hyperedges $\{(123), (234), (23)\}$. By inspection it is acyclic, so that we can apply the factorization of equation (37) (omitting the dependence of $\mathbf{x}$ for notational simplicity):

$$\begin{aligned} p &= \varphi_{123} \;\; \varphi_{234} \;\; \varphi_{23} \\ &= \frac{P_{123}}{\varphi_{23}} \; \frac{P_{234}}{\varphi_{23}} \; \varphi_{23} \\ &= \frac{P_{123}\, P_{234}}{P_{23}} \end{aligned}$$

If we were to construct the junction tree for this simple graph, it would have maximal cliques $\{(123), (234)\}$ and a single separator set (23). Therefore, the expression that we obtained from the hypertree factorization (37) (after some re-arrangement) agrees with the junction tree representation that would be obtained from equation (38).

(c) Now consider the hypergraph specified by $\{(1245), (2356), (4578), (25), (45), (5)\}$, as illustrated in Figure 16(b). Earlier we claimed that this hypergraph is acyclic. By applying the hypertree factorization (where we again omit the explicit dependence on $\mathbf{x}$), we obtain:

$$\begin{aligned} p &= \frac{P_{1245}}{\varphi_{25}\varphi_{45}\varphi_5} \; \frac{P_{2356}}{\varphi_{25}\varphi_5} \; \frac{P_{4578}}{\varphi_{45}\varphi_5} \;\; \varphi_{25} \;\; \varphi_{45} \;\; \varphi_5 \\ &= \frac{P_{1245}}{\frac{P_{25}}{P_5}\frac{P_{45}}{P_5}P_5} \; \frac{P_{2356}}{\frac{P_{25}}{P_5}P_5} \; \frac{P_{4578}}{\frac{P_{45}}{P_5}P_5} \;\; \frac{P_{25}}{P_5} \; \frac{P_{45}}{P_5} \; P_5 \\ &= \frac{P_{1245}\, P_{2356}\, P_{4578}}{P_{25}\, P_{45}} \end{aligned} \tag{39}$$

Again, this expression agrees with what would be obtained from the junction tree formula (38). Moreover, observe that we would obtain the same result if we applied the hypertree factorization to Figure 16(a) (which is more obviously a hypertree). Therefore, the representations in Figures 16(a) and (b), though ostensibly different, are effectively equivalent. □

5.4 Hypertree reparameterization max-product

We now have the necessary machinery to develop generalizations of the max-product algorithm. Our starting point is a hypergraph G_{HYP}, and a distribution $p(\mathbf{x})$ that is formed as the product of compatibility functions over its hyperedges. The underlying assumption is that the size of the largest hyperedge (say $k+1$) is sufficiently small such that it is feasible to perform exact computations on hypertrees of width k.

We then consider a sequence of hypertree reparameterization updates, entirely analogous to those of the tree-based max-product updates of Algorithm 1. Specifically, consider a collection of hypertrees $\{\mathcal{T}^0, \ldots \mathcal{T}^{L-1}\}$ that covers the hypergraph; that is, each hyperedge belongs to at least one hypertree. At any given iteration, we pick some hypertree $\mathcal{T}^i$ from this set, and then split the distribution $p(\mathbf{x})$ into the product of a *hypertree component* $p^i(\mathbf{x})$, consisting of all those compatibility functions on hyperedges in $\mathcal{T}^i$, and a residual component $r^i(\mathbf{x})$, consisting of those terms on hyperedges not in the hypertree. We then compute the exact max-marginals for the hypertree, and use them to reparameterize the hypertree distribution $p^i(\mathbf{x})$. The computation of the hypertree max-marginals can be carried out by an exact algorithm applied to the hypertree (e.g., in its junction tree representation).

More formally, as with tree-based max-product, these updates can be described as a sequence of functional updates on a collection $\mathbf{T} = \{T_h \mid h \in E\}$ of pseudo-max-marginals on the hyperedges of the hypergraph. Given the pseudo-max-marginal T_h and any hyperedge g that is covered by h, we define:

$$T_{g<h}(\mathbf{x}_g) = \max_{\{\, \mathbf{x}'_h \,|\, \mathbf{x}'_g = \mathbf{x}_g \,\}} T_h(\mathbf{x}'_h) \tag{40}$$

That is, $T_{g<h}$ is the pseudo-max-marginal on the hyperedge g induced by T_h. For the moment, we do not assume that the collection of pseudo-max-marginals is fully consistent. For example, even if both h and h' cover the same hyperedge g, the pseudo-max-marginal $T_{g<h}$ need not be equal to $T_{g<h'}$; moreover, neither of these quantities has to be equal to the pseudo-max-marginal T_g defined for hyperedge g. Ultimately, however, at fixed points of the reparameterization updates, *all* of these consistency conditions will hold.

Now for each hyperedge h with pseudo-max-marginal T_h, we define the following quantity:

$$\varphi_h(\mathbf{x}_h; T_h) \quad = \quad \frac{T_h(\mathbf{x}_h)}{\prod_{g \in \mathcal{D}(h)} \varphi_g(\mathbf{x}_g; T_{g<h})} \tag{41}$$

The quantity $\varphi_h(\mathbf{x}_h; T_h)$ is very similar to the earlier $\varphi_h(\mathbf{x}_h)$ defined in equation (36), except that it should be viewed as a function of both $\mathbf{x}_h$ and the pseudo-max-marginal T_h. In addition, note that for any hyperedge $g < h$, it is the induced marginal $T_{g<h}$ that is used in the denominator of the definition of equation (41).

With these definition, the collection $\mathbf{T}$ specifies an alternative parameterization of the original distribution $p(\mathbf{x})$ on the full hypergraph (which includes cycles in general) as follows:

$$p(\mathbf{x}; \mathbf{T}) \quad = \quad \frac{1}{Z(\mathbf{T})} \prod_{h \in E} \varphi_h(\mathbf{x}_h; T_h) \tag{42}$$

Example 6. (a) To illustrate these definitions, first consider the hypergraph representation of an ordinary graph with pairwise maximal cliques, so that the hyperedge set consists of the union of the vertex set and (ordinary) edge set. Suppose that we are given a collection of pseudo-max-marginals $\mathbf{T} = \{T_s, T_{st}\}$. Then applying the definition in equation (41) yields, for a maximal hyperedge $h = (st)$ and size one hyperedge $g = (s)$ respectively, the following expressions:

$$\varphi_{st}(\,(x_s, x_t);\, T_{st}) \quad = \quad \frac{T_{st}(x_s, x_t)}{\max_{x'_s} T_{st}(x'_s, x_t)\ \max_{x'_t} T_{st}(x_s, x'_t)} \tag{43a}$$

$$\varphi_s(\,x_s;\, T_s) \quad = \quad T_s(x_s) \tag{43b}$$

Consequently, in this special case, the parameterization of equation (42) reduces to the earlier "tree-like" decomposition in equation (16).

(b) Now consider the hypergraph in Figure 15(c). As an illustration, let us compute $\varphi_{1245}(\mathbf{x}; \mathbf{T})$ by applying equation (41). For simplicity in notation, we omit the dependence on $\mathbf{x}$, and we assume that $\mathbf{T}$ is locally consistent (so that we can ignore the distinction between $T_{g<h}$ and T_h). By doing so, we obtain:

$$\begin{aligned} \varphi_{1245} \quad &= \quad \frac{T_{1245}}{\varphi_{25}\ \varphi_{45}\ \varphi_5} \\ &= \quad \frac{T_{1245}\ T_5}{T_{25}\ T_{45}} \end{aligned}$$

□

In order to initialize the reparameterization updates, it is convenient to assume that $p(\mathbf{x})$ is defined by a product of compatibility functions $\{\psi_h\}$ over only maximal hyperedges h. There is no loss of generality in making this assumption, since we can always absorb any compatibility function on a non-maximal hyperedge into some maximal hyperedge that contains it. Given a hypertree $\mathcal{T}^i$ with hypertree edge set $E(\mathcal{T}^i)$, the hypertree component $p^i(\mathbf{x}; \mathbf{T}^n)$ and residual component $r^i(\mathbf{x}; \mathbf{T}^n)$ are given, respectively, by the expressions:

$$p^i(\mathbf{x}; \mathbf{T}^n) \quad \propto \quad \prod_{h \in E(\mathcal{T}^i)} \varphi_h(\mathbf{x}_h; T^n_h) \tag{44a}$$

$$r^i(\mathbf{x}; \mathbf{T}^n) \quad \propto \quad \prod_{h \in E \backslash E(\mathcal{T}^i)} \varphi_h(\mathbf{x}_h; T^n_h) \tag{44b}$$

With this set-up, the hypertree max-product algorithm consists of the following steps:

Algorithm 2 (Reparameterization max-product on hypertrees).

1. At iteration $n = 0$, initialize $\varphi_h(\mathbf{x}; T_h^0)$ as follows:

$$\varphi_h(\mathbf{x}; T_h^0) \quad = \quad \begin{cases} \psi_h(\mathbf{x}_h) & \text{if } h \text{ is maximal} \\ 1 & \text{otherwise} \end{cases} \tag{45}$$

2. At iterations $n = 1, 2, \ldots$, choose hypertree $\mathcal{T}^{i(n)}$ with hyperedge set $E^{i(n)}$. Update pseudo-max-marginals as follows:

$$T_h^{n+1}(\mathbf{x}_h) \quad = \quad \begin{cases} \max_{\{\mathbf{x}' \mid \mathbf{x}'_h = \mathbf{x}_h\}} p^{i(n)}(\mathbf{x}'; \mathbf{T}^n) & \text{if } h \in E^{i(n)} \\ T_h^n(\mathbf{x}_h) & \text{if } h \in E \backslash E^{i(n)} \end{cases} \tag{46}$$

Remarks: As a natural generalization of the tree reparameterization form of ordinary max-product given in Algorithm 1, similar remarks are applicable to Algorithm 2. First of all, it is clear that the initialization[4] given in Step 1 ensures that $p(\mathbf{x}; \mathbf{T}^0)$, as defined in equation (42), is equivalent to the original distribution $p(\mathbf{x})$. Computing max-marginals on a given hypertree, as stipulated by Step 2, can be performed by any exact algorithm applied to the hypertree.

If the updates of Algorithm 2 converge to a fixed point $\mathbf{T}^*$, then we are guaranteed that for any of the hypertrees $\mathcal{T}^i$, the elements $\{T_h^* \mid h \in E^i\ \}$ are a consistent set of max-marginals for the hypertree distribution $p^i(\mathbf{x}; \mathbf{T}^*)$. Therefore, given any hyperedge $h \in E^i$ and any other hyperedge $g < h$, the following local consistency condition must hold:

$$T_g^*(\mathbf{x}_g) \quad = \quad \kappa \, T_{g<h}^*(\mathbf{x}_g) \; \equiv \; \kappa \max_{\{\mathbf{x}'_h \mid \mathbf{x}'_g = \mathbf{x}_g\ \}} T_h^*(\mathbf{x}'_h) \tag{47}$$

Since every hyperedge h belongs to at least one of the hypertrees $\mathcal{T}^0, \ldots, \mathcal{T}^{L-1}$, we are guaranteed that equation (47) holds for every nested pair $g < h$ of hyperedges in the full hypergraph. Given any hypertree $\mathcal{T}$ with edge set $E(\mathcal{T})$, let us isolate those terms corresponding to edges in the hypertree so as to form a hypertree-structured distribution:

$$p^{\mathcal{T}}(\mathbf{x}; \mathbf{T}^*) \quad \propto \quad \prod_{h \in E(\mathcal{T})} \varphi_h(\mathbf{x}; T_h^*)$$

By the junction tree theorem (Cowell et al., 1999), the local consistency of $\{T_h^* \mid h \in E(\mathcal{T})\}$ on all nested sets of hyperedges implies that the pseudo-max-marginals $\{T_h^* \mid h \in E(\mathcal{T})\ \}$ are equivalent to the exact max-marginals for the hypertree-structured distribution $p^{\mathcal{T}}(\mathbf{x}; \mathbf{T}^*)$. The result that we have just established is the analog of Proposition 2(b), generalized to the case of hypertrees.

Moreover, it should be clear that our other results on the ordinary max-product can be generalized to the hypertree reparameterization updates of Algorithm 2:

1. All of the iterates $\mathbf{T}^n$, as well as any fixed point $\mathbf{T}^*$, simply specify a different parameterization of the distribution $p(\mathbf{x})$. That is, the distribution $p(\mathbf{x})$ is invariant, as stated in Proposition 2(a) for ordinary max-product.

2. As with our earlier analysis, we can define modified cost functions that include only those terms corresponding to a particular embedded hypergraph. We then can state and prove results about the hypertree max-product assignment $\mathbf{x}^*$, analogous to those of Theorem 3.

3. Finally, it is possible to derive upper bounds on the difference between the log probability of the generalized max-product assignment and the log probability of the MAP assignment, similar to those of Theorem 4.

[4]There is a subtle point here: the initialization of equation (45) does not directly specify $\mathbf{T}^0$, but rather the quantities $\varphi_h(\mathbf{x}; T_h^0)$ (which suffice to specify the parameterization of the distribution). Once each hyperedge has been updated at least once, there will be an explicitly defined T_h^n for all h. Prior to this point, the update of equation (46) should be interpreted in the following way: for any $h \in E \backslash E^{i(n)}$, we set $\varphi_h(\mathbf{x}_h; T_h^{n+1}) = \varphi_h(\mathbf{x}_h; T_h^n)$.

6 Discussion

We have developed a reparameterization framework for understanding and analyzing the max-product algorithm, as well as a rich class of its generalizations. We have shown how these algorithms can be interpreted as seeking a particular factorization of a graph-structured distribution in terms of pseudo-max-marginals over (hyper)edges. These pseudo-max-marginals are, in fact, the exact max-marginals for a tree-structured graph, but approximate for a general graph with cycles. The significance of this max-marginal representation is in facilitating solution of the MAP estimation problem.

The reparameterization viewpoint gives considerable insight into the nature of max-product fixed points, and the associated max-product assignment. Key results include the fact the original distribution is never altered by reparameterization updates, and that any max-marginal fixed point is guaranteed to be consistent on every tree of the graph. Our analysis also establishes the fact, not obvious for a graph with cycles, that fixed points of this nature always exist for positive distributions. We exploited the reparameterization perspective to characterize the max-product assignment, thereby generalizing earlier work (Freeman and Weiss, 2001; Weiss, 2000). We also derived computable upper bounds on the difference in the log probability of the MAP assignment, and that of the max-product assignment.

An open problem not addressed by our work is convergence of the max-product updates to one of the fixed points that are guaranteed to exist. For a graph with cycles, neither the parallel message-passing updates nor the tree-based update of Algorithm 1 are guaranteed to converge, and indeed (at least in coding applications) failure of convergence appears to be the dominant error mode (e.g., Benedetto et al., 1996; Weiss, 2000). Therefore, of interest are algorithms for finding fixed points with improved or guaranteed convergence. In earlier work on the sum-product algorithm (Wainwright et al., 2002), we found that more global tree-based updates, such as those of Algorithm 1, often have convergence properties superior to those of parallel message-passing updates. Preliminary experiments suggest that such tree-based updates may also lead to improved convergence for the max-product case. Another direction worth exploring is the applicability of known techniques (e.g., Ortega and Rheinboldt, 2000) for solving fixed point equations.

This paper, in conjunction with our previous work on the sum-product or belief propagation algorithm (Wainwright et al., 2002) establishes the notion of reparameterization as a unifying conceptual framework for analyzing a variety of algorithms for approximate inference on graphs with cycles. More generally, approximate dynamic-programming algorithms, like sum-product and max-product applied to graphs with cycles, have analogs for any commutative semi-ring (Verdú and Poor, 1987). In this context, it is interesting to speculate about analogs of our reparameterization results for semi-rings more exotic than the sum-product and max-product cases.

A Proof of Proposition 1

Since Lemma 1 shows that $p(\mathbf{x};\mathbf{T}^*) \equiv p(\mathbf{x})$, it is equivalent to prove that

$$T^*_{ab}(x_a, x_b) \;=\; \kappa \max_{\{\mathbf{x}' \mid (x'_a, x'_b) = (x_a, x_b)\}} p(\mathbf{x}';\mathbf{T}^*) \tag{48}$$

for each pair $(a,b) \in E$. Note that the equivalence of the pairwise max-marginals implies the equivalence of the single node max-marginals (i.e., $T^*_a = P^*_a$).

Given any tree $\mathcal{T}$, we can arbitrarily designate one node — say node r — as the *root*. Every other node $w \in V\backslash\{r\}$ then has a unique parent, which we denote $\mathrm{pa}(w)$. See Figure 5 of Section 3.1 for an illustration. Using the parent-child relations, we decompose the distribution $p(\mathbf{x};\mathbf{T}^*)$ as follows:

$$p(\mathbf{x};\mathbf{T}^*) \quad = \quad T^*_r(x_r) \prod_{w \in V\backslash\{r\},\; v \equiv \mathrm{pa}(w)} \frac{T^*_{vw}(x_v, x_w)}{T^*_v(x_v)} \tag{49}$$

where we have used $v \equiv \mathrm{pa}(w)$ as short-hand for the parent of node w.

If the tree $\mathcal{T}$ consists of a pair of nodes, then equation (48) is satisfied trivially. Otherwise, any tree on $N \geq 2$ nodes has at least two leaf nodes (Bollobás, 1998), so that we can perform a *leaf-stripping* procedure to prove equation (48) for the root node and one of its neighbors. Consider a leaf node s with parent t, as illustrated in Figure 5(c). From the decomposition in equation (49), the only part of $p(\mathbf{x};\mathbf{T}^*)$ that depends on x_s is the quantity $T^*_{st}(x_s, x_t)/T^*_t(x_t)$. From equation (15), we have:

$$\max_{x'_s} \frac{T^*_{st}(x'_s, x_t)}{T^*_t(x_t)} \quad = \quad \kappa\, \frac{T^*_t(x_t)}{T^*_t(x_t)} \quad = \quad \kappa$$

which is a constant for all $x_s \in \mathcal{X}$. Therefore, if we maximize $p(\mathbf{x}; \mathbf{T}^*)$ over x_s, then we obtain:

$$\begin{aligned}\max_{x'_s} p(\mathbf{x}; \mathbf{T}^*) &= T^*_r(x_r) \prod_{u \in V \backslash \{r,s\},\ v \equiv \mathrm{pa}(w)} \frac{T^*_{vw}(x_v, x_w)}{T^*_v(x_v)} \ \max_{x'_s} \frac{T^*_{st}(x'_s, x_t)}{T^*_t(x_t)} \\ &\propto T^*_r(x_r) \prod_{w \in V \backslash \{1,s\},\ v \equiv \mathrm{pa}(w)} \frac{T^*_{vw}(x_v, x_w)}{T^*_v(x_v)}\end{aligned}$$

Thus, maximizing over x_s eliminates node s, thereby leaving us with the tree $\mathcal{T}\backslash\{s\}$ with $N-1$ nodes. We can proceed in this manner until we reach the tree with a single edge — say (r, u) where u is a neighbor of r. At this point, we will have shown that

$$T^*_{ru}(x_r, x_u) \propto \max_{\{\mathbf{x}' \mid (x'_r, x'_u) = (x_r, x_u)\}} p(\mathbf{x}'; \mathbf{T}^*) \tag{50}$$

so that T^*_{ru} is equivalent to the max-marginal P_{ru}. By varying our choice of the leaf elimination ordering, we can arrange so that u is any of the neighbors of node r. Finally, since our choice of the root node was arbitrary, we can prove a statement of the form in equation (50) for any edge $(a, b) \in E$.

B Proof of Theorem 2

B.1 Proof of Lemma 2

It is equivalent to show that each $\mathcal{L}_{st}$ is continuous. The function $\mathcal{L}_{st}(\cdot; k)$ is a maximum over the finite collection (namely, $j \in \mathcal{X}$) of a linear function of the log messages $\boldsymbol{\zeta}$. Consequently, it suffices to show that the maximum over a finite collection is continuous. We first make note of the relation $\max\{a, b\} = \frac{1}{2}[\,(a+b) + |a-b|\,]$, so that the maximum of two numbers is continuous. This relation can be extended to maximizing over an arbitrary finite collection by the relation $\max\{a, b, c\} = \max\{\max\{a, b\}, c\}$.

B.2 Proof of Lemma 3

Let (s, t) be an arbitrary edge, and let i, j, k be arbitrary elements of $\mathcal{X}$. For compactness in notation, we define

$$\widetilde{\phi}_{st;s}(j, k) \triangleq \log \psi_{st}(j, k) + \log \psi_s(j)$$

Note that $|\widetilde{\phi}_{st;s}(j, k)| < \infty$ for all $j, k \in \mathcal{X}$ since we have assumed that the compatibility functions $\boldsymbol{\psi}$ were strictly positive.

Now for any $\boldsymbol{\zeta} \in \mathbb{R}^D$, we have

$$\begin{aligned}\mathcal{L}_{st}(\boldsymbol{\zeta}; k) &= \max_{j \in \mathcal{X}} \Big[\widetilde{\phi}_{st;s}(j, k) + \sum_{u \in \Gamma(s)/t} \zeta_{us}(j)\ \Big] \\ &= \max_{j \in \mathcal{X}} \Big[\widetilde{\phi}_{st;s}(j, i) + \sum_{u \in \Gamma(s)/t} \zeta_{us}(j)\ + \big[\widetilde{\phi}_{st;s}(j, k) - \widetilde{\phi}_{st;s}(j, i)\big]\ \Big] \\ &\leq \max_{j \in \mathcal{X}} \Big[\widetilde{\phi}_{st;s}(j, i) + \sum_{u \in \Gamma(s)/t} \zeta_{us}(j)\ \Big] + \max_{j \in \mathcal{X}} \Big[\widetilde{\phi}_{st;s}(j, k) - \widetilde{\phi}_{st;s}(j, i)\Big] \\ &= \mathcal{L}_{st}(\boldsymbol{\zeta}; i) + \max_{j \in \mathcal{X}} \Big\{\widetilde{\phi}_{st;s}(j, k) - \widetilde{\phi}_{st;s}(j, i)\Big\} \\ &\leq \mathcal{L}_{st}(\boldsymbol{\zeta}; i) + \max_{j \in \mathcal{X}} \Big|\widetilde{\phi}_{st;s}(j, k) - \widetilde{\phi}_{st;s}(j, i)\Big|\end{aligned}$$

By interchanging k and i, we obtain for all $\boldsymbol{\zeta} \in \mathbb{R}^D$:

$$\begin{aligned}|\mathcal{L}_{st}(\boldsymbol{\zeta}; k) - \mathcal{L}_{st}(\boldsymbol{\zeta}; i)| &\leq \max_{j \in \mathcal{X}} \Big|\widetilde{\phi}_{st;s}(j, k) - \widetilde{\phi}_{st;s}(j, i)\Big| \\ &\leq \max_{k,i \in \mathcal{X}} \max_{j \in \mathcal{X}} \Big|\widetilde{\phi}_{st;s}(j, k) - \widetilde{\phi}_{st;s}(j, i)\Big| \\ &\triangleq c_{st} < \infty\end{aligned}$$

To complete the proof of the lemma, we write:

$$\begin{aligned}
\|\mathcal{F}(\boldsymbol{\zeta})\|_\infty &= \max_{(s,t)\in E} \max_{k\in\mathcal{X}} \left|\mathcal{F}_{st}(\boldsymbol{\zeta};k)\right| \\
&= \max_{(s,t)\in E} \max_{k\in\mathcal{X}} \left|\ \mathcal{L}_{st}(\boldsymbol{\zeta};k) - \frac{1}{m}\sum_{i\in\mathcal{X}} \mathcal{L}_{st}(\boldsymbol{\zeta};i)\ \right| \\
&= \max_{(s,t)\in E} \max_{k\in\mathcal{X}} \left|\ \frac{1}{m}\sum_{i\in\mathcal{X}} \left[\mathcal{L}_{st}(\boldsymbol{\zeta};k) - \mathcal{L}_{st}(\boldsymbol{\zeta};i)\right]\ \right| \\
&\leq \max_{(s,t)\in E} \max_{k\in\mathcal{X}} \frac{1}{m}\sum_{i\in\mathcal{X}} \left|\mathcal{L}_{st}(\boldsymbol{\zeta};k) - \mathcal{L}_{st}(\boldsymbol{\zeta};i)\right| \\
&\leq \max_{(s,t)\in E} c_{st} \\
&\triangleq c \ < \ \infty
\end{aligned}$$

C Proof of Corollary 1

Let $S^c = V \backslash S$ denote the complement of S in the vertex set V. Then define a partition of the edge set into three disjoint subsets as follows:

$$\begin{aligned}
E(H[S]) &\triangleq \{\ (s,t)\in E \mid s,t\in S\ \} \\
E' &\triangleq \{\ (s,u)\in E \mid s\in S, u\in S^c\ \} \\
E(H[S^c]) &\triangleq \{\ (u,v)\in E \mid u,v\in S^c\ \}
\end{aligned}$$

In this notation, we have

$$F(\mathbf{x}^*;\mathbf{T}^*;H[S]) \;=\; \sum_{s\in S}\log T^*_s(x^*_s) + \sum_{(s,t)\in E(H[S])} \log\frac{T^*_{st}(x^*_s,x^*_t)}{T^*_s(x^*_s)T^*_t(x^*_t)}$$

We now express $F(\mathbf{x}^*;\mathbf{T}^*;G)$ in terms of $F(\mathbf{x}^*;\mathbf{T}^*;H[S])$ and some extra terms:

$$F(\mathbf{x}^*;\mathbf{T}^*;G) \;=\; F(\mathbf{x}^*;\mathbf{T}^*;H[S]) + \sum_{u\in S^c}\log T^*_u(x^*_u) + \sum_{(u,v)\in E(S^c)} \log\frac{T^*_{us}(x^*_u,x^*_s)}{T^*_u(x^*_u)T^*_s(x^*_s)} + \sum_{(s,u)\in E'} \log\frac{T^*_{su}(x^*_s,x^*_u)}{T^*_s(x^*_s)T^*_u(x^*_u)} \tag{51}$$

First of all, by Theorem 3, we have

$$F(\mathbf{x}^*;\mathbf{T}^*;H[S]) \;\geq\; F(\mathbf{x};\mathbf{T}^*;H[S]) \tag{52}$$

for all $\mathbf{x}\in\mathcal{X}$. Secondly, by Lemma 4, we have $\log\left[T^*_{su}(x^*_s,x^*_u)/T^*_s(x^*_s)\right] \;\geq\; \log\left[T^*_{su}(x_s,x^*_u)/T^*_s(x_s)\right]$ for all $x_s\in\mathcal{X}$. Subtracting $\log T^*_u(x^*_u)$ from both sides yields:

$$\log\frac{T^*_{su}(x^*_s,x^*_u)}{T^*_s(x^*_s)T^*_u(x^*_u)} \;\geq\; \log\frac{T^*_{su}(x_s,x^*_u)}{T^*_s(x_s)T^*_u(x^*_u)} \tag{53}$$

for all $(s,u)\in E'$, and for all $x_s\in\mathcal{X}$.

Finally, making use of equations (52) and (53) in equation (51), we have:

$$\begin{aligned}
F(\mathbf{x}^*;\mathbf{T}^*;G) &\geq F(\mathbf{x};\mathbf{T}^*;H[S]) + \sum_{u\in S^c}\log T^*_u(x^*_u) + \sum_{(u,v)\in E(S^c)} \log\frac{T^*_{us}(x^*_u,x^*_s)}{T^*_u(x^*_u)T^*_s(x^*_s)} + \sum_{(s,u)\in E'} \log\frac{T^*_{su}(x_s,x^*_u)}{T^*_s(x_s)T^*_u(x^*_u)} \\
&= F((\mathbf{x}_S;\mathbf{x}^*_{S^c});\mathbf{T}^*;G)
\end{aligned}$$

By the equivalence of F to J, we have proved equation (29).

D Proof of Lemma 6

We explicitly construct the smallest node-induced subgraph $G[S]$ with more than one cycle, and show that $|S| = l(g)$. We first choose some $U \subset V$ such that $G[U]$ is a subgraph consisting of a single cycle on $|U| = g$ nodes. We now add the minimal number of nodes to U so as to form a second cycle, recalling the fact that this cycle must have length at least g. In particular, given two nodes u_1, u_2 in the cycle $G[U]$, we add a set of extra nodes W so as to form a second path $P(W)$ joining u_1 and u_2. In this way, the node-induced subgraph $G[U \cup W]$ has two additional cycles; each corresponds to following $P(W)$ from u_1 to u_2, and then returning to u_1 via one of the two directions of the original cycle. See Figure 14 for an illustration. By choosing u_1 and u_2 appropriately, we can ensure that they are separated in $G[U]$ by a path formed of at most $\lfloor g/2 \rfloor$ edges, and $\lfloor g/2 \rfloor + 1$ nodes. Therefore, in order to form a second cycle with at least g edges (and nodes), we need to add at least $g - (\lfloor g/2 \rfloor + 1)$ additional nodes to U. The subgraph $G[S]$ constructed in this way has $l(g) = 2g - (\lfloor g/2 \rfloor + 1)$ nodes in total.

References

S. M. Aji, G. Horn, R. J. McEliece, and M. Xu. Iterative min-sum decoding of tail-biting codes. In *Proc. IEEE Information Theory Workshop, Killarney Ireland*, pages 68–69, June 1998.

S.M. Aji and R.J. McEliece. The generalized distributive law. *IEEE Trans. Info. Theory*, 46:325–343, March 2000.

F. Barahona. On the computational computational complexity of the Ising model. *Journal of Physics A: Mathematical and General*, 15:3241–3253, 1982.

S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara. Soft-output decoding algorithms in iterative decoding of turbo codes. Technical report, Jet Propulsion Laboratory, 1996.

C. Berge. *Graphs and hypergraphs*. North-Holland Publishing Company, Amsterdam, 1976.

C. Berge. *Hypergraphs*. North-Holland Publishing Company, Amsterdam, 1989.

D.P. Bertsekas. *Dynamic programming and stochastic control*, volume 1. Athena Scientific, Belmont, MA, 1995.

J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*, 48(3):259–279, 1986.

H. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–21, 1993.

B. Bollobás. *Modern graph theory*. Springer-Verlag, New York, 1998.

P. Brémaud. *Markov chains, Gibbs fields, Monte Carlo simulation, and queues*. Springer, 1991.

Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Numerical Mathematics and Scientific Computation. Oxford University Press, 1988.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probablistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer-Verlag, 1999.

A. P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.

G. D. Forney, Jr. The Viterbi algorithm. *Proc. IEEE*, 61:268–277, March 1973.

G. D. Forney, Jr., F. R. Kschischang, B. Marcus, and S. Tuncel. Iterative decoding of tail-biting trellises and connections with symbolic dynamics. In *Codes, systems and graphical models*, pages 239–264. Springer, 2001.

W. T. Freeman and E.C. Pasztor. Learning to estimate scenes from images. In *NIPS*, volume 11, 1999.

W. T. Freeman and Y. Weiss. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Trans. Info. Theory*, 47:736–744, 2001.

B. J. Frey and R. Koetter. Exact inference using the attenuated max-product algorithm. In *Advanced mean field methods: Theory and Practice*. MIT Press, 2000.

B. J. Frey, R. Koetter, and A. Vardy. Skewness and pseudocodewords in iterative decoding. In *International Symp. Info. Theory*, 1998.

A. Fridman. Topology-corrected belief revision in the Ising model. Presented at Snowbird Learning workshop, Snowbird, UT, April 2002.

R. G. Gallager. *Low-density parity check codes*. MIT Press, Cambridge, MA, 1963.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Pat. Anal. Mach. Intell.*, 6:721–741, 1984.

M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer-Verlag, Berlin, Germany, 1993.

G. Horn. *Iterative decoding and pseudocodewords*. PhD thesis, California Institute of Technology, 1999.

M. Jordan. *Learning in graphical models*. MIT Press, Cambridge, MA, 1999.

S. L. Lauritzen. *Graphical models*. Oxford University Press, Oxford, 1996.

R. J. McEliece and M. Yildirim. Belief propagation on partially ordered sets. In *Mathematical Theory of Systems and Networks*, August 2002.

T. P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT, January 2001.

D. Nilsson. An efficient algorithm for finding the M most probable configurations in a probabilistic expert system. *Statistics and Computing*, 8:159–173, 1998.

J. M. Ortega and W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Classics in applied mathematics. SIAM, New York, 2000.

P. Pakzad and V. Anantharam. Iterative algorithms and free energy minimization. In *CISS*, March 2002.

J. Pearl. *Probabilistic reasoning in intelligent systems*. Morgan Kaufman, San Mateo, 1988.

R. P. Stanley. *Enumerative Combinatorics*, volume 1. Cambridge University Press, Cambridge, UK, 1997.

S. Verdú and H. V. Poor. Abstract dynamic programming models under commutativity conditions. *SIAM J. Control and Optimization*, 25(4):990–1006, July 1987.

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Info. Theory*, IT-13:260–269, April 1967.

M. J. Wainwright. *Stochastic processes on graphs with cycles: geometric and variational approaches*. PhD thesis, MIT, January 2002.

M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree-based reparameterization analysis of belief propagation and its generalizations. *IEEE Trans. Info. Theory*, 2002. Appeared as LIDS Tech. report P-2510; May, 2001.

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

N. Wiberg. *Codes and decoding on general graphs*. PhD thesis, University of Linkoping, Sweden, 1996.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS 13*, pages 689–695. MIT Press, 2001.