Robust Message-Passing for Statistical Inference in Sensor Networks

Jeremy Schiff, Dominic Antonelli, Alexandros D. G. Dimakis,
David Chu, Martin Wainwright
Department of Electrical Engineering and Computer Science
University of California, Berkeley, CA 94704
{jschiff, dantonel, adim, davidchu, wainwrig}@eecs.berkeley.edu

ABSTRACT

Large-scale sensor network applications require in-network processing and data fusion to compute useful summaries of the sensed measurements. Statistical inference has often been regarded as a particularly appropriate fit due to its ability to handle data uncertainty. In this paper we investigate the use of distributed message-passing algorithms for performing inference, in which neighboring nodes in the network convey statistical summaries of local information relevant to a global computation. We focus on the class of reweighted belief propagation algorithms, which includes as special cases the standard sum-product and max-product algorithms for general networks with cycles. The family of reweighted belief propagation (RBP) algorithms provide an approximate but practical solution to in-network inference. In contrast to the standard sum- and max-product algorithms, certain RBP algorithms have a number of attractive theoretical guarantees, including uniqueness of fixed points, convergence, and robustness.

We further design and implement a practical and modular architecture for implementing RBP algorithms in real networks and show that they are an ideal fit for sensor networks due to their distributed nature, requiring only local knowledge and coordination, and little requirements on other services such as reliable transmission. Our simulation and Mica2 mote deployment indicate that the proposed algorithms achieve accurate results despite real-world problems such as dying motes, dead and asymmetric links, and dropped messages. Finally, we also show how intelligent scheduling of RBP messages can be used to minimize communication between motes and prolong the lifetime of the network.

1. INTRODUCTION

Recent advances in hardware and software are leading to sensor network applications with increasingly large numbers of motes. In such large-scale deployments, the straightforward approach of routing all data to a common base station may no longer be feasible. Moreover, such an aggregation strategy—even when feasible—can be wasteful, since it ignores which aspects of the data are relevant (or irrelevant) to addressing a given query. Consequently, an important research challenge is the development and practical implementation of distributed algorithms for performing data fusion in an in-network manner, thereby leading to useful statistical summaries of sensed measurements [20, 1].

The theory of statistical inference [5] provides the appropriate framework for formalizing problems of this nature. In particular, this framework subsumes a broad range of tasks that arise in sensor network applications, ranging from estimation and regression (e.g., predicting a "smoothed" version of a temperature gradient) to hypothesis testing (e.g., determining whether or not a fire has occurred). Past work on sensor networks [2, 1, 3, 9, 12, 13] has established the utility of formulating such inference problems in terms of Markov random fields, a type of graphical model in which vertices represent variables of interest and edges correspond to correlations between them. Moreover, given such a graphical model, it is possible to define simple message-passing algorithms for performing inference, in which any given node passes "messages" to its neighbors that represent statistical summaries of local information relevant to a global computation. In a sensor network scenario, each mote is assigned a subset of variables of the Markov random field and we use the distributed wireless network of motes to execute the message-passing algorithm. The fact that message passing algorithms for graphical models require no global coordination translates to very simple and robust message passing algorithms for the sensor motes. The mapping of the graphical model to sensor motes and related issues are discussed in subsequent sections.

1.1 Related work and our contributions

It is well known that statistical inference for graphical models is NP-hard in general, but can be performed in exponential time using the junction tree algorithm. Paskin et al. [13] developed and implemented a robust architecture for the junction tree algorithm, suitable for performing exact inference in sensor networks. However, for realistic networks (like grids or grid-like topologies) that have multiple cycles, the complexity of the junction tree method scales exponentially in the number of nodes, so that exact inference quickly becomes intractable. In such settings where exact inference is computationally intractable, an alternative approach is provided by the belief propagation (BP)

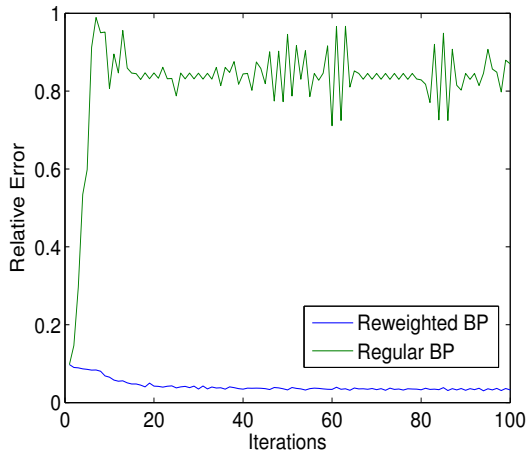

Figure 1: Graph of error versus iteration number for message-passing on a 9×9 nearest-neighbor grid. Standard belief propagation (BP, blue curve) vs. reweighted belief propagation (RBP, red curve).

algorithm, and variants thereof.[1] The BP algorithm performs exact inference for trees, but also provides reasonable approximations when applied to many networks with cycles, and has been suggested by numerous researchers [3, 12, 9] for sensor network applications. The recent survey paper by Cetin et al. [1] (and references therein) provides further discussion of the issues that arise with graphical models and message-passing in sensor networks.

In contrast to previous work, this paper focuses on the class of *reweighted belief propagation* (RBP) algorithms, a larger family of algorithms which includes as special cases the standard BP and max-product algorithms for general networks with cycles. Based on previous work [17, 15, 16, 6, 19], this class of algorithms is attractive from the theoretical perspective: in particular, certain RBP algorithms (different algorithms from standard BP) have guarantees of fixed point uniqueness, convergence, and robustness to message errors (for reweighted belief propagation), and correctness guarantees (for reweighted max-product). These properties are *not* shared by the standard BP algorithm for general networks; as illustrated in Figure 1, for certain network structures, the standard BP algorithm can yield highly inaccurate and unstable solutions to inference problems. In contrast to this instability, appropriately designed RBP algorithms are theoretically guaranteed to be robust to both message errors, and model mis-specification. A central thrust of this paper is that such issues of stability and robustness are of paramount importance in sensor network applications.

While a number of researchers [3, 12, 9] have studied the use of standard BP for sensor networks at both the theoretical and simulation level, the work described here is (to the best of our knowledge) the first to actually implement a class of loopy message-passing algorithms—including BP as one exemplar—for sensor motes. More specifically, we design and implement an architecture for implementing RBP algorithms in real sensor networks. Our design does not rely on infrastructure such as reliable messaging, time synchronization and routing, which dramatically simplifies implementation compared to other algorithms such as junction tree, while providing substantially better results compared to other message passing approaches. We present simulation results evaluating algorithm performance under the real-world problems of failing motes and communication problems of failing links, asymmetric links, and dropped messages. We also show that intelligent scheduling, with greater communication between nodes on the same mote than nodes across motes, can make the algorithms converge with much less communication. We present experimental results from a prototype implementation using Mica2 motes. Our nesC implementation is modular and can be easily adapted for any message passing algorithm and scheduling policy, which as we show, is useful for numerous applications.

[1]Belief propagation, when executed on graphs with cycles is often called the "loopy" belief propagation algorithm. There are different variants of belief propagation, including the *sum-product* updates for approximate marginalization, and the *max-product* updates for approximate maximiation. As discussed at more depth in Section 2, the collective family of algorithms are jointly referred to as message-passing algorithms.

2. BACKGROUND

We begin by providing background on graphical models, and then turn to discuss of message-passing algorithms, including the standard and reweighted sum- and max-product algorithms.

2.1 Graphical models

We focus here on Markov random field (MRF) models, which are defined in terms of an undirected graph G, with vertex set $V = \{1, \ldots, n\}$ and edge set E. Associated with each node $s \in V$ is a random variable X_s, representing (for our purposes) some type of sensor measurement or a quantity of interest, such as temperature in a particular location, which takes values either in some continuous space (e.g., $\mathcal{X} = \mathbb{R}$) or a discrete space (e.g., $\mathcal{X} = \{0, 1, \ldots, m-1\}$). The structure of the graph G constrains the nature of the statistical interactions among the collection of random variables $\vec{X} = (X_1, \ldots, X_n)$; in particular, we associate with each vertex s a function $\psi_s : \mathcal{X} \to \mathbb{R}_+$ (called single-site compatibility function), and with each edge (s,t) a function $\psi_{st} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ (called edgewise compatibility function). Under the MRF asssumption, the joint distribution of $\vec{X}$ factorizes as

$$p(\vec{x}) \quad = \quad \frac{1}{Z} \prod_{s \in V} \psi_s(X_s) \prod_{(s,t) \in E} \psi_{st}(x_s, x_t), \qquad (1)$$

where $Z = \sum_{\vec{x} \in \mathcal{X}^n} \left[\prod_{s \in V} \psi_s(X_s) \prod_{(s,t) \in E} \psi_{st}(x_s, x_t) \right]$ is a normalization factor. Although it can be convenient to also define compatibility functions on larger subsets $|S| > 2$ of nodes, there is no loss of generality in making the pairwise assumption (1).

2.2 Statistical inference in MRFs

Given an MRF of the form (1), it is frequently of interest to solve one (or more) of the following inference problems:

(a) computing the mode or maximum a posteriori (MAP) assignment, $\vec{x}^* \in \arg\max_{\vec{x} \in \mathcal{X}^n} p(\vec{x})$, corresponding to the configuration with highest probability under the model. For example, MAP estimation can determine

the most likely temperature field given a set of noisy measurements.

(b) computing the normalization constant Z defined following equation (1). Knowledge of Z is necessary for many inference problems, for example testing a hypothesis about the state of the network based on data.

(c) computing marginal probabilities at a particular node (or over some subset of nodes). As with computation of Z, such a marginalization problem involves summing over (subsets of) configurations, to determine for example, the marginal probability distribution of the temperature at some location given (possibly noisy) measurements at other locations.

All three problems are computationally challenging because the size of the search space (problem (a)) or summation (problems (b) and (c)) grows exponentially in the size of the network (more specifically, we have $|\mathcal{X}^n| = m^n$ for a discrete MRF). Indeed, all three of these problems are computationally intractable (NP-hard (a), or #P complete (b) and (c)) for general MRFs. For graphs with special structure—in particular, trees and more generally graphs of bounded treewidth[2]—all of these problems can be solved exactly with the junction tree algorithm [7], albeit with exponential complexity in treewidth. Many graphs commonly used to model sensor networks, including grid-based graphs and random geometric graphs, have unbounded treewidth, so that approximate algorithms are needed in practice.

In this paper, we focus on the problem of computing marginal probabilities (marginals).

2.3 Message-passing algorithms

In message-passing algorithms for graphical models, each node s in the MRF performs a local computation, and then transmits a summary to each of its graph neighbors. The message from node s to its neighbor $t \in N(s)$ is a vector $M_{st}(x_t)$, representing information that node t requires to perform the next round of computation. The *sum-product* form of message-passing is designed to solve inference problems (b) and (c), whereas the *max-product* algorithm applies to the MAP problem (a). For tree-structured graphs (and with modifications for junction trees), both forms of these message-passing algorithms are guaranteed to be exact. The same updates are widely applied to general graphs with cycles, in which case they provide approximate solutions to inference problems.

Here we first describe the family of reweighted belief propagation (RBP) algorithms, and then illustrate how a particular choice of edge weights yields the standard updates. Various researchers have studied and used such reweighted algorithms for the sum-product updates [8, 17, 15], generalized sum-product [18], and max-product updates [16, 6, 11, 19]. Each algorithm in this family is specified by a vector $\vec{\rho} = \{\rho, (s,t) \in E\}$ of edge weights. The choice $\rho_{st} = 1$ for all edges $(s,t) \in E$ corresponds to the standard updates; different choices of $\vec{\rho}$ yield distinct algorithms with convergence, uniqueness, and correctness guarantees [17, 16]. For any fixed set of edge weights $\vec{\rho}$, the associated reweighted BP algorithm begins by initializing all of the messages M_{st} to constant vectors; the algorithm then operates by updating the message along each edge according to the recursion

$$M_{st}(x_t) \leftarrow \sum_{x_s} \psi_s(x_s) \left[\psi_{st}(x_s, x_t)\right]^{\frac{1}{\rho_{st}}} \frac{\prod_{u \in N(s) \setminus t} [M_{us}(x_s)]^{\rho_{us}}}{[M_{ts}(x_s)]^{1-\rho_{st}}}. \quad (2)$$

These updates are repeated until the vector of messages $\vec{M} = \{M_{st}, M_{ts} \mid (s,t) \in E\}$ converge to some fixed vector $\vec{M}^*$. The order in which the messages are updated is a design parameter, and various schedules (e.g., parallel, tree-based updates, etc.) exist. Upon convergence, the message fixed point $\vec{M}^*$ can be used to compute approximations to the marginal distributions at each node and edge (inference problem (c)) via

$$q_s(x_s) \quad \propto \quad \psi_s(x_s) \prod_{t \in N(s)} [M_{ts}(x_s)]^{\rho_{st}}, \quad (3)$$

and also to generate an approximation to the normalization constant (inference problem (b)). The update (2) corresponds to the sum-product algorithm; replacing the summation $\sum_{x_s}$ by the maximization $\max_{x_s}$ yields the reweighted form of the max-product algorithm.

2.4 Theoretical guarantees for reweighted BP

The standard sum-product algorithm is a special case of the updates (2), which correspond to setting the weights $\rho_{st} = 1$ for all edges (s,t). For this choice, there are in general many fixed point solutions for the updates (2), and the final solution can depend heavily on the initialization. For suitable settings of the weights [17] different from $\rho_{st} = 1$, in contrast, the updates are guaranteed to have a unique fixed point *for any* network topology and choice of compatibility functions. Moreover, such reweighted BP algorithms are known to be *globally Lipschitz stable* in the following sense [15]. Suppose that $q(\psi)$ denotes the approximate marginals when a RBP algorithm is used for approximate inference with data (compatibility functions) ψ. Then there is a global constant L, depending only on the MRF topology, such that $\|q(\psi) - q(\psi')\| \leq L\|\psi - \psi'\|$, where $\|\cdot\|$ denotes any norm. In loose terms, this condition guarantees that bounded changes to input yield bounded output changes, which is clearly desirable when applying an algorithm to statistical data. Again, note that standard BP does not satisfy such a stability condition, due to the presence of phase transitions [14, 4].

3. PROPOSED ARCHITECTURE

In the following section, we present *StatSense*, an architecture for implementing RBP algorithms in sensor networks. Our primary focus is the challenges that arise in implementing message-passing algorithms over unreliable networks with severe communication and coordination constraints.

3.1 Mapping from graphical models to motes

As described in Section 2, any Markov random field (MRF) consists of a collection of nodes, each representing some type of random variable, joined by edges that represent statistical correlations among these random variables. A sensor network can also be associated with a type of graph, in general *different* from the MRF graph, in which each vertex represents a mote and the links between motes represent

[2]In loose terms, a graph of bounded treewidth is one which consists of a tree over clusters of nodes; see the paper [7] for further details.

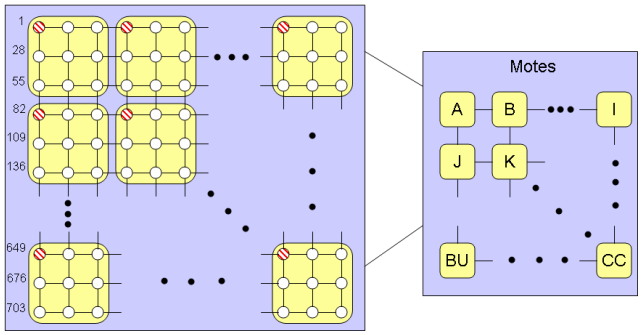

Figure 2: Illustration of the two layers of the graphical model (left-hand subplot) and motes (right-hand subplot) used for simulation. Each node in the graphical model is mapped to exactly one mote, whereas each mote contains some number (typically more than one) of nodes. The red angle-slashed dots denote observation nodes (i.e., locations with sensor readings).

communication links. As we discuss here, there are various issues associated with the mapping between the MRF graph and the sensor network graph [1].

For sensor network applications, some of MRF random variables will be associated with measurements, whereas other variables might be *hidden* random variables (e.g., temperature at a room location without any sensor, or an indicator variables for the event "Room is on fire"). Any observable node may be associated with multiple measurements, which can be summarized in the form of a histogram representing a probability distribution over the data. In this context, the role of message-passing is to incorporate this partial information in order to make inferences about hidden variables.

Each node s in the MRF is mapped to a unique mote $\Gamma(s)$ in the sensor network; conversely, each mote A is assigned some subset $\Lambda(A)$ of MRF nodes. For any node $s \in \Lambda(A)$, the mote A is responsible for handling all of its computation and communication with other nodes t in the MRF neighborhood $N(s)$. Each node $t \in N(s)$ might be mapped either to a different mote (requiring communication across motes) or to the same mote (requiring no communcation). Figure 2 illustrates one example assignment for a sensor network with 81 motes, and an MRF with 729 nodes. For instance, in this example, mote A is assigned nodes 1,2,3,26,27,28, 55, 56 and 57, so that $\Lambda(A) = \{1, 2, 3, 26, 27, 28, 55, 56, 57\}$, and $\Gamma(1) = A$ etc. The sensor associated with each mote corresponds to an evidence node in the MRF; for instance, {1, 4, . . . , 82, . . . , 673} are evidence nodes in this example and we represent them as hashed nodes in Figure 2. Similar to previous work [10, 13], we assume a semi-static topology in creating the mote communication graph, so that we place an edge in this graph if there is a high-quality wireless communication link between these two motes.

Note that the assignment of MRF nodes to motes may have a substantial impact on communication costs, since only the messages between nodes assigned to different motes need be transmitted. There is an additional issue associated with the node-mote assignment: in particular, the following property is necessary to preserve the distributed nature of message-passing when implemented on the sensor network link graph: for any pair (s, t) of nodes joined by the edge in the MRF, we require that the associated motes $\Gamma(s)$ and $\Gamma(t)$ are either the same ($\Gamma(s) = \Gamma(t)$), or are joined by an edge in the mote communication graph. We refer to this as the *no-routing* property, since it guarantees that message-passing on the MRF can be implemented in motes with only nearest-neighbor mote communication, and hence no routing. Thus, for a given MRF and mote communciation graph, a question of interest is whether the no-routing property holds, or can be made to hold. It is straightforward to construct MRFs and mote graphs for which it fails. However, the following result guarantees that it is always possible to modify the MRF so that this property holds:

Proposition 1. *Given any MRF model and connected mote communication graph, it is always possible to define an extended MRF, which when mapped onto the same mote communication graph, satisfies the no-routing property, and that message-passing on the mote communication graph yields equivalent inference solutions to the original problem.*

Proof. Our proof is constructive in nature, based on a sequence of additions of nodes to the MRF model such that:

(a) the final extended model satisfies the no-routing property, and

(b) running message-passing on the mote communication graph yields identical inferences to message-passing *with routing* in the original model.

Throughout the proof, the set of motes $A, B, C, \ldots$ and the associated mote communication graph remains fixed. The number of nodes and compatibility functions in the MRF as well as the mapping from nodes to motes are quantities that vary. Given any MRF model with variables $(X_1, \ldots, X_n)$ and mote assignments $(\Gamma(1), \ldots, \Gamma(n))$, suppose that the no-routing property fails for some pair (s, t), meaning that pair of motes $\Gamma(s)$ and $\Gamma(t)$ are distinct, and *not* joined directly by an edge in the mote link graph. Since the mote communication graph is connected, we can find some path $\mathcal{P} = \{\Gamma(s), A_2, \ldots, A_{p-1}, \Gamma(t)\}$ in the mote communication graph that joins $\Gamma(s)$ and $\Gamma(t)$. Now for each mote A_i, $i = 2, \ldots (p-1)$, we add a new random variable Y_i to the original MRF; each random variable Y_i is mapped to mote A_i (i.e., $\Gamma(Y_i) = A_i$)). Moreover, let us remove from the MRF factorization (1) the compatibility function $\psi_{st}(x_s, x_t)$, and add to it the following compatibility functions

$$\begin{aligned} \widetilde{\psi}_{s\,2}(x_s, y_2) &= \mathbb{I}[x_s = y_2] \\ \widetilde{\psi}_{(p-1)\,t}(y_{p-1}, x_t) &= \psi_{st}(y_{p-1}, x_t). \end{aligned}$$

Here the function $\mathbb{I}(a, b)$ is an indicator for the event that $\{a = b\}$. The intuition underlying this construction is that the variables $(Y_2, \ldots, Y_{p-2})$ represent duplicated copies of X_s that are used to set up a communication route between node s and t. By construction, the communication associated with each of the new compatibility functions $\widetilde{\psi}$ can be carried out in the mote graph without routing. Moreover, the new MRF has no factor $\psi_{st}(x_s, x_t)$ that directly couples X_s to X_t, so that edge (s, t) no longer violates the no-routing property. To complete the proof, we need to verify that when message-passing is applied in the mote graph associated with

the new MRF, we obtain the same inferences upon convergence (for the relevant variables $(X_1, \ldots, X_n)$) as the original model. This can be established by noting that the indicator functions $\mathbb{I}$ collapse under summation/maximization operations, so that the set of message-fixed points for the new model are equivalent to those of the original model.

Thus, we have shown that an edge (s, t) which fails the no-routing property can be disposed of, without introducing any additional links. The proof is completed by applying this procedure recursively to each troublesome MRF pair (s, t). ■

3.2 Message updating

As mentioned previously, the message update scheme is a design parameter. The most common message update scheme is one in which all nodes update and communicate their messages at every time step according to (2), and proceed synchronously to new iterations. We term this scheme *SyncAllTalk*.

Unfortunately, several factors discourage the direct application of SyncAllTalk to sensor networks. First, radio transmission and reception on embedded hardware devices consume nontrivial amounts of energy. Passing messages *inter-mote* is expensive and may overburden the already resource-constrained network. On the other hand, passing messages *intra-mote* incurs no significant energy cost because it is entirely local. This dichotomy indicates that we should limit messaging across motes when possible.

Second SyncAllTalk relies on synchronous message passing that inherently exhibits "bursty" communication patterns. For shared communication channels such as wireless, burstiness results in issues such as the *hidden terminal problem*, which further exacerbates the cost of radio transmission and reception.

Lastly, for many situations, we expect sensor informativeness to vary greatly. For example, in a building monitoring scenario, the first sensor to detect a fire will generate much more informative messages than other sensors. Other scenarios have also resulted in similar observations [13]. Thus, evidence nodes, which often correspond directly to sensors, vary greatly in informativeness. We would like to favor more informative messages and thereby accelerate the rate of convergence.

We have investigated a number of schemes which take advantage of these factors. The first scheme, *SyncConstProb*, exchanges all intra-mote messages at each time step as before, but only exchanges each inter-mote message with probability p at each time step. This means there is a direct decrease in the average period of inter-mote message transmission from once every time step to once every $1/p$ time steps. This scheme can trade convergence time for communication overhead. Probabilistic sending also reduces the burstiness that causes the hidden terminal problem.

We can further decrease burstiness by relaxing the global time constraint. Instead, each mote proceeds at its own local start time and clock rate. As an additional benefit, this scheme does not rely on any time synchronization service. This scheme is *AsyncConstProb*.

Our last scheme, *AyncPropProb*, makes the message transmission probability proportional to the informativenss of the message. Rather than reference a global constant probability of sending, p, each host sends with probability proportional to the *total variation distance* raised to a power. For messages between discrete random variables, this is

$$p_{send}(M'_{st}) = \left(\frac{1}{2} \sum_{x_t} |M'_{st}(x_t) - M_{st}(x_t)| \right)^{\eta} \qquad (4)$$

where M'_{st} denotes a new message that we may want to send from node s to node t, M_{st} denotes the previously sent message from node s to node t and η is a tunable "politeness" factor.

All of the message passing schemes we investigate are extremely simple to implement, requiring no additional service infrastructure and operating exclusively with local computation. The traditional BP literature has proposed more sophisticated schemes in which all nodes exchange messages along graph overlays in organized sequence e.g. from the root of the overlay breadth-first. Unfortunately, these schemes typically require global coordination and are thus not as readily applicable in a sensor network context.

3.3 Handling communication failure

In general, when adapting algorithms to run on sensornets, one must deal with network problems such as unreliable or asymmetric network links and mote failure. In the case of RBP, however, little adaptation is required. When a mote fails, the resulting computation is equivalent to solving the inferrence problem on a reduced graphical model with all the nodes that belonged to that mote removed. In addition, the resulting increase in error is localized near the removed nodes. If a network link fails, the resulting computation is equivalent to removing the corresponding edges from the graphical model, and the error is, again, localized. No such argument holds for asymmetric links; however, as we see in our evaluation (see Section 5), they also cause only limited error.

4. IMPLEMENTATION

In order to provide a robust and easily extensible system for performing RBP on sensornets, we implemented a general belief propagation framework in TinyOS/nesC. A diagram of our system is shown in Figure 3.

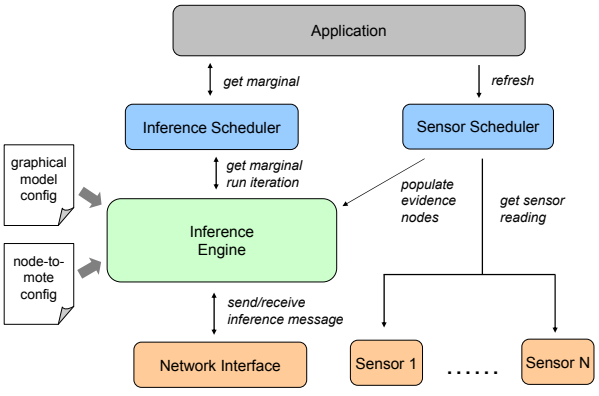

Figure 3: The StatSense architecture as implemented in nesC.

The core component of our design is the inference module. It receives incoming messages from other motes via the

network interface module, new sensor readings via the sensor scheduler module, and instructions on when to compute new messages from the inference scheduler module. When instructed to, it uses the incoming messages and sensor data to compute its outgoing messages and sends any inter-mote message to their destination via the network interface module. It handles intra-mote communication via loopback.

The network module provides very simple networks services such as buffering, marshalling and unmarshalling for incoming and outgoing messages. As a result of the robustness of RBP, we do not need any reliabile delivery. It is also the natural place to extend to more advanced network services such as data-centric multi-hop routing for cases of sophisticated node-to-mote mappings.

The sensor scheduler determines how often sensor readings are taken. Likewise, the inference scheduler determines when the inference module computes new messages according to one of the scheduling scheme detailed in Section 3.2. The inference scheduler also provides marginals of the graphical model, wrapping the functionality of the inference module.

We permit the user to input graphical models via configuration files. We created a pre-processor that converts this file into a nesC header file that contains all the data that the inference module needs to perform inference such as the compatability functions and node to mote mappings. Thus, it is straightforward for any StatSense user to build new graphical models and to use the outputs (marginals) of this model in her application.

We also architected the system in a modular fashion such that users are free to design new functional pieces independently. For example, the inference module, which currently supports BP, is easy to swap out for different message passing algorithms such as Max-Product, Min-Sum, *etc.* This allows those with a backgrounds in graphical modeling to test new algorithms without familiarity with sensornet systems issues. Likewise, it is easy to explore different scheduling schemes by replacing the inference scheduler and sensor scheduler.

5. EVALUATION

In this section, we evaluate the performance of StatSense and RBP, the main message-passing algorithm investigated in the StatSense framework. We are primarily interested in understanding: (1) the impact of sensor noise on inference results; (2) the resilience of inference in the face of changing network conditions; and (3) inference convergence under different scheduling schemes.

We used both simulation and a Mica2 mote testbed for our experimental platform. We define error to be the average over all nodes of the difference between the ground-truth reading and the mean of the distribution described by the corresponding random variable. In all places where appropriate, we use a confidence interval of 95%.

As a summary of our evaluation, we highlight:

1. StatSense with RBP shows resiliency to many types of network failures. As failures increase, error grows linearly with no sharp increases, and maintains a low absolute value.

2. Improved scheduling schemes offer substantial, and in some cases, up to 50% benefit over naive scheduling schemes.

3. Online inference in our testbed deployments exhibit accurate estimates of actual ground truth.

5.1 The Temperature Estimation Problem

In order to evaluate our system, we examine RBP's application to temperature monitoring. In simulation, we model the room as a 27x27 grid, similar to Figure 2. For each of the 100 training and 10 test runs, we randomly choose a "hot" and "cold" source placed as a 2x2 cell on this grid. We use the standard discretized heat diffusion process to calculate realistic steady-state temperatures which act as ground-truth. Since temperature has strong spatial correlations, our graphical model is a lattice, where each of the discrete locations in the room has a corresponding node in the graphical model. The edges in the model connect each node to its four nearest neighbors in a grid pattern. We discretize the temperature which ranges from 0 to 100 degrees into an eight bucket histogram.

Denote the ith observed temperature reading at location s to be $y_s^{(i)}$. Then, given M i.i.d. samples $y^{(i)} = \{y_1^{(i)}, \ldots, y_M^{(i)}\}$, we determine the compatibility functions empirically according to the standard equations:

$$\bar{\mu}_{st}(x_s, x_t) = \frac{1}{M}\sum_{i=1}^{M}\delta(x_s = y_s^{(i)})\delta(x_t = y_t^{(i)}) \quad (5)$$

$$\widehat{\psi}_s(x_s) = \frac{1}{M}\sum_{i=1}^{M}\delta(x_s = y_s^{(i)}) \quad (6)$$

$$\widehat{\psi}_{st}(x_s, x_t) = \frac{\bar{\mu}_{st}(x_s, x_t)}{\widehat{\psi}_s(x_s)\widehat{\psi}_t(x_t)} \quad (7)$$

We also arrange our motes in a 9x9 grid-pattern, and perform a straightforward mapping of each 3x3 subgraph of the graphical model onto the corresponding motes. Each mote is physically located at the top left of its subgraph and is responsible for computation of messages for the the other eight nodes assigned to it (as in Figure 2).

5.2 Resilience to Sensor Error

We begin our tests by evaluating how sensor reading error affects inference performance. We induce unbiased gaussian error to the temperature reading, with increasing standard deviation. We observe in Figure 4 that the algorithm performs well as error increases linearly, exhibiting no sharp threshold of breakdown. This is because RBP fuses data using correlation across many sensor readings, producing more robust results.

5.3 Resilience to Systematic Communication Failure

In order to test our algorithms resilience to various types of failure, we ran several experiments where we systematically increase the number of such failures. We ran three different experiments: Host Failure, Bidirectional Link Failure, and Unidirectional Link Failure. In all experiments, the we added gaussian error with a standard deviation of 20 degrees to each sensor reading.

In the Mote Failure experiment, we increase the number of motes that are unable to communicate with all other motes. This removes all nodes in the graphical model owned by any "dead" hosts. We limit our error computation to nodes that are on live hosts. The results for the Mote Failure experiment are shown in Figure 5. The figure illustrates

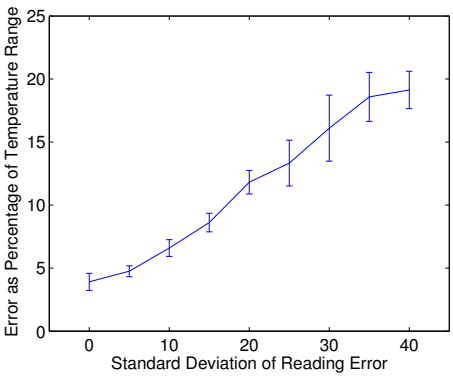

Figure 4: Average error as the gaussian error applied to the sensor observations increases.

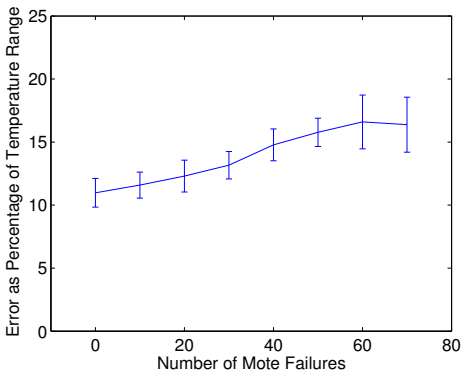

Figure 5: Average error as the number of dead motes increases.

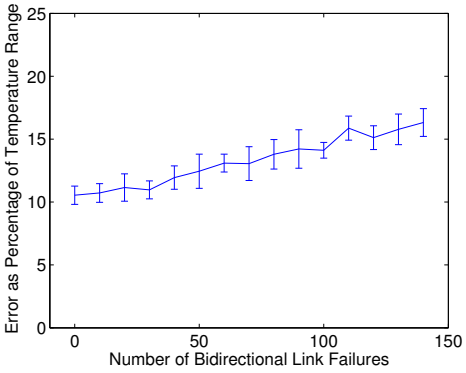

Figure 6: Average error as the number of dead symmetric links increases.

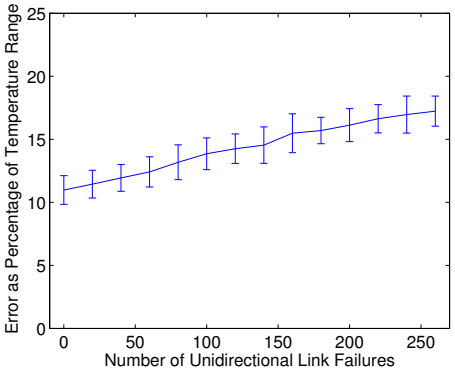

Figure 7: Average error as the number of dead asymmetric links increases.

that while error grows gradually as hosts die, the standard deviation of error increases dramatically. As more motes die, they become much more reliant on their own readings, and thus are more susceptible to unreliable readings.

In the Bidirectional Link Failure experiment, we gradually sever an increasing number of links in the network connectivity graph. We currently restrict an edge in the graphical model to require a link in the network connectivity graph. Potentially, multicast routing and dynamic detection of link failures could overcome these errors, at the cost of more messages. Instead, severing a link between two hosts removes all edges in the graphical model for nodes on the first host that are connected to nodes to the other host, and vice versa. We show the results for this experiment in Figure 6, which shows that the error increases linearly as we sever links.

In the Unidirectional Link Failure experiment, we gradually cause more links in the network graph to only allow communication in one direction. This reflects asymmetric communication problems which are experienced in typical sensor-network deployments. The results in Figure 7 are nearly identical to those in Figure 6. Therefore, we conclude that our algorithm is resiliant to unidirectional link failure.

5.4 Resilience to Transient Failure

The algorithm is very resilient to transient failures. We induce the equivalent of transient failures in the AsyncConstProb scheduling algorithm, whereby motes randomly do not transmit at every iteration. The magnitude of transience will just modify the number of iterations required before convergence. Thus, the algorithm does not need to rely on any robust messaging protocols. If a transmission does not get through, resending later will be sufficient for correctness.

5.5 Scheduling

For scheduling, we experiment with AsyncConstProb and AyncPropProb, evaluating the average error over all differences between the mean of the inferred random variable and the corresponding value of the diffusion process ground truth. We also evaluate how these algorithms affect the number of transmissions required for convergence.

As was previously discussed in the Section 5.4, and as illustrated in Figure 8, the error is not affected by changes in the probability that inter-mote messages are sent (or get through). Interestingly, the number of messages sent does not monotonically increase as the probability of sending increases. For extremely low probabilities, the chance that the important message, which we need to send to cause convergence, also has a low probability transmission. As a result, irrelevant messages get sent, increasing overall transmissions

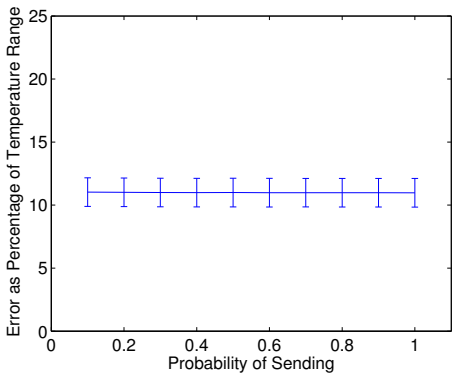

Figure 8: Average error as the probability of sending in a single iteration for SyncConstProb increases.

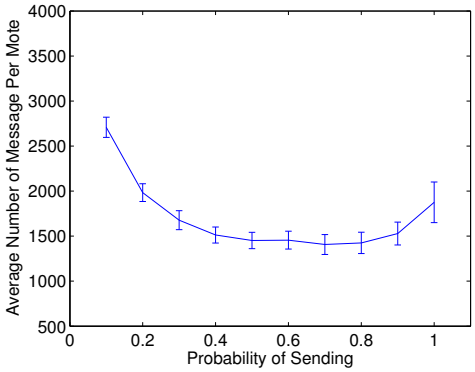

Figure 9: Number of messages sent as the probability of sending in a single iteration increases for SyncConstProb.

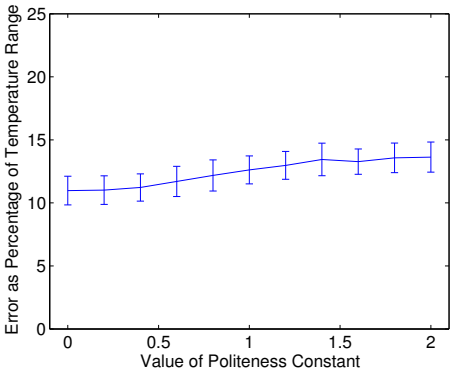

Figure 10: Average error as the exponent of the total variation difference between old and new messages increases for AyncPropProb.

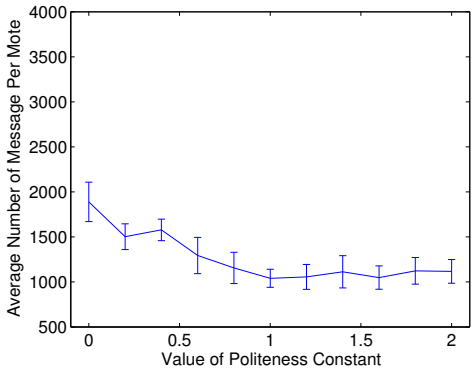

Figure 11: Number of messages sent as the exponent of the total variation difference between old and new messages increases for AyncPropProb.

without accelerating convergence. For sufficiently low probabilities, these irrelevant messages overwhelm the communication savings by transmitting less frequently, as can be observed in Figure 9. This motivates the investigation of AyncPropProb.

By changing the politeness constant η for AyncPropProb as described in 4, we control the likelihood of transmission as a function of how informative the message is, as defined by the total variation distance. Setting $\eta = 0$, causes the algorithm to degenerates to AsyncAllTalk. As we increase η, we decrease the likelihood that the message will be transmitted for the same difference in messages between the last-sent and current message. Figure 10 illustrates that the amount of error incurred is minimal, while Figure 11 shows that we save approximately 50% of the messages, even compared to the minimum value of AsyncConstProb. Another interesting property is that the number of messages is monotonically decreasing, as opposed to simple AsyncConstProb, which is convex. An advantage of AsyncConstProb is that modifying the politeness constant η directly trades time to convergence for the number of messages sent, while a sufficiently small choice of the probability p to send in AsyncConstProb can increase convergence time while also increasing messages sent.

5.6 Deployment Experiment

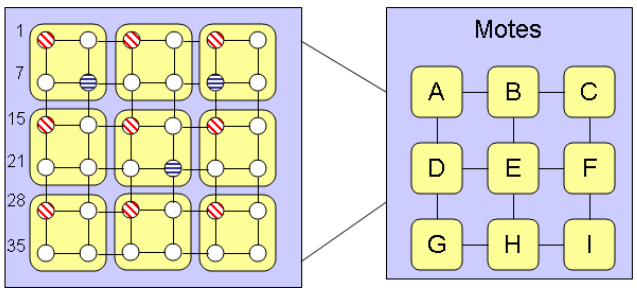

Figure 12: This figure illustrates the two layers of the Graphical Model (left) and Motes (right) which we used for our in-lab experiment. The red angle-slashed dots denote observation nodes, ie locations where we have sensor readings, and the blue-horizontal nodes denote locations where we logged unobserved temperature readings to determine error.

Node	actual temp	mean of marginal
7	36.8°C	34.7°C
10	66.0°C	55.5°C
21	33.7°C	35.1°C

Table 1: Comparison between actual and inferred temperatures for three nodes in the graphical model

In our testbed, we inferred the temperature of a room in unobserved locations through RBP. We reduced the size of our graphical model to be a 6x6 node lattice, with each 2x2 section assigned to a host and the mote's true location again corresponding to the top-left node of its section. This setup yielded a deployment of 9 motes in a 3x3 grid where each mote was 6 inches apart from its closest neighbors as we show in Figure 12. To create temperature gradients, we used two 1500 watt space heaters. To calculate our initial compatibility functions, we used the same 9 motes, but at a distance of 3 inches.

Similarly to the simulated experiments, we placed the heaters in 10 different setups, and used the collected temperature data to determine the compatibility functions according to equations (5), (6), and (7). We then ran our test by placing both space-heaters in the top right corner of our lab, and running belief propagation to infer the temperatures of the unobserved nodes. We compare the inferred temperatures at unobserved locations with the temperatures determined using 3 additional "spying" motes which collected direct measurements while the experiment ran, but were not involved in any information provided to the 9 belief propagation motes.

We plot the distributions of the three spying motes in Figures 13 ,14, and 15, which are discretized as histograms. We can see that the confidence of motes in their unobserved variables varies greatly, for instance there is very low variance in Node 7 (Figure 13), while significantly more variance in Node 10 (Figure 14). The red bar indicates the bucket in which the ground truth lies. The figures illustrate that the mean of this distribution well approximates the actual sensor reading. Interestingly, the distribution for Node 10 is bimodal. This is because when learning the model, the system never observed adjacent nodes both resolving to a temperature bucket of three. In this test, one of the readings next to Node 10 was 3, which resulted in the bimodal distribution.

Table 1 shows the actual and inferred temperatures for these three nodes. The reason that the error for node 10 is so large is that the actual temperature is higher than any we saw during calibration, and therefore, the quantized values do not have enough dynamic range to reach that high of a temperature. The other two nodes do not have this problem, and we can see that the inference is correct to within 2°C.

Lastly, we ran experiments to validate our simulations. We ran inference on the mote deployment with the model we learned from the real temperature readings. We then ran inference in simulation using the same graphical model and the sensor readings obtained from the motes. Figure 16 shows the expected value of the temperatures at each grid location when we ran the inference on the deployment, and Figure 17 shows the expected value of the temperatures at each grid location when we ran the inference in the simulation. As we can see, they are almost identical. In fact, the maximum difference between the two at any location is 3.2°C and the average difference is 0.8°C. The small differences can be explained by differences in message ordering and packet loss in the real deployment.

6. FUTURE WORK AND CONCLUSIONS

Currently, our architecture does not consider dynamic models that exploit time-correlations. This extension could be useful for applications such as tracking and our work can be extended to exploit such temporal correlations. Addressing mote mobility is another issue we plan to explore especially the implications it might have in the dynamic mapping of nodes to motes

We presented a general architecture for using message passing algorithms for inference in sensor networks, using reweighted belief propagation. We demonstrate that RBP is robust to communication and node failures and hence constitutes an effective fit for sensor network applications. The robustness of our architecture is demonstrated in simulations and real mote deployment. An important feature of the proposed scheme is that it does not rely on a network layer to provide multi-hop routing and that our architecture provides meaningful results even under severe noise in measurements or network partitions. Note that, even though we show theoretically that any graphical model can be mapped to motes without requiring routing, in practice, some long-range correlations might introduce additional variables. This can be circumvent by simply ignoring the long-range links. In our temperature experiments, we found that no such long-range correlation edges existed. We therefore believe that our architecture will be useful for many applications that involve statistical inference or data uncertainty in sensor networks.

7. REFERENCES

[1] M. Cetin, L. Chen, J. W. Fisher, A. T. Ihler, R. L. Moses, M. J. Wainwright, and A. S. Willsky. Distributed fusion in sensor networks. *IEEE Signal Processing Magazine*, 23:42–55, 2006.

[2] L. Chen, M. J. Wainwright, M. Cetin, and A. Willsky. Multitarget-multisensor data association using the tree-reweighted max-product algorithm. In *SPIE Aerosense Conference*, April 2003.

[3] C. Crick and A. Pfeffer. Loopy belief propagation as a basis for communication in sensor networks. In *Uncertainty in Artificial Intelligence*, July 2003.

[4] A. Ihler, J. Fisher, and A. S. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6:905–936, May 2005.

[5] K. Knight. *Mathematical Statistics*. Chapman and Hall, New York, 2000.

[6] V. Kolmogorov. Convergent tree-reweighted message-passing for energy minimization. *IEEE Trans. PAMI*, 2006. To appear.

[7] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society B*, 50:155–224, January 1988.

[8] A. Levin and Y. Weiss. Learning to combine bottom-up and top-down segmentation. In *European Conference on Computer Vision (ECCV)*, June 2006.

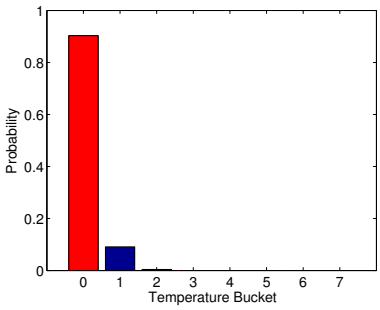

Figure 13: This is the histogram of the inferred distribution on Node 7 of our graphical model. The red bar is the bucket for the actual sensor reading.

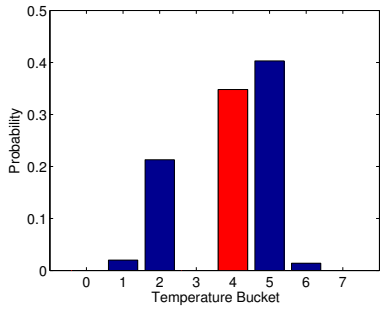

Figure 14: This is the histogram of the inferred distribution on Node 10 of our graphical model. The red bar is the bucket for the actual sensor reading.

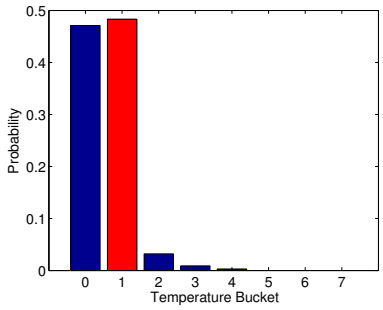

Figure 15: This is the histogram of the inferred distribution on Node 21 of our graphical model. The red bar is the bucket for the actual sensor reading.

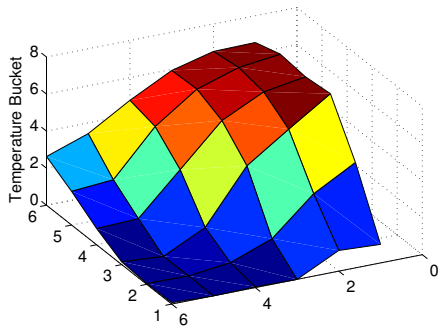

Figure 16: Temperature field for the experiment run on the deployment

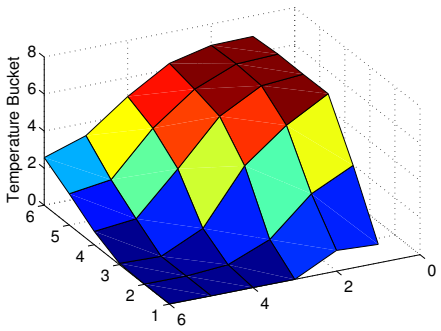

Figure 17: Temperature field for the experiment run in simulation

[9] E. C. Liu and J. M. F. Moura. Fusion in sensor networks: convergence study. In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing*, volume 3, pages 865–868, May 2004.

[10] A. Meliou, D. Chu, C. Guestrin, J. Hellerstein, and W. Hong. Data gathering tours in sensor networks. In *Information Processing in Sensor Networks (IPSN)*, April 2006.

[11] T. Meltzer, C. Yanover, and Y. Weiss. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *International Conference on Computer Vision*, June 2005.

[12] J. M. F. Moura, J. Lu, and M. Kleiner. Intelligent sensor fusion: a graphical model approach. In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing*, April 2003.

[13] M. Paskin, C. Guestrin, and J. McFadden. A robust architecture for distributed inference in sensor networks. In *Information Processing in Sensor Networks (IPSN)*, April 2005.

[14] S. Tatikonda and M. I. Jordan. Loopy belief propagation and Gibbs measures. In *Proc. Uncertainty in Artificial Intelligence*, volume 18, pages 493–500, August 2002.

[15] M. J. Wainwright. Estimating the "wrong" graphical model: Benefits in the computation-limited regime. *Journal of Machine Learning Research*, pages 1829–1859, September 2006.

[16] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Exact MAP estimates via agreement on (hyper)trees: Linear programming and message-passing. *IEEE Trans. Information Theory*, 51(11):3697–3717, November 2005.

[17] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. *IEEE Trans. Info. Theory*, 51(7):2313–2335, July 2005.

[18] W. Wiegerinck. Approximations with reweighted generalized belief propagation. In *Workshop on Artificial Intelligence and Statistics*, January 2005.

[19] C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation: An empirical study. *Journal of Machine Learning Research*, 7:1887–1907, September 2006.

[20] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich. Collaborative Signal and Information Processing: an Information-Directed Approach. *IEEE Trans. Commun.*, 91(8):1199–1209, August 2003.