Local Conditioning on Undirected Graphs

Matthew G. Reyes
mgreyes@umich.edu

David L. Neuhoff
neuhoff@umich.edu

EECS Department, University of Michigan

***Abstract*—This paper proposes Local Conditioning as a truly distributed exact version of Belief Propagation for cyclic undirected graphical models. It is shown how to derive explicit recursive updates for messages and beliefs that are truly distributed in the sense that messages are passed between individual nodes of the graph rather than between clustered nodes. Such a distributed algorithm is especially relevant for problems that require a distributed implementation, for example sensor networks. In order to compare its complexity and ease of implementation with a clustered version of Belief Propagation, we illustrate both in a Min-Sum block interpolation problem within the context of an Ising model.**

I. Introduction

This paper develops Local Conditioning as a variant of Belief Propagation for truly distributed exact inference in a graphical model on a cyclic undirected graph $G = (V, E)$ of nodes V and undirected edges E.

Such a model is specified by a collection of random variables (X_i) associated with the nodes in V, a set of functions (ϕ_i) and (ψ_{ij}) defined, respectively, on indvidual nodes and the endpoints of edges, and a function

$$P(\mathbf{x}) = \prod_i \phi_i(x_i) \prod_{i,j} \psi_{ij}(x_i, x_j)$$

that decomposes as a product of these functions[1]. Given such a model, the two primary *graphical model inference* problems are to compute the *Sum-Product beliefs*

$$\begin{aligned} Z_i(x_i) &\triangleq \sum_{x'_{V\setminus i}} P(x_i, x'_{V\setminus i}) \\ &= \sum_{x_{V\setminus i}} \prod_i \phi_i(x_i) \prod_{i,j} \psi_{ij}(x_i, x_j) \end{aligned}$$

and the *Min-Sum beliefs*[2]

$$\begin{aligned} Q_i(x_i) &\triangleq \min_{x'_{V\setminus i}} \Big(-\log P(x_i, x'_{V\setminus i}) \Big) \\ &= \min_{x_{V\setminus i}} \left[\sum_i \Phi_i(x_i) + \sum_{i,j} \Psi_{ij}(x_i, x_j) \right] \end{aligned}$$

where ϕ_i and ψ are *node* and *edge potential functions*, respectively, Φ and Ψ denote the negative logarithms of these functions, and with some abuse of terminology, they are also referred to as potential functions or simply potentials. The term *inference* is used because often the function P is normalized in which case it has the interpretation of a probability distribution. Moreover, if each node is set to an arg maximum of its respective Z_i, the resulting configuration on G will be a concatenation of per-node optima, and as a result, determining Z_i for each node leads to *per-symbol MAP* estimation. Alternatively, if each node is set to an arg minimum of the respective Q_i, the value at each node is the component at the node of some block optima and the resulting configuration on G *may be* a joint or block optima, depending on whether the choices at the individual nodes are consistent with each other. Indeed, computing Q_i for each node, followed by a "backwards" step to enforce consistency, leads to *global MAP* estimation.

It was shown in [2] that when viewed as operations on a semi-ring, these two problems are mathematically equivalent, and that both can be solved with an appropriate generalization of Belief Propagation (BP) [3], and that the underlying graphical structure is what determines the implementation, as well as correctness and efficiency, of BP. If the graph is acyclic, it is referred to as a *tree* and BP involves recursive messages passed between nodes of the graph and converges to the correct beliefs with complexity linear in the number of edges, thus providing an accurate, computationally efficient, and truly distributed algorithm for computing beliefs. In cyclic graphs, one can implement BP as if it were a tree by following the same rules for computing and passing messages to neighbors as one does in an acyclic graph. This is called *Loopy* Belief Propagation (LBP) and it is not guaranteed to converge or be correct even if it does. Thus, in general, while it is computationally efficient and truly distributed, it is not necessarily accurate. Alternatively, one can cluster the nodes into supernodes to make an acyclic graph and then use BP on the graph of supernodes. Complexity of this method is exponential in the size of the supernodes. More importantly for the purposes of this paper, Cluster BP is not truly distributed with respect to the original graph. This paper develops Local Conditioning, a truly distributed, accurate implementation of Belief Propagation for cyclic undirected graphs. The complexity is comparable to that of Cluster BP. However, it has the advantage of being truly distributed and is thus suited for applications such as sensor networks in which the notion of clustering nodes does not have a natural physical interpretation.

The idea of Local Conditioning was initially introduced in the context of directed graphs [4]. It is a modification

[1]Note that generalizations with potential functions defined on higher order cliques of G are also possible, though the present formulation in terms of *pairwise graphical models* suffers no loss of generality [1]

[2]Taking the $\max$ of $P(\cdot)$ is more common in the literature, in which case the problem is one of finding *Max-Product* beliefs.

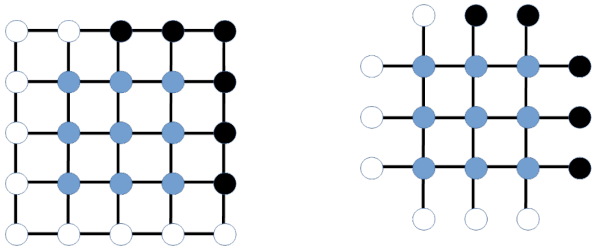
to the idea of Conditioning [3, 5], which can be thought of as doing inference "by cases", performing inference under different assignments or hypotheses on a given *conditioning* subset of nodes, and then combining the conditional beliefs. The effect of conditioning on a node can be represented by an equivalent graph in which each conditioned node has been split into multiple copies, one attached to each of the split node's neighbors in the original graph, where beliefs computed in the second graph equal those in the first graph. A subset of nodes whose removal eliminates all cycles is called a *loop cutset*. By conditioning on a loop cutset, acyclic Belief Propagation can be used to compute beliefs in the original graph that are conditioned on the particular configuration to which the subset has been fixed. The results from BP on all possible conditioning values of the loop cutset can be combined to obtain the desired beliefs in the original graphical model. This method was introduced [6] around the same time that research into clustering took place [7]. With the method of Conditioning just described, the complexity of computing beliefs in the original graph is exponential in the size of the loop cutset, because we have to perform BP for each loop cutset configuration. If one performs parallel rather than serial conditioning, there are savings that can be achieved by recognizing that a given message need not be conditioned on all loop cutset nodes. Rather, for each edge there is a smaller *relevant* subset of loop cutset nodes on which a message passed over that edge must be conditioned. This version of Conditioning is called Local Conditioning [4, 8] and can attain great savings compared to standard Conditioning. There was a great deal of research into efficiently finding small loop cutsets.

Research into conditioning algorithms eventually stalled, and this likely had to do with the primary difference between clustering and conditioning algorithms. In clustering algorithms, once the new graph is formed through clustering, the messages that are passed are strictly local in the sense that a message from one supernode to an adjacent supernode is a function of the values that the receiving supernode can assume. In conditioning algorithms, on the other hand, the messages are matrices where each column is conditioned on a different configuration on the appropriate relevant set. Also in Local Conditioning, the relevant set of nodes will not necessarily be connected. Indeed, there is a tradeoff between locality and distributedness of the messages in Cluster BP and conditioning, and while the distributed nature of conditioning is appealing for truly distributed applications, the nonlocality makes analysis and implementation more difficult. As with clustering algorithms, initial research into conditioning focused on directed graphs. Because the nonlocal regions determining the conditioned messages are sensitive to the directionality of the edges within this region, generalization was likely difficult. Indeed, the main explication of the ideas of Local Conditioning given in [4] is through reference to a specific example. In addition, the nonlocality of conditioned message passing means that loop cutset nodes follow different rules for updating messages to neighbors depending on the particular neighbor. This is unlike clustering algorithms, where supernodes follow the standard Belief Propagation rules between supernodes. Since at the time that this research was being conducted, problems for which distributed calculations were sought were still for the most part being performed on centralized computers, it seems likely that the additional implementation complexity of conditioning algorithms was not deemed worth it.

In this paper we pick up the discussion of Local Conditioning in the context of undirected graphs with the goal of producing a truly distributed algorithm. This was begun in [8]. A prime motivation for doing so is the preponderance of applications involving distributed sensors where there is no centralized processor having access to all values in the network at once, thus requiring a truly distributed algorithm for performing inference throughout the network. This makes it essential to confront the implementation complexity of conditioning algorithms. This complexity is reduced somewhat by considering undirected graphs, in which all edges are the same, which simplifies the notion of topology and with it analysis of message passing updates by allowing the introduction of terms that are more easily defined. Undirected graphs are a good model for sensor networks of connected devices, so doing the analysis with such graphs is not a shortcoming. Moreover, it is known that even if such a network is better represented as a directed graph, a directed graph can be straightforwardly converted to an undirected graph, making the developments in this paper general. It should be said that even though the motivation for this paper are applications with distributed devices, there are scenarios in which cluster heads may be introduced that can in a sense serve as the cluster nodes used in Cluster BP, enabling Cluster BP to be used in truly distributed settings. Nevertheless, we feel that a full understanding of Local Conditioning, and eventually, of the tradeoffs with Cluster BP, is required for taking advantage of the full promise of the distributed systems that are growing in popularity. We note that there has been recent work in what the authors are calling "locally conditioned BP", which is based on the same idea of Conditioning, but which is an algorithm for approximate rather than exact inference [9].

The main contribution of this paper is rules for Local Conditioning for undirected graphs by formalizing the concepts inherent in [4] and providing recursive update equations analogous to those for standard Belief Propagation. An aspect of our demonstration is a development of Conditioning that operates on the original cyclic graph rather than on a so-called associated tree.

Though our development of Local Conditioning is for a general undirected graphical model, in this paper we illustrate the principles with reference to the following interpolation problem. Consider an $N \times N$ square grid graph with nodes arranged in a rectangular lattice and edges connecting horizontally and vertically adjacent nodes. We observe binary values on the boundary nodes, and wish to interpolate the interior to minimize the number of *odd bonds* within the block and between the block and its boundary [10, 11]. Such a scenario



(a) (b)

Fig. 1. (a) Block boundary; (b) Graph indicating conditional distribution of block given its boundary. White and black nodes indicate fixed values on the boundary, and blue nodes indicate unknown values that must be interpolated.

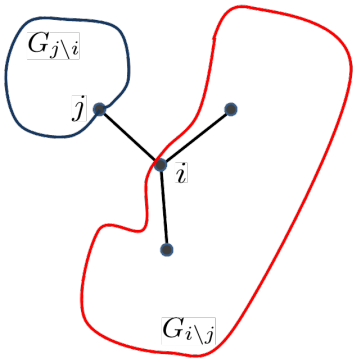


Fig. 2. An edge $\{i,j\}$ and the components $G_{i\setminus j}$ and $G_{j\setminus i}$.

might arise, for example, in a network in which communication cost was cheap relative to sensing cost, motivating the use of fewer sensing devices followed by communication for interpolating the values at the devices not performing sensing.

The rest of the paper is organized as follows. In Section II we introduce background on graphs, the minimum odd bond interpolation problem used for illustration, solution of this problem using Belief Propagation, Loopy Belief Propagation, and Cluster BP. In Section III, we develop Conditioning in the context of undirected graphs, and in Section IV, we develop Local Conditioning for undirected graphs.

II. Background

As mentioned in the introduction, we will illustrate the general local conditioning method to solve a min sum problem in the context of an interpolation problem, which we now describe.

A. Minimum Odd Bond Interpolation

Consider an $(N+2)\times(N+2)$ grid graph $G=(V,E)$ with the *4-nbr. topology* in which the nodes in V are arranged in a square lattice and the undirected edges in E consist of horizontally and vertically adjacent nodes of V. Two sites connected by an edge are referred to as *neighbors*. The set of nodes to which a node i is connected is denoted ∂i and referred to as the *boundary* of i. For each node $i\in V$ there is an associated variable X_i where a particular assignment to i is denoted x_i. In this paper, x_i takes values in $\{0,1\}$ and an assignment to all the nodes is denoted $\mathbf{x}$ and is referred to as a *configuration*. For a given configuration, an edge $\{i,j\}$ where $x_i\neq x_j$ is called an *odd bond*. We let $B\subset V$ denote the set of nodes that have four neighbors, and let $\partial B=V\setminus B$ denote the boundary of B.

The application problem of this paper is to find a configuration $\mathbf{x}_B$ on B with a minimum number of odd bonds, given a configuration $\mathbf{x}_{\partial B}$ of its boundary. Figure 1 illustrates a block B of unknown values and a configuration of fixed values on its boundary. This can indeed be cast as a min-sum problem by choosing the edge potential function to be

$$\Psi_{ij}(x_i,x_j) \quad = \quad \mathbf{I}_{(x_i\neq x_j)},$$

where $\mathbf{I}_A$ is the indicator function for event A. A minimum odd bond configuration is also a MAP configuration with respect to a uniform Ising model which assigns the following probabilities to configurations:

$$P(\mathbf{x}) \quad \propto \quad \exp\{\theta\sum_{i,j}\mathbf{I}_{x_i=\ x_j}\},$$

with positive edge parameter θ [10].

The minimum number of odd bonds over all $\mathbf{x}_B$ is

$$Q^*=\min_{\mathbf{x}_B}\sum_{i,j}\mathbf{I}_{(x_i\neq x_j)}$$

where the summation is only over $\{i,j\}\in E$, and the value x_j is fixed for $j\in\partial B$.

Again with x_j fixed for $j\in\partial B$, for each node i and possible value x_i, we want to compute

$$Q_i(x_i)=\min_{\mathbf{x}_{B\setminus i}}\left[\sum_{j,k\neq i}\mathbf{I}_{(x_j\neq x_k)}+\sum_{j\in\partial i}\mathbf{I}_{(x_j\neq x_i)}\right].$$

We say that $Q_i(x_i)$ is the *min-sum belief* for x_i at node i.

B. Min-Sum Belief Propagation

In this section we describe the conventional Belief Propagation (BP) approach to solving the minimum odd bond problem. The algorithm operates iteratively, where in each iteration nodes compute and send messages to neighbors, an outgoing message being a function of previously received messages and the edge potential with the recipient node. The key issue is how a node computes beliefs from incoming messages and recursively computes outgoing messages from incoming messages and the potential functions between it and its neighbors. Here we review the well-known approach for trees. After the BP algorithm computes messages and beliefs, a subsequent set of steps uses the beliefs to find a minimum odd bond configuration.

In this section, let $G=(V,E)$ be a tree and let B be a connected subset of V such that $B\cup\partial B=V$. We again observe a configuration $x_{\partial B}$ on the boundary of B, and wish to compute the min-sum beliefs $Q_i(x_i)$ for each node $i\in B$ conditioned on the boundary. Each edge $\{i,j\}$ in B is a

cut edge, meaning that removing $\{i, j\}$ separates G into two components. We let $G_{j\setminus i} = (V_{j\setminus i}, E_{j\setminus i})$ be the component that contains j after removing the edge connecting i and j. This is illustrated in Figure 2. We now let $Q_{j\setminus i}(x_j)$ be the min-sum belief for x_j at node j in $G_{j\setminus i}$, that is, the minimum number of odd bonds in $G_{j\setminus i}$ when j has the value x_j:

$$Q_{j\setminus i}(x_j) \quad = \quad \min_{\mathbf{x}_{B_{j\setminus i}}} \left[\sum_{k,l \in B_{j\setminus i}\setminus j} \mathbf{I}_{(x_k \neq x_l)} + \sum_{k \in \partial j \setminus i} \mathbf{I}_{(x_k \neq x_j)} \right],$$

where $B_{j\setminus i} = V_{j\setminus i} \cap B$ and the nodes in ∂B are fixed.

For node i, if we know $Q_{j\setminus i}(x_j)$ for each of its neighbors $j \in \partial i$, then it is straightforward to find a solution for the belief $Q_i(x_i)$ at node i. For instance, it is straightforward to show that for any neighbor $j \in \partial i$,

$$Q_i(x_i) \quad = \quad Q_{i\setminus j}(x_i) + \min_{x_j} \left[\mathbf{I}_{x_j \neq x_i} + Q_{j\setminus i}(x_j) \right],$$

and by proceeding inductively,

$$Q_i(x_i) \quad = \quad \sum_{j \in \partial i} \min_{x_j} \left[\mathbf{I}_{x_j \neq x_i} + Q_{j\setminus i}(x_j) \right].$$

We define the *message* $m_{j\to i}$ from node j to a neighbor i as the vector consisting of one component for each value that node i can assume,

$$m_{j\to i} = \begin{bmatrix} m_{j\to i}(0) \\ m_{j\to i}(1) \end{bmatrix},$$

where the components of $m_{j\to i}$ are defined as

$$m_{j\to i}(x_i) \quad \triangleq \quad \min_{\mathbf{x}_j} \left[\mathbf{I}_{(x_j \neq x_i)} + Q_{j\setminus i}(x_j) \right].$$

Likewise, the belief Q_i at node i is a vector consisting of one component for each value that node i can assume:

$$Q_i = \begin{bmatrix} Q_i(0) \\ Q_i(1) \end{bmatrix}. \tag{1}$$

The message $m_{j\to i}$ can be thought of as summarizing information about the subgraph $G_{j\setminus i}$ for node i for the problem at hand. For example, in our problem, $m_{j\to i}$ communicates to node i, for each value x_i that i can assume, the minimum number of odd bonds due to $G_{j\setminus i}$ and the edge connecting j and i.

A node k is said to be *upstream* of message $m_{j\to i}$ if $k \in G_{j\setminus i}$ and *downstream* of $m_{j\to i}$ if $k \in G_{i\setminus j}$. As such, $m_{j\to i}$ is not a function of downstream nodes, i.e., the message $m_{j\to i}$ would be the same regardless of the node values and potentials in $G_{i\setminus j}$. On the other hand, $m_{j\to i}$ is a function of upstream nodes, and as a part of the solution to $Q_i(x_i)$, $m_{j\to i}$ has already minimized over the potentials involving upstream nodes. Figure 3 illustrates upstream and downstream nodes.

We now summarize the above development with the following belief and message recursion rules for min-sum BP.

Proposition 2.1 (Min-Sum Belief Propagation):

$$Q_i(x_i) \quad = \quad \sum_{j \in \partial i} m_{j\to i}(x_i)$$

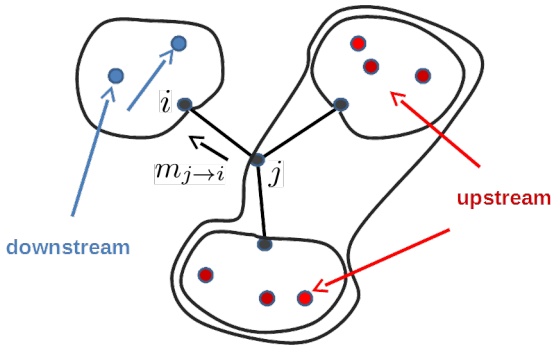


Fig. 3. A message from node j to i with upstream and downstream nodes indicated.

$$m_{j\to i}(x_i) \quad = \quad \min_{x_j} \left[\mathbf{I}_{(x_j \neq x_i)} + \sum_{k \in \partial j \setminus i} m_{k\to j}(x_j) \right]$$

Using the above update rules, BP message passing proceeds through a sequence of steps, beginning at the leafs of the tree, with each node computing an outgoing message after receiving incoming messages from each of its other neighbors. Once a node has received messages from all of its neighbors, it computes its belief. Note that there are versions of BP in which nodes pass messages to all neighbors at all iterations of the algorithm, where incoming messages are initialized in some manner, typically to the all-zero vector. It is straightforward to show that messages in this second, parallel message-passing scheme will converge to the same messages as in the initial, sequential scheme described. However, our development of Conditioning in Section III relies on the use of decision-logic in the passing of messages, so in light of this we prefer the interpretation of BP in which a node passes an outgoing message only after it has received messages from all of its other neighbors.

Proposition 2.1 generalizes to give the messages sent when the sending node j is (a) a leaf node, as

$$m_{j\to i}(x_i) \quad = \quad \min_{x_j} \mathbf{I}_{(x_i \neq x_j)},$$

(b) a node whose value is fixed to $\bar{x}_j$, as

$$m_{j\to i}(x_i) \quad = \quad \mathbf{I}_{(x_i \neq \bar{x}_j)} + \sum_{k \in \partial j \setminus i} m_{k\to j}(\bar{x}_j), \tag{2}$$

and (c) a leaf node fixed to value $\bar{x}_j$, as

$$m_{j\to i}(x_i) \quad = \quad \mathbf{I}_{(x_i \neq \bar{x}_j)}. \tag{3}$$

C. Backward Pass: Finding a Global Solution

Ultimately we would like to use min-sum BP to find a global MAP solution. From the arg minima of its min-sum belief Q_i, each node i determines what values it *could* assume in *some* global MAP configuration on B. A global configuration

$(x_i^* : i \in B)$ can be achieved by implementing a *backward pass* in which the nodes decide, one by one, the values they will assume in a particular global minimizing configuration.

For instance, given that B is a tree, an arbitrary site $r \in B$ can be chosen as the root. The selection of r as the root defines *parent-child* relations throughout the tree. Then, starting at the root node r, r computes its belief Q_r, selects an arg minimizer x_r^* of this belief, then for each of its children i passes the message

$$m_{r\to i}^{(x_r^*)}(x_i) = \mathbf{I}_{(x_r^* \neq x_i)}.$$

Proceeding in this way, after a node i receives the message $m_{\pi(i)\to i}^{(x^*_{\pi(i)})}$ from its parent $\pi(i)$, it combines it with the messages $m_{j\to i}$, $j \in \partial i \setminus \pi(i)$, received from its other neighbors, computes its (conditioned) belief $Q_i^{(x^*_{\pi(i)})}$, selects an arg minimizer of this belief, and continues by sending a conditioned message to its children.

This is summarized in the following backward pass update rules.

Proposition 2.2 (Min-Sum Backward Pass): A node i computes its conditioned belief

$$Q_i^{(x^*_{\pi(i)})} = m_{\pi(i)\to i}^{(x^*_{\pi(i)})} + \sum_{j\in\partial i\setminus\pi(i)} m_{j\to i},$$

selects an arg minimizer

$$x_i^* \in \arg\min_{x_i} Q_i^{(x^*_{\pi(i)})},$$

and passes to its children $j \in \partial i \setminus \pi(i)$ the conditioned message

$$m_{i\to j}^{(x_i^*)} = \left[\mathbf{I}_{(x_i\neq x_j)}\right] + m_{\pi(i)\to i}^{(x^*_{\pi(i)})} + \sum_{k\in\partial i\setminus(j\cup\pi(i))} m_{k\to i}$$

The resulting configuration

$$\mathbf{x}^* = (x_i^* : i \in B)$$

is minimum odd bond configuration.

D. Loopy Belief Propagation

For the remainder of the paper we will be in the setting introduced in Section II-A, in which $G = (V, E)$ is a grid graph, B are the interior sites, ∂B is the boundary of B, and we wish to minimize the number of odd bonds in B given a configuration on ∂B. Since G is not a tree, we could apply *Loopy* Belief Propagation, in the same message-passing recursions are applied as if G were a tree. Meaning, the update rules are as follows.

Proposition 2.3 (Min-Sum Loopy Belief Propagation):

$$Q_i^n(x_i) = \sum_{j\in\partial i} m_{j\to i}^n(x_i)$$

$$m_{j\to i}^n(x_i) = \min_{x_j}\left[\mathbf{I}_{(x_j\neq x_i)} + \sum_{k\in\partial j\setminus i} m_{k\to j}^n(x_j)\right]$$

Loopy BP is typically implemented with a parallel message-passing scheme, in which all nodes send messages to all neighbors at all iterations, and incoming messages are initialized in some way. Convergence properties of this algorithm have been well-studied, though much remains unknown about general performance.

As for finding a configuration that minimizes the number of odd bonds, there is no "backward pass" but we can use the individual beliefs to assign values to the nodes in B. For example, if $Q_i(1) > Q_i(0)$, then one can set $x_i^* = 1$, and likewise if $Q_i(1) < Q_i(0)$. However, as in the case of a tree, when $\arg\min_{x_i} Q_i^{(x_{\pi(i)})}$ consists of more than one value, values at such nodes cannot be chosen independently of one another. However, there is no generally applicable approach to making a backward pass to determine a consistent configuration on such nodes. Hence heuristics are usually employed.

E. Cluster Belief Propagation

In the Clustering version of Belief Propagation, nodes of the original graph are grouped into clusters C_k in such a way that an acyclic graph is formed when each cluster is viewed as a *supernode*, and edges are placed between every two supernodes C_k and C_j for which there are nodes i in C_k and j in C_j that are joined by an edge in the original graph. Figure 4 illustrates the clustering of columns into supernodes for a square grid graph, the result being a line graph. A self-potential Φ_{C_k} for a cluster node C_k is determined by summing the potentials defined on nodes within C_k, as in

$$\Phi_{C_k}(\mathbf{x}_{C_k}) = \sum_{i,i'\in C_K} \mathbf{I}_{(x_i\neq x_{i'})},$$

where the summation is over edges $\{i, i'\}$ in the original graph both of whom are in C_k. Likewise, edge-potentials Φ_{C_k,C_j} are defined by summing over the edge-potentials containing a node in C_k and a node in C_j, as in

$$\Psi_{C_k,C_j}(\mathbf{x}_{C_k}, \mathbf{x}_{C_j}) = \sum_{i\in C_k, i'\in C_j} \mathbf{I}_{(x_i\neq x_{i'})},$$

where again the summation is restricted to nodes i,i' that are joined by an edge in the original graph.

Since the cluster graph is acyclic, Belief Propagation can then be performed on the cluster graph following the same algorithm as outlined in Section II-B. The belief vectors computed in Cluster BP will now be larger vectors. For example, if three nodes are clustered, the belief vector will have the form

$$Q_{C_k}(x_{C_k}) = \begin{bmatrix} Q_{C_k}(000) \\ Q_{C_k}(001) \\ \vdots \\ Q_{C_k}(111) \end{bmatrix}.$$

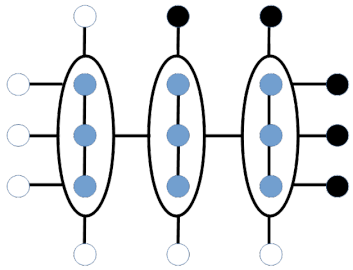


Fig. 4. Block interior clustered into a chain graph, the original boundary nodes adjusting the node potentials of the supernodes.

Thus belief vector sizes are exponential in the size of the cluster. Likewise, Cluster BP messages will be larger vectors such as

$$m_{C_j \to C_k}(x_{C_k}) \quad = \quad \begin{bmatrix} m_{C_j \to C_k}(000) \\ m_{C_j \to C_k}(001) \\ \vdots \\ m_{C_j \to C_k}(111) \end{bmatrix}.$$

Again, the vector will have length that is exponential in the size of the receiving cluster. Therefore the complexity of performing Cluster BP on an $N \times M$ grid graph is exponential in the smaller of N and M, and linear in the larger of the two. The belief for an individual node of the original graph can be obtained by minimizing the clusered belief for the cluster containing the given node, over the other nodes contained in that cluster.

The following provides the rule for computing beliefs and the message recursion for clustered nodes, as well as the beliefs for individual nodes from the cluster node beliefs. Note that some arguments are supressed due to space limitations.

Proposition 2.4 (Min-Sum Clustered Belief Propagation): The belief at a cluster node C_k is computed as

$$Q_{C_k}(x_{C_k}) \quad = \quad \sum_{j \in \partial i} m_{C_j \to C_k}(x_{C_k}),$$

a message from cluster C_j to neighboring cluster $C_k \in \partial C_j$ is

$$m_{C_j \to C_k} = \min_{x_{C_j}} \left[\Phi_{C_j} + \Psi_{C_j, C_k} + \sum_{C_l \in \partial C_j \setminus C_k} m_{C_l \to C_j} \right]$$

and the belief at an individual node i in cluster C_k is computed as

$$Q_i(x_i) \quad = \quad \min_{(x_{C_k})_i = x_i} Q_{C_k}(x_{C_k})$$

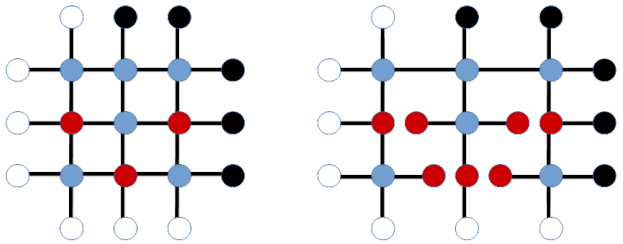
A backward pass analogous to that in Section II-C an be performed to find an optimal configuration on the clustered nodes. Indeed, performing the backward pass on the clustered graph is actually more efficient than attempting to define a new backward pass using beliefs computed for individual nodes, as an arg minimizing configuration on a supernode is self-consistent.

III. Conditioning for Undirected Graphs: A New Formulation

As mentioned earlier, Conditioning was previously introduced in the context of directed graphs. Due to their complex topologies, a concise, general formulation of the belief and message update rules analogous to those of Proposition 2.1 was not possible. Instead, conditioning was developed by exhibiting the principles through examples [5, 6]. In contrast, the simpler topologies of undirected graphs do lend themselves to such a concise formulation. In this section we exploit this simplicity, and leverage the underlying principles inherent in the earlier work, to develop belief and message update rules for Conditioning analogous to those of Proposition 2.1.

As mentioned in Section I, conditioning can be formulated based on the idea of creating an *associated* tree by splitting conditioned nodes into multiple copies. However, in this paper we are principally concerned with application to distributed settings, where creating a new graph and making multiple copies of nodes does not make physical sense. Thus, while we view the construction of the associated tree as a useful schematic, and indeed will begin our development of Conditioning by introducing this construction, ultimately we view Conditioning not as an algorithm on an associated tree, but rather as an algorithm on the original graph with neighbor-specific message-passing rules for loop cutset nodes guided by the associated tree.

Recall that a *loop cutset* L is a subset of nodes such that if we remove L and all edges incident to it, the resulting graph $G \setminus L$ has no cycles. To obtain the associated tree, we first remove all loop cutset nodes and edges incident to them, and then for each loop cutset node $l \in L$ and each neighbor $j \in \partial l$, create a new node $l^{(j)}$, called a *copy*, and attach it to j by adding the edge $\{j, l^{(j)}\}$. The resulting graph still has no cycles, and there is a one-to-one correspondence between edges in the original cyclic graph and edges in the new acyclic graph. If the resulting acyclic graph is disconnected, one can form a tree by re-identifying copies of the loop cutset nodes in an iterative manner. The result is an *associated tree* in which copies of loop cutset nodes may be leaf nodes or may have multiple neighbors. Figure 5 shows the original block with loop cutset nodes indicated, as well as the associated tree obtained by re-identifying some copies of loop cutset nodes. Without loss of generality, we will assume that for a loop cutset node, at most one copy has multiple neighbors, i.e., is not a leaf. By appropriately copying edge potentials from edges in the original graph to the corresponding edges in the associated tree, standard (acyclic) Belief Propagation can be performed on the ayclic graph. In order for beliefs computed on the associated tree to be useful for computing beliefs in the original graph, all copies of a given loop cutset node need to be constrained to have the same value. For such a constrained configuration on the copies of loop cutset nodes, the resulting beliefs on the associated tree will correspond to *conditioned beliefs* in the original graph. That is, beliefs $Q_i^{(x_L)}$ conditioned on a specific configuration x_L of loop cutset



(a) (b)

Fig. 5. (a) 3×3 grid graph with loop cutset nodes indicated in red; (b) the associated tree based on a particular re-identification of copies of loop cutset nodes.

nodes, and correspondingly, the messages $m_{j\to i}^{(x_L)}$ passed will be conditioned on specific configurations of loop cutset nodes.

As mentioned at the beginning of this section, we view Conditioning as an algorithm involving computation of beliefs and messages on the original graph. However, in order for the messages on the original graph to permit computation of the correct beliefs, the messages must respect the *unwrapped topology* of the associated tree. That is, if a copy $l^{(j)}$ of a loop cutset node l is a leaf in the associated tree, then in the original graph, l will pass to neighbor j a message formed *as if* l were a leaf. Moreover, since the message $m_{l\to j}^{(x_L)}$ from l to j is conditioned on a configuration of the loop cutset nodes, the message from l to j is formed as if l were a leaf with a fixed value, i.e., it must be formed according to (3):

$$m_{l\to j}^{(x_L)}(x_j) \quad = \quad \mathbf{I}_{(x_j \neq x_l)},$$

where x_l is the value of node l in x_L. For a loop cutset node l, we let $\mathcal{L}_l \subset \partial l$ denote the neighbors of l for which the copy of l attached to the neighbor is a leaf node in the associated tree. On the other hand, we let $\mathcal{N}_l = \partial l \setminus \mathcal{L}_l$ be the neighbors of l that are connected to a non-leaf copy of l in the associated tree. In this case, l will pass a message to neighbor $j \in \mathcal{N}_l$ formed according to the rule for a fixed non-leaf node given in (2), *as if* the neighbors in $\mathcal{N}_l$ are its only neighbors. That is, it passes to neighbor $j \in \mathcal{N}_l$ the message

$$m_{l\to i}^{(x_L)}(x_i) \quad = \quad \mathbf{I}_{(x_l \neq x_i)} + \sum_{k \in \mathcal{N}_l \setminus i} m_{k\to l}^{(x_L)}(x_l).$$

For example, in Figure 6, loop cutset node l_3 will pass to its northern neighbor a conditioned message formed by combining the message from its eastern neighbor, which is a fixed boundary node, the message from its southern neighbor, and the potential between it and the northern neighbor, all using the fixed value x_{l_3} of l_3 in the loop cutset configuration x_L corresponding to the conditioned message. On the other hand, l_3 will pass to its western neighbor a message consisting only of the potential between it and the western neighbor, again with the fixed value of x_{l_3}.

Messages from non-loop cutset nodes, on the other hand, are computed in the usual way as stated in Proposition 2.1, as

$$m_{j\to i}^{(x_L)}(x_i) \quad = \quad \min_{x_j} \left[\mathbf{I}_{(x_j \neq x_i)} + \sum_{k \in \partial j \setminus i} m_{k\to j}^{(x_L)}(x_j) \right]$$

with the obvious difference of conditioning on a loop cutset configuration. In this way the utility of the associated tree is not in constructing a new graph on which messages can be passed, but rather in defining neighbors-specific message passing rules for loop cutset nodes.

In Conditioning the beliefs and messages are now matrices, each column corresponding to conditioning on a particular configuration of the loop cutset nodes. For example, if as in Figures 5 and 6 there are three loop cutset nodes, the beliefs would have the form shown below.

$$\begin{aligned} Q_i^{(L)} \quad &= \quad \begin{bmatrix} Q_i^{(000)}(0), & Q_i^{(001)}(0), & \ldots, & Q_i^{(111)}(0) \\ Q_i^{(000)}(1), & Q_i^{(001)}(1), & \ldots, & Q_i^{(111)}(1) \end{bmatrix} \\ &= \quad \begin{bmatrix} Q_i^{(000)}, & Q_i^{(001)}, & \ldots, & Q_i^{(111)} \end{bmatrix} \end{aligned}$$

and the messages would have the form

$$M_{i\to j}^{(L)} \quad = \quad \begin{bmatrix} m_{i\to j}^{(000)}, & m_{i\to j}^{(001)}, & \ldots, & m_{i\to j}^{(111)} \end{bmatrix}.$$

The following are the Belief Propagation update rules to use with Conditioning.

Proposition 3.1 (Conditioning Min-Sum BP):
For a non loop cutset node $j \notin L$, element x_j of column x_L of the conditioned beliefs and messages are computed as

$$Q_j^{(x_L)}(x_j) \quad = \quad \sum_{k \in \partial j} m_{k\to j}^{(x_L)}(x_j)$$

$$m_{j\to i}^{(x_L)}(x_i) \quad = \quad \min_{x_j} \left[\mathbf{I}_{(x_j \neq x_i)} + \sum_{k \in \partial j \setminus i} m_{k\to j}^{(x_L)}(x_j) \right].$$

For a loop cutset node $l \in L$, conditioned beliefs are computed as

$$Q_l^{(x_L)}(x_l) \quad = \quad m_{j\to l}^{(x_L)}(x_l),$$

where j is a neighbor in $\mathcal{L}_l$; the conditioned message to a neighbor $i \in \mathcal{N}_l$ is computed as

$$m_{l\to i}^{(x_L)}(x_i) \quad = \quad \mathbf{I}_{(x_l \neq x_i)} + \sum_{k \in \mathcal{N}_l \setminus i} m_{k\to l}^{(x_L)}(x_l).$$

and the conditioned message to a neighbor $i \in \mathcal{L}_l$ is computed as

$$m_{l\to i}^{(x_L)}(x_i) \quad = \quad \mathbf{I}_{(x_l \neq x_i)}.$$

Finally, unconditioned beliefs are computed as

$$Q_i(x_i) \quad = \quad \min_{x_L} Q_i^{(x_L)}(x_i)$$

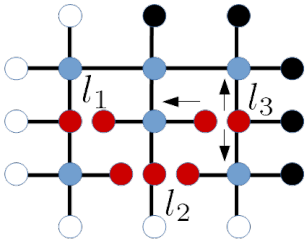


Fig. 6. Loop cutset nodes l_1, l_2, and l_3 on the associated tree. Messages from loop cutset nodes respect the unwrapped topology of the associated tree.

To select a configuration that minimizes the number of odd bonds on B, the nodes would perform a backward pass as in Proposition 2.2, with the difference that the messages in the backward pass would be conditioned on the loop cutset nodes in addition to the parent.

At this point it is worth observing that a couple of key differences between Cluster BP and Conditioning. Whereas in the former messages and beliefs are long vectors corresponding to configurations on large supernodes, in the latter messages and beliefs are matrices where rows correspond to the values of individual nodes and columns correspond to values on the loop cutset. In Cluster BP, one circumvents the cycles of the original graph by clustering nodes and the resulting complexity is exponential in the size of the clusters. In Conditioning, on the other hand, one gets around cycles by introducing neighbor-specific message-passing rules motivated by the associated tree. Related to this, an advantage of Conditioning over Cluster BP is that it is a truly distributed algorithm between nodes of the original graph.

IV. Local Conditioning for Undirected Graphs

Local Conditioning is an adaptation of the basic Conditioning algorithm presented in the previous section that achieves computational savings by identifying for each edge and corresponding message, a set of loop cutset nodes on which the message does not depend. Equivalently, one identifies a strict subset of loop cutset nodes on which the message matrix or belief computation must be conditioned. To understand this, we again appeal to the associated tree. Of course, once the *relevant* subsets of loop cutset nodes are identified through recourse to the unwrapped tree, the actual message-passing takes place on the orginal cyclic graph.

Using Proposition 3.1, we can express the belief $Q_i(x_i)$ at node i as

$$\begin{aligned} Q_i(x_i) &= \min_{x_L} Q_i^{(x_L)}(x_i) \\ &= \min_{x_L} \sum_{j\in\partial i} \min_{x_j} \left[\mathbf{I}_{(x_j\neq x_i)} + Q_{j\backslash i}^{(x_L)}(x_j)\right]. \quad (4) \end{aligned}$$

Recall from Section II-B that a node k is upstream of message $M_{j\to i}$ if $k \in G_{j\backslash i}$, and downstream of $M_{j\to i}$ if $k \in G_{i\backslash i}$. For a given message $M_{j\to i}$, we let $L_{j\backslash i}$ be the set of loop cutset nodes all of whose copies are upstream of $M_{j\to i}$. Note that for message $M_{j\to i}$, the loop cutset nodes in $L_{j'\backslash i}$ for all $j' \neq j$ are such that all copies of them are downstream of $M_{j\to i}$, and note further that $L_{i\backslash j}$ is the set of loop cutset nodes all of whose copies are downstream of $M_{j\to i}$. The message $M_{j\to i}$ from j to i is not a function of the loop cutset nodes in the downstream $L_{i\backslash j}$, thus message columns corresponding to loop cutset configurations differing only on loop cutset nodes in $L_{i\backslash j}$ will be identical. This means that fewer columns are required in the message $M_{j\to i}$. Similarly, again regarding $M_{j\to i}$, one can minimize over message columns corresponding to loop cutset configurations differing only on loop cutset nodes in upstream $L_{j\backslash i}$, again resulting in fewer columns.

For a given node i, let $j_1, j_2, \ldots, j_{n_i}$ indicate the neighbors of i. The set of loop cutset nodes L can be partitioned into $L_{j_1\backslash i}$, $L_{j_2\backslash i}$, …, $L_{j_{n_i}\backslash i}$, and the set

$$R_i = L \setminus \left(\bigcup_{j\in\partial i} L_{j\backslash i} \right)$$

of loop cutset nodes at least one of whose copies is upstream and at least one of whose copies is downstream of $M_{j\to i}$ for some neighbor $j \in \partial i$. The set R_i is called the *relevant set* for i because its members are the loop cutset nodes who values are required for computing Q_i. Figure 7 illustrates loop cutset nodes all of whose copies are upstream, loop cutset nodes all of whose copies are downstream, and loop cutset nodes where at least one copy is downstream and at least one copy is upstream.

Expanding the minimization in (4), $Q_i(x_i)$ can be computed as

$$\min_{x_{R_i}} \min_{x_{L_{j_1\backslash i}}} \cdots \min_{x_{L_{j_{n_i}\backslash i}}} \sum_{m=1}^{n_i} \min_{x_{j_m}} \left[\mathbf{I}_{(x_{j_m}\neq x_i)} + Q_{j_m\backslash i}^{(x_L)}(x_{j_m})\right].$$

Note that for neighbor j_1, the quantity $\left[\mathbf{I}_{(x_{j_1}\neq x_i)} + Q_{j_1\backslash i}^{(x_L)}(x_{j_1})\right]$ is not a function of the loop cutset nodes in $L_{j_m\backslash i}$, $m \neq 1$. Noting this for $j_2, j_3, \ldots, j_{n_i}$ as well, we see that we can re-write the righthand side of (4) as

$$\begin{aligned} Q_i(x_i) &= \min_{x_{R_i}} \sum_{m=1}^{n} \min_{x_{L_{j_m\backslash i}}} \min_{x_{j_m}} \left[\mathbf{I}_{x_{j_m}\neq x_i} + Q_{j_m\backslash i}^{(x_L)}(x_{j_m})\right] \\ &= \min_{x_{R_i}} \sum_{m=1}^{n} \min_{x_{L_{j_m\backslash i}}} m_{j_m\to i}^{(x_L)}(x_i) \\ &= \min_{x_{R_i}} \sum_{m=1}^{n} \hat{m}_{j\to i}^{(x_{R_i})}(x_i) \end{aligned}$$

where the $\hat{m}_{j\to i}^{(x_{R_i})}$ are columns of *reduced message* matrices $M_{j\to i}^{(R_i)}$ having only columns corresponding to different configurations on the relevant set for i. This then implies that the belief matrix $Q_i^{(R_i)}$ at a node i need not be conditioned on the entire loop cutset either, but rather, only on the relevant set R_i for i. That is, instead of a belief matrix $Q^{(L)}$ having a column for every configuration of the loop cutset nodes, it computes

a belief matrix $Q^{(R_i)}$ having a column for every configuration of its relvant nodes.

It turns out that further reduction in the message matrices is possible. For message $M_{j\to i}^{(R_i)}$ conditioned on the relevant set for i, we note that $R_i \cap L_{i\backslash j}$ is not necessarily empty. This is because there may be loop cutset nodes all of whose copies are downstream of $M_{j\to i}$ yet not all copies of whom are upstream of $M_{j'\to i}$ for one of i's other neighbors $j' \neq j$. Therefore, for each edge $\{i,j\}$, we define

$$R_{ij} = L \setminus \left(L_{j\backslash i} \cup L_{i\backslash j}\right)$$

to be the relevant set for $\{i,j\}$.

Thus each neighbor j of i sends to i the message matrix $M_{j\to i}^{(R_{ij})}$ whose columns correspond to different configurations on the relevant set for $\{i,j\}$. When i receives the messages $M_{j_1\to i}^{(R_{ij_1})}, \ldots, M_{j_{n_i}\to i}^{(R_{ij_{n_i}})}$ from all of its neighbors, it creates the columns of its belief matrix $Q_i^{(R_i)}$ by selecting the column from each of its incoming message matrices that agree on the appropriate subset of R_i.

The following proposition summarizes the Belief Propagation update rules used with Local Conditioning.

Proposition 4.1 (Local Conditioning Min-Sum BP): For a non loop cutset node $j \notin L$, local conditioned beliefs and messages are computed as

$$Q_j^{(x_{R_j})}(x_j) \quad = \quad \sum_{k\in\partial j} \hat{m}_{k\to j}^{(x_{R_{kj}})}(x_j)$$

and

$$\hat{m}_{j\to i}^{(x_{R_{ji}})}(x_i) = \min_{x_j}\left[\mathbf{I}_{(x_j\neq x_i)} + \min_{x_{A_{j\backslash i}}} \sum_{k\in\partial j\backslash i} \hat{m}_{k\to j}^{(x_{R_{jk}})}(x_j)\right],$$

where $A_{j\backslash i} = L_{j\backslash i} \setminus (\cup_{k\in\partial j\backslash i} L_{k\backslash j})$ is the set of loop cutset nodes all of whose copies are upstream of $M_{j\to i}$ but are not in any of the $L_{k\backslash j}$ for $k \in \partial j \setminus i$. Unconditioned beliefs are computed as

$$Q_j(x_j) \quad = \quad \min_{x_{R_j}} Q_j^{(x_{R_j})}(x_j).$$

For a loop cutset node $l \in L$, local conditioned beliefs are computed as

$$Q_l^{(x_{R_l})}(x_l) \quad = \quad m_{j\to l}^{(x_{R_{jl}})}(x_l),$$

where $j \in \mathcal{L}_l$. The message to a neighbor $i \in \mathcal{N}_l$ is computed as

$$\hat{m}_{l\to i}^{(x_{R_{li}})}(x_i) \quad = \quad \mathbf{I}_{(x_l\neq x_i)} + \min_{x_{A_{l\backslash i}}} \sum_{k\in\mathcal{N}_l\backslash i} \hat{m}_{k\to l}^{(x_{R_{kl}})}(x_l),$$

where $A_{l\backslash i} = L_{l\backslash i} \setminus (\cup_{k\in\partial l\backslash i} L_{k\backslash l})$ is the set of loop cutset nodes all of whose copies are upstream of $M_{l\to i}$ but are not in any of the $L_{k\backslash l}$ for $k \in \partial l \setminus i$. Unconditioned beliefs are computed as

$$Q_l(x_l) \quad = \quad \min_{x_{R_l}} Q_l^{(x_{R_l})}(x_l).$$

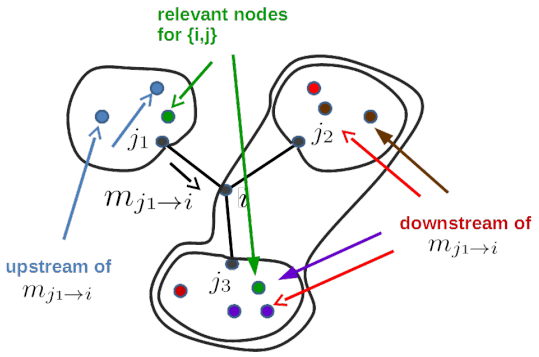


Fig. 7. Loop cutset nodes. Blue loop cutset nodes are in $L_{j_1\backslash i}$, that is, all copies of the blue loop cutset node are upstream of $m_{j_1\to i}$; brown and purple loop cutset nodes are in $L_{j_2\backslash i}$ and $L_{j_3\backslash i}$, respectively, that is, all copies of the brown loop cutset nodes are upstream of $m_{j_2\to i}$ and all copies of purple loop cutset nodes are upstream of $m_{j_3\to i}$. All copies of brown and purple loop cutset nodes are downstream of $m_{j_1\to i}$, that is, brown and purple loop cutset nodes are both in $L_{i\backslash j_1}$. Red loop cutset nodes are in $L_{i\backslash j_1}$ but are not in either $L_{j_2\backslash i}$ nor $L_{j_3\backslash i}$.

V. Concluding Remarks

In this paper we have introduced Local Conditioning as a truly distributed algorithm for performing exact inference on an undirected cyclic graph. This included a development of the basic Conditioning algorithm for undirected cyclic graphs. While the idea of the associated tree used in Conditioning had been introduced in the context of directed graphs, for truly distributed settings, the associated tree is not a realistic object on which actual message-passing can take place. In this paper, however, we propose to use the associated tree as a schematic for defining neighbor-specific message rules for loop cutset nodes on the original cyclic graph. Such an algorithm on the cyclic graph requires nodes with more sophisticated decision-logic than the standard BP algorithm on an associated tree, but as the capabilities of sensors continue to improve, this is becoming, if it is not already, a realistic possibility. Moreover, and more importantly, for truly distributed settings, it is a necessity.

Conditioning by itself, while permitting truly distributed communication through a cyclic network, still carries with it a tremendous cost in complexity due to conditioning on all configurations of the loop cutset. Local Conditioning provides a dramatic decrease in complexity by reducing the conditioning set for the individual messages as well as the beliefs at each node. While the underlying ideas of Local Conditioning were introduced earlier in the context of directed graphs, it was by way of example, rather than through the general topological relations employed in this paper. This is in part due to the simpler topologies of undirected graphs. Keeping in mind application to truly distributed settings, Local Conditioning places even greater demands on the decision-logic and memory of nodes of the network. For example, in order to compute its belief, a node need only keep track of the configurations on its relevant set. However, the relevant

sets of incoming messages are likely to be smaller than the relevant set for the node, and furthermore, the relevant sets of incoming messages are likely to be different from one another. Therefore, a node must maintain a record of what subsets of its relevant set are represented in the incoming messages it receives from different neighbors.

Despite the complexity gains of Local Conditioning, the complexity will likely still be prohibitive for many large networks. For example, on a square grid graph the complexity of Local Conditioning is commensurate with that of Cluster BP. Of course, for truly distributed settings, Local Conditioning has the advantage of involving messages between individual nodes of the network rather than requiring the clustering of nodes. However, it will be necessary to find approximate versions of Local Conditioning that involve shrinking the relevant sets for messages and beliefs. However, even in seeking approximate conditioning algorithms, we feel that a thorough understanding of the topological questions related to performing exact inference is necessary for engineering approximate versions exhibiting graceful degradation of performance.

REFERENCES

[1] M. J. Wainwright and M. I. Jordan, *Graphical models, exponential families and variational inference*, Berkeley Tech. Report 649, Sept. 2003.

[2] S.M. Aji, R.J. McCliece, "The generalized distributive law", *IEEE Trans. Info. Thy.*, vol. 46, issue 2, March 2000.

[3] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufman, June 2014.

[4] F.J. Diez, "Local Conditioning in in Bayesian Networks", *Artificial Intelligence*, vol. 87, pps. 1-20.

[5] H.J. Suermondt and G.F. Cooper, "Probabilistic Inference in Multiply Connected Belief Networks Using Loop Cutsets", *Intl. Jrnl. of Approximate Reasoning*, vol. 4, 1990, pps. 283-306.

[6] J. Pearl, "A Constraint Propagation Approach To Probabilistic Reasoning", *Uncertainty in Artificial Intelligence*, Elsevier, New York, pps. 357-369, 1986.

[7] S.L. Lauritzen and D.J. Spiegelhalter, "Local Computations with Probabilities on Graphical Structure and their Application to Expert Systems", *Jrnl. Royal Stat. Soc. B*, vol. 50, no. 2, 1988.

[8] M.G. Reyes, *Cutset Based Processing and Compression of Markov Random Fields*, Ph.D. thesis, University of Michigan, April 2011.

[9] T. Geier, F. Richter, and S. Biundo, "Locally Conditioned Belief Propagation", AUAI, 2015.

[10] R.J. Baxter, *Exactly Solved Models in Statistical Mechanics*, New York: Academic, 1982.

[11] M. G. Reyes, D. L. Neuhoff, T. N. Pappas, "Lossy Cutset Coding of Bilevel Images Based on Markov Random Fields," *IEEE Trans. Img. Proc.*, vol. 23, pp. 1652-1665, April 2014.