Projected Belief Propagation: A Sum-Product Alternative for Severely Resource Constrained Applications

Abstract

We propose Projected Belief Propagation, an alternative to Sum-Product belief propagation that has reduced requirements for computation and communication per iteration. This is accomplished by strategic partial updating of the beliefs in the network. We present theoretical results that specify when Projected BP inherits exponentially fast convergence properties from Sum-Product BP, assuming such is the case for Sum-Product. We show convergence with respect to various norms, making standard assumptions on the convergence of Sum-Product, as well as finite time convergence in trees.

1. Introduction

Algorithms on graphs are important in many decision making, inference, and detection tasks. Belief Propagation (BP), one example being Sum-Product BP, is an example of an algorithmic approach to computations on graphs that has applications in diverse areas such as communications, signal processing, and artificial intelligence (Kschischang et al., 2001). Belief propagation provides the advantages of distributed computation and fast approximation for hard inference problems. However, further reductions in the computational complexity or communication overhead of the algorithm may be possible, beyond a basic flooding Sum-Product implementation of BP. One technique for increasing the efficiency of computation is Residual BP (Elidan et al., 2006), which involves prioritizing belief updates according to the most recent change of the inputs to the computation of that belief. There has also been some amount of work on alternatives to the basic Sum-Product algorithm that specifically targets applications with strict low power requirements, such as sensor networks. One example is the Stochastic BP Algorithm (Noorshams & Wainwright, 2013), which reduces both the computational cost of an iteration of the algorithm and the communication overhead at the expense of convergence rate. In (Çetin et al., 2006), an overview is given of work toward accounting for the particular issues that arise in using belief propagation for information fusion in sensor networks. Much of the focus of that paper is on methods for networks with communication constraints, such as particle-based messaging, message censoring, and message approximation (typically from quantization). It may also be desirable to develop alternatives to belief propagation that are robust to computation/communication errors, or to otherwise understand how robust belief propagation is in a particular application with such errors. The small amount of work in this direction includes analysis of LDPC decoding subject to errors (Varshney, 2011; Tabatabaei Yazdi et al., 2013), the error-resilient Markov random field message passing architecture for stereo matching (Choi et al., 2013), and analysis of belief propagation subject to certain types of messaging errors (Ihler et al., 2005).

In this paper, we develop a novel algorithm for the approximation of marginals of functions representable by a factor graph. This algorithm is designed as a replacement for Sum-Product in applications with strict computation and communication resource constraints. In Section 2, we give background on the Sum-Product algorithm and discuss the potential areas for increased efficiency. In Section 3, we present our alternative belief propagation algorithm, Projected BP. We also present theoretical results about the fixed points and convergence properties of the algorithm. These results include conditions for exponential convergence in general graphs, and convergence in finite iterations for trees. Finally, we give concluding remarks in Section 5, and discuss some potential topics for future investigation.

2. Basics of the Sum-Product Algorithm

We consider the design of alternatives to the Sum-Product algorithm that are more efficient, with respect to both computation and communication, for the situation where all variables live in finite sets and the kernels at the function nodes are bounded above zero, but otherwise arbitrary (i.e. non-parametric). We begin by reviewing Sum-Product BP in this scenario of interest.

Let $v \in \mathcal{V} = \{1, ..., |\mathcal{V}|\}$ be the variable nodes, let $f \in \mathcal{F} = \{1, ..., |\mathcal{F}|\}$ be the function nodes, and let

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

$e \in \mathcal{E} \subset \mathcal{V} \times \mathcal{F}$ be the undirected edges in a bipartite graph. Associate with each variable node v a variable $X_v \in \mathcal{X}_v$ where $D \triangleq |\mathcal{X}_v|$. We have defined every $\mathcal{X}_v$ to be the same size for simplicity. Extending to the case of variables living in finite sets of varying sizes would be a trivial matter. Now, associate with each function node f a kernel function

$$\psi_f : \prod_{v:(v,f)\in\mathcal{E}} \mathcal{X}_v \to \mathbb{R}^+,$$

i.e., $\psi_f(\cdot)$ is a function mapping the variables of the nodes neighboring f to strictly positive real numbers. We make the assumption of strict positivity in this work, but this is not generally a necessary condition. For convenience, let $\mathcal{N}_v \subset \mathcal{F}$ be the neighbors of v and let $\mathcal{N}_f \subset \mathcal{V}$ be the neighbors of f. As a slight abuse of notation, we may use $\mathcal{S} = \prod_{v\in\mathcal{S}} \mathcal{X}_v$ for $\mathcal{S} \subset \mathcal{V}$. Therefore, we have that the bipartite graph, which we call a *factor graph*, is a graphical representation of the global function

$$\Psi(\mathcal{V}) = \Psi(X) = \prod_{f\in\mathcal{F}} \psi_f(\mathcal{N}_f),$$

where $X = (X_1, ..., X_{|\mathcal{V}|})$. Often, the factor graph is meant to represent a joint probability distribution over the variables X_v, $v \in \mathcal{V}$. In this case, we have that

$$\mathrm{P}_X(\mathcal{V}) = \mathrm{P}(X) \propto \prod_{f\in\mathcal{F}} \psi_f(\mathcal{N}_f).$$

Inference within the factor graph often involves computing the single variable marginal distributions

$$\mathrm{P}_{X_v}(i) = \sum_{x:x_v=i} \mathrm{P}_X(X = x).$$

Approximating these marginal distributions is the objective of the Sum-Product belief propagation algorithm, and this is the problem we study in this work.

The Sum-Product algorithm iteratively updates messages over the edges of the graph. Let $\mu^t_{v\to f}(X_v)$ be a message from variable node v to function node f in iteration t, and let $\theta^t_{f\to v}(X_v)$ be a message from function node f to variable node v in iteration t. The Sum-Product algorithm with a flooding messaging schedule is given in Algorithm 1. In this algorithm, we have that

$$\begin{aligned}&\mathrm{Marginal}(f \to v)\\ &\quad = \sum_{X_u:u\in\mathcal{N}_f\backslash v} \psi_f(\mathcal{N}_f) \prod_{w\in\mathcal{N}_f\backslash v} \mu^{t-1}_{w\to f}(X_w), \end{aligned} \qquad (1)$$

and Z is a normalizing constant to ensure that the respective result sums to 1. It is not necessary in a tree graph.

Note that the computational complexity of the function to variable message update in Equation (1) is $\mathcal{O}(D^{|\mathcal{N}_f|})$. In some applications, the function node kernels $\psi_f(\cdot)$ have structure that allow simplifications that lead to computational savings in this computation. However, in this work we consider the general case, which does not allow such computational savings. Also, note that each of the messages involves the transfer of D (for function to variable node messages) or $D - 1$ (for variable to function node messages) real numbers, and this may be prohibitive if D is large or if communication is severely constrained. Finally, we note that this presentation of the Sum-Product is for a flooding messaging schedule. Of course, other schedules for updating message in the graph are possible, and this has been extensively studied (Kfir & Kanter, 2003; Elidan et al., 2006; Goldberger & Kfir, 2008; Sutton & McCallum, 2007).

Algorithm 1 Sum-Product Belief Propagation

```
1: Data: $\mathcal{V}$, $\mathcal{F}$, $\mathcal{E}$, $\psi_f(\cdot)$ for each $f \in \mathcal{F}$
2: Result: $\hat{\mathrm{P}}_{X_v}(i)$ for each $v \in \mathcal{V}$
3: Initialize $\mu^0_{v\to f}(X_v) = \frac{1}{D}$ for each $(v,f) \in \mathcal{E}$. $t = 0$
4: repeat
5:   $t \leftarrow t + 1$
6:   for all $(v,f) \in \mathcal{E}$ do
7:     if $|\mathcal{N}_f| = 1$ then
8:       $\theta^t_{f\to v}(X_v) = \psi_f(X_v)$
9:     else
10:      $\theta^t_{f\to v}(X_v) = \mathrm{Marginal}(f \to v)$
11:    end if
12:  end for
13:  for all $(v,f) \in \mathcal{E}$ do
14:    if $|\mathcal{N}_v| = 1$ then
15:      $\mu^t_{v\to f}(X_v) = \frac{1}{D}$
16:    else
17:      $\mu^t_{v\to f}(X_v) = \frac{1}{Z}\prod_{g\in\mathcal{N}_v\backslash f} \mu^t_{g\to v}(X_v)$
18:    end if
19:  end for
20: until some stopping condition
21: return $\hat{\mathrm{P}}_{X_v}(i) = \frac{1}{Z}\theta^t_{f\to v}(i)\mu^t_{v\to f}(i)$ for each $v \in \mathcal{V}$
    and any $f \in \mathcal{N}_v$
```

3. Projected Belief Propagation

The belief propagation algorithms we will develop are what we call Projected BP algorithms. On a high level, these are most closely related to the Residual BP algorithm from (Elidan et al., 2006). This is because both Residual BP and Projected BP essentially involve intelligently selecting small subsets of the algorithm's state space to update or transmit in each iteration. In the case of Residual BP, the granularity is on the level of messages, where we choose to update a message if the portion of state that the update depends on has changed significantly since the last time that message was updated. What results is a message update schedule that prioritizes updating the messages with inputs that have changed the most since the last update of the message. This reduces the amount of computation by computing only the high priority updates, but it has the additional benefit of reducing the amount of data transferred between nodes in the network, because a message does not need to be transferred if that message was not updated in that iteration. Of course, this message update prioritization scheme involves some amount of global coordination within the network. In Projected BP, the granularity of the state subset selection is much finer, on the level of the individual components of the beliefs that are passed between nodes in the graph. In this way, we can realize greater computational and communications efficiency without the need for any global coordination. This makes our algorithms better suited to resource constrained distributed networks, where any kind of global coordination is difficult, if not infeasible. However, the computational benefits still extend to centralized uses of Projected BP.

3.1. Description of Projected Belief Propagation

Projected BP simply involves choosing a subset of elements from each variable to function message to send to the function nodes. The description is given as Algorithm 2. In line 10 of Algorithm 2, the expression Marginal$(f \to v)$ differs from Equation (1) only in that $\mu^{t-1}_{w\to f}(X_w)$ is replaced with $\hat{\mu}^{t-1}_{w\to f}(X_w)$. Again, Z is a normalizing constant to ensure that the respective result sums to 1.

There are two important differences between Projected BP and Sum-Product BP. First, we point out lines 20-23 of Algorithm 2. Here, we see that the algorithm state vectors $\hat{\mu}_{v\to f}$ are not updated with the full variable-to-function beliefs $\mu_{v\to f}$. Instead, only a subset of the entries take on the values in $\mu_{v\to f}$, while the rest remain unchanged. Because of this, if the sizes of the subsets $\mathcal{U}^t_{v\to f}$ are much smaller than D, then this potentially represents large savings in communications overhead per iteration for the variable-to-function updates. Note that it is possible to use the same communication saving method for the function to variable messages, but this does not give a corresponding significant computational savings, since the computation is dominated by the updates of each $\hat{\mu}_{f\to v}$. (The communications savings may nevertheless still be desirable.) This is broadly similar to Stochastic BP (Noorshams & Wainwright, 2013), with the most important difference being that our careful deterministic choice of information to send in the message is much more informative than sending a random value sampled from the distribution $\mu^t_{f\to v}(X_v)$.

The other important difference is on line 10 of Algorithm 2. First, the full variable to function messages $\mu^t_{v\to f}(X_v)$ are not available for the updates of the function to variable messages $\theta^t_{f\to w}(X_w)$. Only the estimates $\hat{\mu}^t_{v\to f}$ are

Algorithm 2 Projected Belief Propagation

```
1:  Data: $\mathcal{V}$, $\mathcal{F}$, $\mathcal{E}$, $\psi_f(\cdot)$ for each $f \in \mathcal{F}$
2:  Result: $\hat{\mathrm{P}}_{X_v}(i)$ for each $v \in \mathcal{V}$
3:  Initialize $\hat{\mu}^0_{v\to f}(X_v) = \frac{1}{D}$ for each $(v,f) \in \mathcal{E}$. $t = 0$
4:  repeat
5:    $t \leftarrow t+1$
6:    for all $(v,f) \in \mathcal{E}$ do
7:      if $|\mathcal{N}_f| = 1$ then
8:        $\theta^t_{f\to v}(X_v) = \psi_f(X_v)$
9:      else
10:       enforce $\theta^t_{f\to v}(X_v) = $ Marginal$(f \to v)$
11:     end if
12:   end for
13:   for all $(v,f) \in \mathcal{E}$ do
14:     if $|\mathcal{N}_v| = 1$ then
15:       $\mu^t_{v\to f}(X_v) = \frac{1}{D}$
16:     else
17:       $\mu^t_{v\to f}(X_v) = \frac{1}{Z}\prod_{g\in\mathcal{N}_v\setminus f}\mu^t_{g\to v}(X_v)$
18:     end if
19:   end for
20:   for all $(v,f) \in \mathcal{E}$ do
21:     choose $\mathcal{U}^t_{v\to f}$ where $\mathcal{U}^t_{v\to f} \subseteq \mathcal{X}_v$
22:     $\hat{\mu}^t_{v\to f}(i) = \begin{cases} \mu^{t-1}_{v\to f}(i) & \text{if } i \in \mathcal{U}^t_{v\to f} \\ \hat{\mu}^{t-1}_{v\to f}(i) & \text{if } i \notin \mathcal{U}^t_{v\to f} \end{cases}$
23:   end for
24: until some stopping condition
25: return $\hat{\mathrm{P}}_{X_v}(i) = \frac{1}{Z}\theta^t_{f\to v}(i)\mu^t_{v\to f}(i)$ for each $v \in \mathcal{V}$
    and any $f \in \mathcal{N}_v$
```

available. Plus, instead of simply computing the update Marginal$(f \to v)$ in full, we will make use of the fact that the messages $\hat{\mu}^{t-1}_{w\to f}(X_w)$ differ from $\hat{\mu}^{t-2}_{w\to f}(X_w)$ only in the elements specified by $\mathcal{U}^t_{w\to f}$. This is why we specify that we *enforce* the equality in that line of pseudocode.

To see how this enforcement is done with reduced computational complexity, consider a function node f with only two neighbors $\mathcal{N}_f = \{v1, v2\}$. The full update of $\theta^t_{f\to v_2}(X_{v_2})$ would be

$$\theta^t_{f\to v_2}(X_{v_2}) = \sum_{X_{v_1}} \psi_f(X_{v_1}, X_{v_2})\hat{\mu}^{t-1}_{v_1\to f}(X_{v_1}).$$

However, we have that

$$\theta^{t-1}_{f\to v_2}(X_{v_2}) = \sum_{X_{v_1}} \psi_f(X_{v_1}, X_{v_2})\hat{\mu}^{t-2}_{v_1\to f}(X_{v_1}),$$

and $\hat{\mu}^{t-1}_{v_1\to f}(X_{v_1}) - \hat{\mu}^{t-2}_{v_1\to f}(X_{v_1}) = 0$ for $X_{v_1} \notin \mathcal{U}^t_{v_1\to f}$. Therefore, we have that

$$\begin{aligned}\Delta\theta^t_{f\to v_2}(X_{v_2}) \\ = \sum_{X_{v_1}\in\mathcal{U}^t_{v_1\to f}} \psi_f(X_{v_1}, X_{v_2})(\Delta\hat{\mu}^{t-1}_{v_1\to f}(X_{v_1})),\end{aligned} \tag{2}$$

where $\Delta\theta^t_{f\to v_2}(X_{v_2}) = \theta^t_{f\to v_2}(X_{v_2}) - \theta^{t-1}_{f\to v_2}(X_{v_2})$ and $\Delta\hat{\mu}^{t-1}_{v_1\to f}(X_{v_1}) = \hat{\mu}^{t-1}_{v_1\to f}(X_{v_1}) - \hat{\mu}^{t-2}_{v_1\to f}(X_{v_1})$. Thus, the update is simply $\theta^t_{f\to v_2}(X_{v_2}) \leftarrow \theta^{t-1}_{f\to v_2}(X_{v_2}) + \Delta\theta^t_{f\to v_2}(X_{v_2})$, which is accomplished with a fraction $|\mathcal{U}^t_{v_1\to f}|/D$ of the full Sum-Product update complexity. This update simplification generalizes easily when the function node degree is more than 2. The only time this simplification is not possible is during the first iteration, when it is necessary to initialize every $\theta^t_{f\to v}(X_v)$ with the full computation of Marginal$(f \to v)$. Of course, this is equivalent to a Sum-Product update of $\theta^t_{f\to v}(X_v)$, so is not a disadvantage of Projected BP compared to Sum-Product.

3.2. Subset Selection Methods

We will consider two versions of Projected BP, where the difference is in the method of selecting the update subsets $\mathcal{U}^t_{v\to f}$. The first will be called K-Projected BP. For this, we choose the set $\mathcal{U}^t_{v\to f}$ as a size $K \le D$ subset of indices to elements of $\hat{\mu}^{t-1}_{v\to f}(X_v)$ to update to produce $\hat{\mu}^t_{v\to f}(X_v)$. Specifically, we construct $\mathcal{U}^t_{v\to f}$ from K elements of $\mathcal{X}_v$ so that for $i \in \mathcal{U}^t_{v\to f}$ and $j \notin \mathcal{U}^t_{v\to f}$ we have that

$$|\hat{\mu}^{t-1}_{v\to f}(i) - \mu^t_{v\to f}(i)| \ge |\hat{\mu}^{t-1}_{v\to f}(j) - \mu^t_{v\to f}(j)|.$$

In other words, choose the indices where $\hat{\mu}^{t-1}_{v\to f}(\cdot)$ and $\mu^t_{v\to f}(\cdot)$ differ the most.

The other Projected BP variant will be called β-Projected BP. This involves selecting subsets of varying sizes in order to ensure that the message estimates $\hat{\mu}^t_{v\to f}(X_v)$ are within a certain relative distance of $\mu^{t-1}_{v\to f}(X_v)$. Specifically, choose a value $\beta \in [0,1)$ that indicates, for some norm N, how small the residual difference $\|\hat{\mu}^t_{v\to f}(X_v) - \mu^t_{v\to f}(X_v)\|_N$ should be compared to the size of the desired update $\|\hat{\mu}^{t-1}_{v\to f}(X_v) - \mu^t_{v\to f}(X_v)\|_N$. Therefore, we refer to this algorithm as β-Projected BP. Hence, we have that

$$\mathcal{U}^t_{v\to f} = \arg\min\nolimits_{\mathcal{U}} |\mathcal{U}|$$

subject to

$$\frac{\|\hat{\mu}^t_{v\to f}(X_v)-\mu^t_{v\to f}(X_v)\|_N}{\|\hat{\mu}^{t-1}_{v\to f}(X_v)-\mu^t_{v\to f}(X_v)\|_N} \le \beta.$$

Note that for both K-Projected BP and β-Projected BP, the messages $\mu^t_{v\to f}(X_v)$ comprise the result of computing a Sum-Product update, starting with the message estimates $\hat{\mu}^{t-1}_{v\to f}(X_v)$. Furthermore, note that for many norms, this discrete optimization is an easy operation.

3.3. Computation and Communication Complexity

We now examine more carefully the complexity of our Projected BP algorithms, in terms of both computation and communication, and compare this to Sum-Product. Specifically, we are interested in how the complexity scales with the variable cardinality D and the size of the subsets $\mathcal{U}^t_{v\to f}$. We will focus on K-Projected BP, such that $|\mathcal{U}^t_{v\to f}| = K$.

First, note that the complexity of the updates to $\mu^t_{v\to f}(X_v)$ are simply $\mathcal{O}(D)$ for both Sum-Product and Projected BP. The significant difference will be in the updates of $\theta^t_{f\to v}(X_v)$. For Sum-Product, the computations for $\theta^t_{f\to v}(X_v)$ are as follows: We must compute D elements of $\theta^t_{f\to v}(X_v)$. Each of these involves a summation of $D^{(|\mathcal{N}_f|-1)}$ elements, each of which involves $|\mathcal{N}_f| - 1$ multiplies. In total, we have $D \times D^{(|\mathcal{N}_f|-1)} \times (|\mathcal{N}_f| - 1) = \mathcal{O}(D^{|\mathcal{N}_f|})$. Now, over the whole graph, we may conclude that the overall computational complexity of a the Sum-Product iteration, with respect to the variable cardinality D for a particular graph topology, is dominated by the function to variable message updates, implying that the overall iteration complexity is $\mathcal{O}(D^{N_{\max}})$, where $N_{\max}$ is the maximum function node degree.

Now, consider the computational complexity of the updates of the function to variable beliefs $\theta^t_{f\to v}(X_v)$ in the K-Projected BP algorithm. We can see in the special case of Equation (2) that we have computational complexity $\mathcal{O}(DK)$ instead of $\mathcal{O}(D^2)$. In the general case for function nodes of potentially higher degree, we have a computational complexity of $\mathcal{O}(KD^{(N_{\max}-1)})$. In updating these higher degree function node messages, the procedure involves separately applying an update of the form in Equation (2), each of which has complexity $\mathcal{O}(KD^{(N_{\max}-1)})$. Apart from this, the computations are nearly the same. As such, the difference in scaling constants for these orders of computational complexity is nearly the factor of $(|\mathcal{N}_f|-1)$ – the number of successive updates involved in a full K-Projected update of $\theta^t_{f\to v}(X_v)$. Therefore, when we compare the complexity of Sum-Product BP to that of K-Projected BP, we see that we are able to save a potentially large factor $\frac{D}{K(N_{\max}-1)}$ of computation per iteration by using the K-Projected BP algorithm. The only exception is the setup for the first iteration of K-Projected BP and β-Projected BP, which is essentially equivalent to one Sum-Product update. As we will examine more closely later, there are a number of applications where D can be quite large (say, over 50), and both K and $(N_{\max} - 1)$ may each be as little as 1.

With respect to communication complexity in a distributed network, this depends on the architecture of the nework and how the nodes of the factor graph are mapped to physical nodes of the network. However, take the example of a network where each sensor is represented by a variable, and the relationships among the nodes are all pairwise. Then the messages that are passed between sensors could correspond with the variable to function messages. Then the only communication over the resource constrained links for Sum-Product would be to transfer $\mu^t_{v\to f}(X_v)$ using $\mathcal{O}(D)$ floating point values. For K-Projected BP, we would transfer only updates to $\hat{\mu}^t_{v\to f}(X_v)$, which would involve transmitting K update values corresponding to the indices $\mathcal{U}^t_{v\to f}$, as well as transmission of the K indices in $\mathcal{U}^t_{v\to f}$. Of course, this is $\mathcal{O}(K)$ communication complexity, which will typically be much less than the communication complexity of Sum-Product.

3.4. Theoretical Convergence Properties

We will now turn to showing a number of theoretical results pertaining to our Projected BP algorithms. In particular, as was done in both (Noorshams & Wainwright, 2013) and (Elidan et al., 2006), we will examine the theoretical convergence properties of our algorithms in relation to the convergence properties of Sum-Product BP. In this analysis, as was done for the original Stochastic BP (Noorshams & Wainwright, 2013) and for Residual BP (Elidan et al., 2006), we make certain assumptions about the convergence of Sum-Product and the application instance, such as contractivity of the Sum-Product updates and positivity of the function kernels, in order to derive the properties of our algorithms.

3.4.1. Correspondence of Fixed Points

Our first theoretical results concern the fixed points of our Projected BP algorithms. Essentially, these results say that there is an exact correspondence between the fixed points of Sum-Product and the fixed points of Projected BP.

For Sum-Product, let $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}$ be the concatenation of all messages $\mu^t_{v\to f}(X_v)$ and let $\mathcal{M}_{\mathcal{F}\to\mathcal{V}}$ be the concatenation of all messages $\mu^t_{f\to v}(X_v)$. Then an iteration of Sum-Product BP may be written as alternating updates $\mathcal{M}_{\mathcal{F}\to\mathcal{V}} = F(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})$ and $\mathcal{M}_{\mathcal{V}\to\mathcal{F}} = G(\mathcal{M}_{\mathcal{F}\to\mathcal{V}})$. Projected BP, on the other hand, can be written as cyclical updates $\mathcal{M}_{\mathcal{V}\to\mathcal{F}} = U(\tilde{\mathcal{M}}_{\mathcal{V}\to\mathcal{F}}, \mathcal{M}_{\mathcal{V}\to\mathcal{F}})$, $\mathcal{M}_{\mathcal{F}\to\mathcal{V}} = F(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})$ and $\tilde{\mathcal{M}}_{\mathcal{V}\to\mathcal{F}} = G(\mathcal{M}_{\mathcal{F}\to\mathcal{V}})$. In this case, $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}$ consists of the message estimates $\hat{\mu}^t_{v\to f}(X_v)$. Note that the function node update function $F(\cdot)$ and the variable node update function $G(\cdot)$ are the same between Sum-Product and Projected BP, such that the only difference between the algorithms is the message estimate update function $U(\cdot)$, which is used only for Projected BP. Recall that, due to the subset selection methods considered, the message estimate update function $U(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}, \hat{\mathcal{M}}_{\mathcal{V}\to\mathcal{F}})$ is equal to $\hat{\mathcal{M}}_{\mathcal{V}\to\mathcal{F}}$ except where $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}$ is the most different from $\hat{\mathcal{M}}_{\mathcal{V}\to\mathcal{F}}$. Hence, we have that $U(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}, \hat{\mathcal{M}}_{\mathcal{V}\to\mathcal{F}}) = \hat{\mathcal{M}}_{\mathcal{V}\to\mathcal{F}}$ if and only if $\mathcal{M}_{\mathcal{V}\to\mathcal{F}} = \hat{\mathcal{M}}_{\mathcal{V}\to\mathcal{F}}$.

Now, to discuss the fixed points of the algorithms, we must specify what the state is. Let the Sum-Product iteration be defined as $\mathcal{M}^t_{\mathcal{V}\to\mathcal{F}} = G(F(\mathcal{M}^{t-1}_{\mathcal{V}\to\mathcal{F}}))$, where the state of the algorithm at time t is taken to be the messages $\mathcal{M}^t_{\mathcal{V}\to\mathcal{F}}$.

Let the Projected BP iteration be defined as $\mathcal{M}^t_{\mathcal{V}\to\mathcal{F}} = U(G(F(\mathcal{M}^{t-1}_{\mathcal{V}\to\mathcal{F}})), \mathcal{M}^{t-1}_{\mathcal{V}\to\mathcal{F}})$. Again, the state of the algorithm at time t is taken to be $\mathcal{M}^t_{\mathcal{V}\to\mathcal{F}}$.

Due to space limitations, we omit the proof. However, it is possible to prove that, for a (fixed) point $\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}$, we have that $\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}} = G(F(\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}))$ if and only if $\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}} = U(G(F(\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}})), \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}})$. Thus, there is an exact correspondence of fixed points between the two algorithms.

3.4.2. Conditions for Guaranteed Convergence to Fixed Points

We will now establish that under standard assumptions, as in (Noorshams & Wainwright, 2013) and (Elidan et al., 2006), β-Projected BP will converge to a fixed point, as long as Sum-Product is guaranteed to converge and the β parameter is suitably chosen. Our methods are similar to (Elidan et al., 2006) in that we first establish some basic properties of iterative algorithms and then proceed to apply these results to Projected BP. Again, due to space limitations, we will typically only give the main ideas for proofs.

We begin with the following definitions. Let $f(x)$ be the update for some iterative algorithm, i.e., $x[t+1] = f(x[t])$. Let $g()$ be another iterative algorithm, such that $g(x) = f(x) + e(x)$. I.e., $g()$ is the original algorithm plus a perturbation $e(x)$ in each step. Let $\mathcal{B}_N(r,x) = \{x' : \|x - x'\|_N \leq r\}$.

Assumption 3.1 (Exponential Convergence)**.** Under norm N, there is a ball $\mathcal{B}_N(r, x^*)$ with $r > 0$ around a point x^* where, for $x \in \mathcal{B}_N(r, x^*)$ and some $\alpha \in [0, 1)$, we have that

$$\|f(x) - x^*\|_N \leq \alpha \|x - x^*\|_N.$$

Assumption 3.1 implies that iterations of algorithm $f()$ converge exponentially quickly toward the fixed point x^* within $\mathcal{B}_N(r, x^*)$. Furthermore, note that this is a weaker assumption than contractivity, which is the assumption used in (Elidan et al., 2006) and (Noorshams & Wainwright, 2013), and results that hold under Assumption 3.1 will therefore also hold under a contractivity assumption.

We begin with a general result on iterative algorithms.

Lemma 3.2. *Suppose the following are true:*

- *Iterative algorithm $f()$ satisfies Assumption 3.1.*
- *Iterative algorithm $g()$ is defined as $g(x) = f(x) + e(x)$.*
- *For $x \in \mathcal{B}_N(r, x^*)$ and $\beta \in \left[0, \left(\frac{1-\alpha}{1+\alpha}\right)\right)$ we have that*

$$\|e(x)\|_N \leq \beta \|f(x) - x\|_N,$$

Then, the iterative algorithm $g()$, defined as $g(x) = f(x) + e(x)$, converges toward x^* exponentially quickly within $\mathcal{B}_N(r, x^*)$, such that

$$\|g(x) - x^*\|_N \leq (\beta(1+\alpha) + \alpha)\, \|x - x^*\|_N.$$

The proof method primarily involves use of the triangle inequality.

We would now like to apply this result to our β-Projected BP algorithm to better understand the convergence of the algorithm. Recall that the global state of Sum-Product and β-Projected BP, $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}$, is a concatenation of the individual variable to function message estimates $\hat{\mu}_{v\to f}$, which each have their own norm $N_{v\to f}$. We will first define some notation. Let $[\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_i$ indicate the i^{th} message $\hat{\mu}_{v\to f}$ from the state $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}$, where the edges (v, f) have been uniquely enumerated. The norm for that particular message is N_i. Furthermore, Let $[\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_{i,j}$ indicate the j^{th} element of the i^{th} message in $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}$. The global norm on $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}$ is simply N.

To apply Lemma 3.2 to derive a relationship between convergence of Sum-Product and β-Projected BP, we make the following assumption relating the global norm to the message norms:

Assumption 3.3. If we have that

$$\|[\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}]_i\|_{N_i} \leq \beta \|[\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}]_i\|_{N_i}$$

for every edge i, then we also have that

$$\|\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}\|_N \leq \beta \|\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}\|_N.$$

We now state our first theorem relating the convergence of β-Projected BP to that of Sum-Product BP.

Theorem 3.4. *Suppose the following are true:*

- *The Sum-Product iteration $f(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) \triangleq G(F(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}))$ satisfies Assumption 3.1 (with $x^* = \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}$).*
- *the message and global norms on $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}$ satisfy Assumption 3.3.*
- *We have that $\beta \in \left[0, \left(\frac{1-\alpha}{1+\alpha}\right)\right)$ for the β-Projected BP iteration*

$$g(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) \triangleq U(G(F(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})), \mathcal{M}_{\mathcal{V}\to\mathcal{F}}).$$

Then β-Projected BP converges toward $\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}$ exponentially quickly in the ball $\mathcal{B}_N(r, \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}})$, such that

$$\begin{aligned}&\|g(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) - \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}\|_N \\ &\qquad \leq (\beta(1+\alpha) + \alpha)\, \|\mathcal{M}_{\mathcal{V}\to\mathcal{F}} - \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}\|_N.\end{aligned}$$

Proving this result essentially involves verifying that Lemma 3.2 applies to this situation.

We now briefly consider what global norms satisfy Assumption 3.3. Proving these claims is straightforward, and therefore omitted.

Claim 3.5. Let the individual message norms be the usual Euclidean norm, i.e., $\|\,[\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_i\,\|_{N_i} = \sqrt{\sum_j ([\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_{i,j})^2}$. The Euclidean norm for the global message, $\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_2$, defined as $\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_2 = \sqrt{\sum_{(i,j)} ([\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_{i,j})^2}$, which was considered in (Noorshams & Wainwright, 2013), satisfies Assumption 3.3.

Claim 3.6. Let N_i be any norm defined for the message $[\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_i$. The composite max norm $\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_C$, defined as $\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_C = \max_i \|[\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_i\|_{N_i}$, which was considered in (Elidan et al., 2006), satisfies Assumption 3.3.

Claim 3.7. Let each N_i be the max norm, i.e., $\|\,[\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_i\,\|_{N_i} = \max_j \left|[\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_{i,j}\right|$. The global max norm $\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_\infty$, defined as $\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_\infty = \max_{(i,j)} \left|[\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_{i,j}\right|$, satisfies Assumption 3.3.

Theorem 3.4 applies quite generally under a broad class of norms satisfying Assumption 3.3. However, we will now present improved convergence rate bounds by assuming the Euclidean norm, as in (Noorshams & Wainwright, 2013), and by considering the particular structure of the perturbation $e(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})$.

Lemma 3.8. *Suppose the following are true:*

- *The norm N is the (global) Euclidian norm, as defined in Claim 3.5*
- *Iterative algorithm $f()$ satisfies Assumption 3.1 with respect to the particular norm N.*
- *We have that the iterative algorithm $g(x) = f(x) + e(x)$.*
- *The perturbation $e(x)$ takes the form $e(x) = b(x) \odot (x - f(x))$, where $b(x)$ is a vector of zeros and ones, and $\odot$ signifies the element-wise product.*
- *For $x \in \mathcal{B}_2(r, x^*)$ and $\beta \in \left[0, \frac{1-\alpha^2}{1+\alpha^2}\right)$ we have that*

$$\|e(x)\|_2 \leq \beta \|f(x) - x\|_2,$$

Then the iterative algorithm $g()$ converges toward x^* exponentially quickly within $\mathcal{B}_2(r, x^*)$, such that

$$\|g(x) - x^*\|_2 \leq \left(\alpha\sqrt{1-\beta^2} + \beta\right) \|x - x^*\|_2.$$

We note that, for α and β in the allowed ranges, it can easily be shown that $0 \leq \alpha\sqrt{1-\beta^2} + \beta \leq \beta(1+\alpha) + \alpha$, demonstrating that Lemma 3.8 achieves a better bound on convergence rate than Lemma 3.2 by specifically accounting for

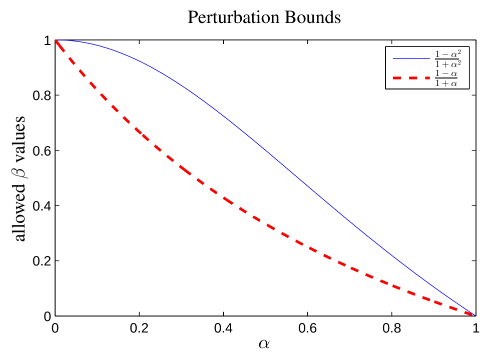


Figure 1. Different upper bounds on the relative perturbation size, β, for which convergence is guaranteed, as a function of α. Note that the bound specialized for the Euclidean norm includes a wider range of β values that guarantee convergence.

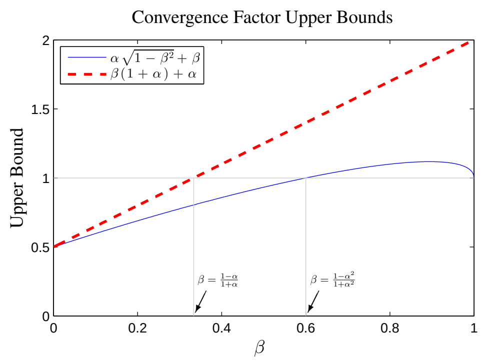


Figure 2. Different upper bounds on the convergence factor, where $\alpha = \frac{1}{2}$. A factor less than 1 guarantees exponential convergence. Smaller factors indicate faster convergence. Values of β where the bounds transition to above 1 are labeled.

the form of the global norm and the type of perturbation. Furthermore, note that $\frac{1-\alpha^2}{1+\alpha^2} \geq \frac{1-\alpha}{1+\alpha}$, and therefore Lemma 3.8 guarantees exponential convergence of β-Projected BP for a wider range of β values than Lemma 3.2.

We will now apply Lemma 3.8 to β-Projected BP.

Theorem 3.9. *Suppose the following are true:*

- *The global and message norms, N and N_i respectively, are the Euclidian norms, as in Claim 3.5*
- *The Sum-Product iteration $f(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) \triangleq G(F(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}))$ satisfies Assumption 3.1 (with $x^* = \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}$) with respect to the specified norm.*
- *We have that $\beta \in \left[0, \left(\frac{1-\alpha^2}{1+\alpha^2}\right)\right)$ for the β-Projected BP iteration*

$$g(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) \triangleq U(G(F(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})), \mathcal{M}_{\mathcal{V}\to\mathcal{F}}).$$

Then β-Projected BP converges toward $\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}$ exponentially quickly in the ball $\mathcal{B}_N(r, \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}})$, such that

$$\begin{aligned}\|g(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) - \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}\|_N \\ \leq \left(\alpha\sqrt{1-\beta^2}+\beta\right) \|\mathcal{M}_{\mathcal{V}\to\mathcal{F}} - \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}\|_N.\end{aligned}$$

Similar to Theorem 3.4, proof of this result essentially involves verifying that Lemma 3.8 applies to this situation.

We now consider the global max norm. In order to indicate composition of a function with itself, let $g^0(x) = x$ and $g^M(x) = g(g^{M-1}(x))$ for $M > 0$.

Lemma 3.10. *Suppose the following are true:*

- *Norm N is the (global) max norm of Claim 3.7*
- *Iterative algorithm $f()$ satisfies Assumption 3.1 with respect to the particular norm N.*
- *We have that $g(x) = f(x) + e(x)$.*
- *The perturbation $e(x)$ takes the form $e(x) = b(x) \odot (x - f(x))$, where $b(x)$ is a vector of zeros and ones, and $\odot$ signifies the element-wise product.*
- *In every sequence of M consecutive iterations of $g(x)$, we have that $[b(x)]_i$, the i^{th} element of $b(x)$, is 1 at least once. Alternatively, we can say that every element of the state has been updated at least once.*

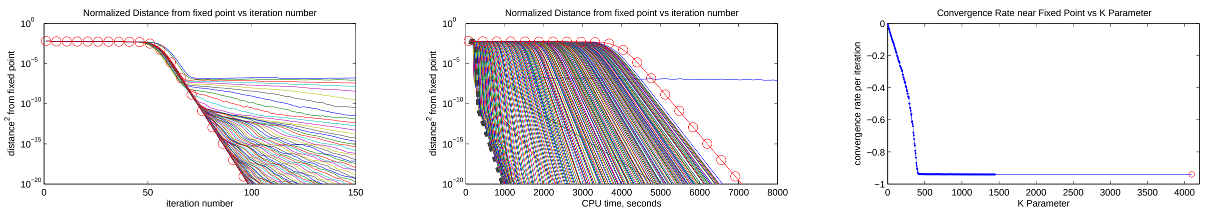
Then the iterative algorithm $g()$ converges toward x^* exponentially quickly within $\mathcal{B}_{r,N}(x^*)$, such that

$$\left\|g^M(x) - x^*\right\|_\infty \leq \alpha \left\|x - x^*\right\|_\infty.$$

Note that this lemma is closely related to Theorem 3.3 and Corollary 3.4 from (Elidan et al., 2006). The general idea of our proof is the following. Suppose $\left\|x^0 - x^*\right\|_\infty = r_0$. Once $[x^t]_i$ is updated by g, we show that $|[x^t]_i - [x^*]_i| \leq \alpha r_0$ for every iteration after. Once this is established, we need only to make sure that every element of the state gets updated, which is guaranteed by assumption within M iterations.

At this point, we note that Lemma 3.10 could be applied to a suitably modified β-Projected BP or K-Projected BP, where the algorithm is modified to ensure that every state element gets updated within some specified bound in number of iterations. Furthermore, if the Projected BP algorithm is reduced to a round-robin updating of the elements of each variable to function message estimate, then we have that Lemma 3.10 applies for $M = D$, i.e., the cardinality of the variables. Since the computational complexity of D iterations of such an algorithm is comparable to a single iteration of Sum-Product, we see that if Sum-Product satisfies Assumption 3.1 with respect to the max norm, then Lemma 3.10 implies that we stand to gain in convergence rate by breaking up the updates into fine-grained element-by-element updates, analogous to the arguments made in support of Residual BP in (Elidan et al., 2006).

We will now consider the convergence of our algorithms specifically for finite tree factor graphs. We will call this a factor tree. Of course, as is well known, the Sum-Product



(a) Squared Error vs. iterations. (b) Squared Error vs. compute seconds. (c) Convergence Rate vs. K.

Figure 3. Results in square grid Potts model. Sum-Product vs. K-Projected BP for various K. Red circles mark Sum-Product results.

algorithm converges to a unique fixed point in a finite number of iterations for factor trees, i.e., has no cycles (Pearl, 1988). We will now state a result saying that K-Projected BP also converges in a finite number of steps to a unique fixed point for a factor tree. Furthermore, as discussed earlier, this fixed point must also be a fixed point of Sum-Product, and hence must be the unique fixed point of Sum-Product that gives the exact solution to the marginal probabilities of the variables. For the following theorem, we assume, without loss of generality, that all leaf nodes are function nodes. This is possible because we can append to any variable node v a function node with the kernel $\psi_f(X_v) = 1$ with no consequential alteration to the overall graphical model.

Theorem 3.11. *In a factor tree of diameter d and variable nodes of cardinality D, the K-Projected BP update,*

$$\mathcal{M}^{t+1}_{\mathcal{V}\to\mathcal{F}} = U(G(F(\mathcal{M}^{t}_{\mathcal{V}\to\mathcal{F}})), \mathcal{M}^{t}_{\mathcal{V}\to\mathcal{F}}),$$

will converge to the unique fixed point $\mathcal{M}^{*}_{\mathcal{V}\to\mathcal{F}}$ in at most $\frac{d}{2}\lceil\frac{D}{K}\rceil$ iterations.

Proving this result involves induction over the tree, and is similar to the proof for Sum-Product.

4. Simulations

We now compare Sum-Product to K-Projected BP. We have implemented the algorithms in C++. Our experiments are with the simple square grid Potts model Markov random field (MRF) from (Noorshams & Wainwright, 2013). In this graph, we have pairwise potential functions, where $\psi_f(i,j) = 1$ if $i = j$, else $\psi_f(i,j) = \gamma$, for each edge in the MRF grid. We also have single variable potentials connected to each MRF node, such that $\psi_f(1) = 1$ and $\psi_f(i) = \mu + \sigma Z_{v,i}$ for $i \neq 1$, where $Z_{v,i}$ is an IID uniform random number from $(-1, 1)$. In our experiments using the square grid Potts MRF model, we have a 10×10 grid, $|\mathcal{X}_v| = D = 4096$, $\gamma = 0.1$, and $\mu = \sigma = 0.13$. Note that all of the function kernels are strictly positive with probability 1. We note that message update simplifications are possible due to the structure of the pairwise potentials, but no such simplifications are being made in the algorithms.

Figures 3(a), 3(b), and 3(c) show some results of these experiments. Figure 3(a) shows how the K-Projections BP algorithm converges to a fixed point in comparison to Sum-Product, as a function of iteration number, wherease Figure 3(b) shows this convergence versus seconds of compute time. What see is that K-Projected BP has slower convergence per iteration than Sum-Product, but faster convergence per compute second for most values of K. The plots include the convergence plots for $K = \{1, 6, 11, 16, 21, ...\}$. As we see in Figure 3(b), $K \approx 65$ gives the best tradeoff between convergence per iteration and computational complexity of iterations.

In Figure 3(c), we plot the per iteration convergence rate, versus K. To be specific, let $|\varepsilon_t|^2$ be the normalized squared distance of the belief propagation state from the BP fixed point at iteration t. When we write rate R, we mean that $|\varepsilon_t|^2$ is decreasing such that $|\varepsilon_{t+1}|^2 \approx e^R|\varepsilon_t|^2$. Interestingly, K-Projected BP achieves the per-iteration convergence rate of Sum-Product for $K > 410$ in this scenario.

5. Conclusion

Motivated in part by the strict low power requirements of distributed sensor networks, we proposed Projected BP, an alternative to Sum-Product belief propagation, developed with special consideration for increased computational and communications efficiency. Our algorithm gains in efficiency in both, while avoiding the slow convergence rate drawback of Stochastic BP. We provide theoretical results that Projected BP converges exponentially quickly under certain conditions in general graphs, and that it converges it a factor tree to the unique Sum-Product fixed point in a finite number of iterations. Finally, we present results on a test graph showing the potential of Projected BP in application.

We point out that our methods are actually complementary to methods for message update scheduling. Specifically, combining the methods of Projected BP with an update schedule like Residual BP (Elidan et al., 2006) could lead to incredible increases in the performance of belief propagation in large loopy graphs with high variable cardinality.

References

Çetin, Mujdat, Chen, Lei, Fisher III, John W., Ihler, Alexander T., Moses, Randolph L., Wainwright, Martin J., and Willsky, Alan S. Distributed fusion in sensor networks. *IEEE Signal Processing Magazine*, 23(4):42–55, July 2006.

Choi, Jungwook, Kim, Eric P., Rutenbar, Rob A., and Shanbhag, Naresh R. Error resilient MRF message passing architecture for stereo matching. In *IEEE Workshop on Signal Processing Systems*, 2013.

Elidan, G., McGraw, I., and Koller, D. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, 2006.

Goldberger, Jacob and Kfir, Haggai. Serial schedules for belief-propagation: Analysis of convergence time. *IEEE Transactions on Information Theory*, 54(3):1316–1319, March 2008.

Ihler, Alexander T., Fisher III, John W., and Willsky, Alan S. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6:905–936, May 2005.

Kfir, Haggai and Kanter, I. Parallel versus sequential updating for belief propagation decoding. *Physica A*, 330: 259–270, November 2003.

Kschischang, F. R., Frey, B. J., and Loeliger, H.-A. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, February 2001.

Noorshams, Nima and Wainwright, Martin. Stochastic belief propagation: A low-complexity alternative to the sum-product algorithm. *IEEE Transactions on Information Theory*, 59(4):1981–2000, April 2013.

Pearl, Judea. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, San Francisco, CA, 1988.

Sutton, Charles and McCallum, Andrew. Improved dynamic schedules for belief propagation. In *Proceedings of the 23nd Conference on Uncertainty in Artificial Intelligence*, 2007.

Tabatabaei Yazdi, S. M. Sadegh, Cho, Hyungmin, and Dolecek, Lara. Gallager B decoder on noisy hardware. *IEEE Transactions on Communications*, 61(5): 1660–1673, May 2013.

Varshney, Lav. Performance of LDPC codes under faulty iterative decoding. *IEEE Transactions on Information Theory*, 57(7):4427–4444, July 2011.