Sum-Product Alternatives for Severely Resource Constrained Applications - New Algorithms and Experimental Results

Andrew J. Bean
University of Illinois at Urbana-Champaign
Urbana, IL 61801 USA
ajbean@illinois.edu

Andrew C. Singer
University of Illinois at Urbana-Champaign
Urbana, IL 61801 USA
acsinger@illinois.edu

Abstract

We present a number of alternatives to Sum-Product belief propagation (BP) and experimental results for these. These BP alternatives are designed for applications that have severe constraints on both computation as well as communication between nodes. Such may be the case, for example, in distributed sensor networks. In particular, we propose a simplified and generalized Stochastic BP, as well as Zoom BP for networks with quantized communication between factor graph nodes. We also examine the performance of Sum-Product BP, our Stochastic BP and Zoom BP algorithms proposed here, as well as our previously proposed Projected BP algorithm. Experiments with a stereo image matching application show that Stochastic BP (original and our new version) converges quickly only if nearby information is reliable for decision making, and Projected BP and Zoom BP provide promising performance enhancements over Sum-Product.

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).

Another class of graph algorithms is gossip and consensus algorithms (Dimakis et al., 2010), where the typical application involves computing the average of observations taken at the nodes in a graph. In contrast to the belief propagation research, where much of the focus is on convergence properties and methods of improving convergence rather than further reductions in computation and communication overhead or improvements to robustness, the primary focus of a significant amount of consensus research has been exactly these issues. This is because sensor networks are a central motivation for consensus algorithms, where power constraints are typical and errors may be expected. For example, methods of average consensus with quantized messages are studied in (Kashyap et al., 2007; Lavaei & Murray, 2009a,b; Benezit et al., 2009; Carli et al.,

2010b,a; Aysal et al., 2008, 2007; Carli et al., 2011b; Kar & Moura, 2010). Consensus in networks with unreliable links has been studied in (Saligrama & Castanon, 2006; Carli et al., 2011b; Kar & Moura, 2010; Patterson & Bamieh, 2008; Kar & Moura, 2009; Carli et al., 2011a). Unfortunately, only limited work has been done in connecting consensus research with probabilistic graphical models, belief propagation, and factor graphs. One paper that does make this connection describes an algorithm reminiscent of belief propagation, which is named Consensus Propagation (Moallemi & Roy, 2006).

The contributions of this paper are the following: We first present a new version of Stochastic BP, which is a generalization and simplification of the algorithm given by (Noorshams & Wainwright, 2013), that avoids the potentially large precomputation required by the original Stochastic BP while allowing for higher order function nodes. We also present Zoom BP, which is designed for belief propagation in distributed networks with quantized communication between nodes. Zoom BP is based on our Projected BP algorithm, given in (Bean & Singer, 2015). We present the results of experiments with these algorithms (Sum-Product BP, Stochastic BP, Zoom BP, and Projected BP), and explore the reasons behind the computational benefits of our methods. Hence, this work is also an extension to (Bean & Singer, 2015), which primarily provided theoretical results on Projected BP. Finally, we give concluding remarks in Section 7, and discuss some potential topics for future investigation.

2 Factor Graphs and Marginalization

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 $e \in \mathcal{E} \subset \mathcal{V} \times \mathcal{F}$ 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

$$\Psi(\mathcal{V}) = \Psi(X) = \prod_{f\in\mathcal{F}} \psi_f(\mathcal{N}_f), \tag{1}$$

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

$$\mathrm{P}_{X_v}(i) = \sum_{x:x_v=i} \mathrm{P}_X(X = x). \tag{2}$$

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

3 The Sum-Product Algorithm

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

$$\begin{aligned}&\text{Marginal}(f \to v)\\ &\quad= \sum_{X_u : u\in\mathcal{N}_f\setminus v} \psi_f(\mathcal{N}_f) \prod_{w\in\mathcal{N}_f\setminus v} \mu^{t-1}_{w\to f}(X_w),\end{aligned} \tag{3}$$

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 (3) 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.

Data: $\mathcal{V}, \mathcal{F}, \mathcal{E}, \psi_f(\cdot)$ for each $f \in \mathcal{F}$
Result: $\hat{\mathrm{P}}_{X_v}(i)$ for each $v \in \mathcal{V}$

Initialize $\mu^0_{v\to f}(X_v) = \frac{1}{D}$ for each $(v,f) \in \mathcal{E}$;
Initialize $t = 0$;
repeat
  $t \leftarrow t+1$;
  /* Update fn to var messages */
  foreach $(v,f) \in \mathcal{E}$ **do**
    if $|\mathcal{N}_f| = 1$ **then**
      $\theta^t_{f\to v}(X_v) = \psi_f(X_v)$;
    else
      $\theta^t_{f\to v}(X_v) = \text{Marginal}(f \to v)$;
    end
  end
  /* Update var to fn messages */
  foreach $(v,f) \in \mathcal{E}$ **do**
    if $|\mathcal{N}_v| = 1$ **then**
      $\mu^t_{v\to f}(X_v) = \frac{1}{D}$;
    else
      $\mu^t_{v\to f}(X_v) = \frac{1}{Z}\prod_{g\in\mathcal{N}_v\setminus f}\theta^t_{g\to v}(X_v)$;
    end
  end
until some stopping condition;
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 $f \in \mathcal{N}_v$

Algorithm 2: SBP0 fn-to-var message updates.

Data: $\mu^t_{v\to f}(X_v)$ for each $(v,f) \in \mathcal{E}$
Result: $\theta^t_{f\to v}(X_v)$ for each $(v,f) \in \mathcal{E}$

foreach $(v,f) \in \mathcal{E}$ **do**
  if $|\mathcal{N}_f| = 1$ **then**
    $\theta^t_{f\to v}(X_v) \leftarrow \psi_f(X_v)$;
  else
    pick w as the only element of $\mathcal{N}_f \setminus v$;
    Generate $J^t_{f\to v} \in \mathcal{X}_w$:
      $J^t_{f\to v} \sim \mathrm{P}^t_{f\to v}(X_w)$
        $\propto \mu^t_{w\to f}(X_w)\beta_{f\to v}(X_w)$;
    $\theta^t_{f\to v}(X_v) \leftarrow$
    $(1-\lambda^t)\theta^{t-1}_{f\to v}(X_v) + \lambda^t\Gamma_{f\to v}(X_v, J^t_{f\to v})$;
  end
end
return $\theta^t_{f\to v}(X_v)$ for each $(v,f) \in \mathcal{E}$

4 Stochastic Belief Propagation

The first alternative to the Sum-Product algorithm that we will explore is called Stochastic Belief Propagation. The original, which we call SBP0, was proposed by (Noorshams & Wainwright, 2013) with the intended goal of providing an alternative to Sum-Product with greatly reduced computational complexity per iteration, as well as reduced communication requirements, in order to tailor belief propagation to settings like distributed sensor networks, where there may be strict computational and communications restrictions. In this section, we present SBP2, which both generalizes their method to arbitrary graphs and simplifies it in order to overcome a potentially significant drawback of the original algorithm—the precompute step.

4.1 Original Stochastic Belief Propagation

The original Stochastic BP, which we will refer to as SBP0, is a randomized algorithm that can approximate the single variable marginals as given by Equation (2). SBP0 maintains the restrictions given above, such as strict positivity of the function kernels and finite variables, but we additionally enforce that the function nodes have maximum degree of two. Each iteration of the algorithm consists of a randomized low complexity update of the function to variable messages, conditional upon the current variable to function messages, such that the expectation of the update is equivalent to a damped version of Sum-Product. Note that the variable to function messages, being samples from the sets $\mathcal{X}_v$, are reminiscent of the types of messages exchanged in the Social Sampling distributed consensus algorithm (Sarwate & Javidi, 2011).

In particular, first define the following precomputed values $\beta_{f\to v}()$ and $\Gamma_{f\to v}()$. These are defined for each factor node of degree 2, where we have that the function kernel $\psi_f(\mathcal{N}_f) = \psi_f(X_v, X_w)$ for $\mathcal{N}_f = \{v, w\}$. Specifically, we have that $\beta_{f\to v}(X_w) = \sum_{i\in\mathcal{X}_v}\psi_f(i, X_w)$ and $\Gamma_{f\to v}(X_v, X_w) = \frac{\psi_f(X_v,X_w)}{\beta_{f\to v}(X_w)}$ for every function to variable edge. Once these have been computed, they are maintained for use in all iterations of the algorithm. The algorithm begins with function to variable messages $\theta^{t-1}_{f\to v}(X_v)$. These are used to update the variable to function messages $\mu^t_{v\to f}(X_v)$ exactly as with Sum-Product BP. The difference is in how these are subsequently used to update the messages $\theta^t_{f\to v}(X_v)$. Rather than computing an update like Equation (3), the update is chosen randomly such that, in expectation, $\theta^0_{f\to v}(X_v)$ moves in the direction of the Sum-Product update. Note that there is a decaying step size parameter λ^t. In our experiments, we use $\lambda^t = \frac{2}{t+1}$. The specifics of the updates of each $\theta^0_{f\to v}(X_v)$ are given in Algorithm 2.

In (Noorshams & Wainwright, 2013), the authors give a number of theoretical results, but the main results state that if the Sum-Product update rule $\mathbf{m}^t = F(\mathbf{m}^{t-1})$ is contractive in the Euclidean norm, where $\mathbf{m}^t$ is the concatenation of all function to variable messages $\theta^t_{f\to v}(X_v)$ throughout the graph, such that $\|F(\mathbf{m}) - F(\mathbf{m}')\|_2 \le \alpha\|\mathbf{m} - \mathbf{m}'\|_2$ for some $\alpha \in [0,1)$, then the expected deviation of the Stochastic BP state from the unique Sum-Product fixed

Algorithm 3: Simplified Generalized Stochastic BP.

Data: $\mathcal{V}, \mathcal{F}, \mathcal{E}, \psi_f(\cdot)$ for each $f \in \mathcal{F}$
Result: $\hat{\mathrm{P}}_{X_v}(i)$ for each $v \in \mathcal{V}$

Initialize $t = 0$;
foreach $(v, f) \in \mathcal{E}$ **do**
  if $|\mathcal{N}_f| = 1$ **then**
    Initialize $\theta^0_{f\to v}(X_v) = \psi_f(X_v)$;
  else
    Initialize $\theta^0_{f\to v}(X_v) = \frac{1}{D}$;
  end
end
repeat
  $t \leftarrow t + 1$;
  /* Update var to fn messages */
  foreach $(v, f) \in \mathcal{E}$ **do**
    if $|\mathcal{N}_v| = 1$ **then**
      $\mu^t_{v\to f}(X_v) = \frac{1}{D}$;
    else
      $\mu^t_{v\to f}(X_v) = \frac{1}{Z}\prod_{g\in\mathcal{N}_v\setminus f}\theta^{t-1}_{g\to v}(X_v)$;
    end
  end
  /* Update fn to var messages */
  foreach $(v, f) \in \mathcal{E}$ **do**
    if $|\mathcal{N}_f| = 1$ **then**
      $\theta^t_{f\to v}(X_v) = \psi_f(X_v)$;
    else
      /* Sample each component of $J^t_{f\to v}$ independently. */
      Generate $J^t_{f\to v} \in \prod_{w\in\mathcal{N}_f\setminus v}\mathcal{X}_w$:
      $J^t_{f\to v} \sim \mathrm{P}^t_{f\to v}(\mathcal{N}_f \setminus v) \propto \prod_{w\in\mathcal{N}_f\setminus v}\mu^t_{w\to f}(X_w)$;
      $\theta^t_{f\to v}(X_v) = (1-\lambda^t)\theta^{t-1}_{f\to v}(X_v) + \lambda^t\psi_f(X_v, J^t_{f\to v})$;
    end
  end
until some stopping condition;
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 $f \in \mathcal{N}_v$

point, i.e., $\mathbb{E}[\|\mathbf{m}^t - \mathbf{m}^*\|_2]$, for the Stochastic BP update rule $\mathbf{m}^t = \hat{F}(\mathbf{m}^{t-1})$ in the same graph decreases like $\frac{1}{\sqrt{t}}$. This holds true for both tree graphs, as well as graphs with cycles that satisfy the stated contraction property.

4.2 Simplified Generalized Stochastic BP (SBP2)

SBP0 has two significant drawbacks. First is the limitation to degree-2 function nodes. The other is the potentially significant precomputation of each $\beta_{f\to v}()$ and $\Gamma_{f\to v}()$. These computations potentially require $\mathcal{O}(D^2)$ computations and $\mathcal{O}(D^2)$ storage for *every edge* in the graph connected to a function node of degree greater than 1. For these reasons, we have developed a simplified and generalized Stochastic BP, which we will refer to as SBP2. This algorithm proceeds as in Algorithm 3 This algorithm completely avoids the computation and storage requirements of SBP0 for $\beta_{f\to v}()$ and $\Gamma_{f\to v}()$. Furthermore, note that this algorithm still amounts to a damped version of Sum-Product, in expectation. The main difference is that we sample from the distributions $\mu^t_{w\to f}(X_w)$ directly, instead of from a distribution with the factor $\beta_{f\to v}(X_w)$. Also, the update adds scaled values from $\psi_f()$ rather than $\Gamma_{f\to v}()$.

5 Quantized Coded Belief Propagation: Zoom BP

We will now present another variation of BP that utilizes a discrete communication channel. Essentially, this algorithm, which we call Zoom BP, is an adaptation of Projected BP (Bean & Singer, 2015) that incorporates ideas from (Carli et al., 2010a) for the coding of real valued messages into a finite alphabet.

The algorithm is similar to Projected BP, with the exception that real values are not transmitted from variable nodes to function nodes in the updating of the message estimates $\hat{\mu}_{v\to f}$. Instead, we transmit quantized representations of a subset of the differences $[\mu_{v\to f} - \hat{\mu}_{v\to f}]_i$, where $[\cdot]_i$ indicates i^{th} vector component. To do so, we employ separate encoder/decoder pairs as described in (Carli et al., 2010a) for each element $[\mu_{v\to f} - \hat{\mu}_{v\to f}]_i$. For simplicity, let us look at the coded transmission for the updates to a particular element $\hat{\mu} \triangleq [\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_{i,j}$ from the element $\mu \triangleq \left[\tilde{\mathcal{M}}_{\mathcal{V}\to\mathcal{F}}\right]_{i,j}$. First, define a scale factor a whose value will be known at both the sending and receiving sides. This will be the state of our encoder/decoder pair. We also define globally known initial values for a and $\hat{\mu}$, in order that their values may be tracked at both the enocoding and decoding sides with only knowledge of the transmitted symbols. Next, we define an encoder function $Q : \mathbb{R}\times\mathbb{R} \to \mathcal{A}$ that takes the scale factor a and the difference $\Delta\mu \triangleq \mu - \hat{\mu}$ and produces a symbol q from the finite alphabet $\mathcal{A} = \{i : i \in \mathbb{Z}, |i| \le \bar{Q}\}$ for some integer $\bar{Q} \ge 1$. We also define a decoder function $H : \mathbb{R}\times\mathcal{A} \to \mathbb{R}$ that takes the scale factor a and the transmitted symbol q and constructs a message estimate update. Finally, we define an encoder/decoder state update function $V : \mathbb{R}\times\mathcal{A} \to \mathbb{R}$ that takes the current encoder/decoder state—the scale factor a—and the transmitted symbol q and determines the updated encoder/decoder state.

Specifically, the encoder function is

$$Q(a, \Delta\mu) \triangleq \begin{cases} \min\left(\left\lfloor \frac{\Delta\mu}{a} \right\rfloor, \bar{Q}\right) & \text{if } \Delta\mu \ge 0 \\ \max\left(\left\lceil \frac{\Delta\mu}{a} \right\rceil, -\bar{Q}\right) & \text{if } \Delta\mu \le 0 \end{cases}.$$

The decoder function is simply $H(a, q) \triangleq aq$. Finally, the

Algorithm 4: Zoom BP, quantized coded messages.

Data: $\mathcal{V}, \mathcal{F}, \mathcal{E}, \psi_f(\cdot)\ \forall f \in \mathcal{F}$

Result: $\hat{\mathrm{P}}_{X_v}(i)\ \forall v \in \mathcal{V}$

1 Initialize $\hat{\mu}^0_{v\to f}(X_v) = \frac{1}{D}\ \forall (v,f) \in \mathcal{E}$;
2 Initialize $\mathcal{U}^0_{v\to f} = \mathcal{X}_v$;
3 Initialize $\theta^0_{f\to v}(X_v) = \hat{\mu}^{-1}_{v\to f}(X_v) = 0\ \forall (v,f) \in \mathcal{E}$;
4 Initialize $a^0_{v\to f}(X_v) = a_0$;
5 Initialize $t = 0$;
6 **repeat**
7 $\quad t \leftarrow t+1$;
$\quad$ /* Update fn to var messages */
8 $\quad$ **foreach** $(v,f) \in \mathcal{E}$ **do**
9 $\quad\quad$ **if** $|\mathcal{N}_f| = 1$ **then**
10 $\quad\quad\quad \theta^t_{f\to v}(X_v) = \psi_f(X_v)$;
11 $\quad\quad$ **else**
12 $\quad\quad\quad$ Enforce $\theta^t_{f\to v}(X_v) = \text{Marginal}(f \to v)$;
13 $\quad\quad$ **end**
14 $\quad$ **end**
$\quad$ /* Update var to fn messages */
15 $\quad$ **foreach** $(v,f) \in \mathcal{E}$ **do**
16 $\quad\quad$ **if** $|\mathcal{N}_v| = 1$ **then**
17 $\quad\quad\quad \mu^t_{v\to f}(X_v) = \frac{1}{D}$;
18 $\quad\quad$ **else**
19 $\quad\quad\quad \mu^t_{v\to f}(X_v) = \frac{1}{Z}\prod_{g\in\mathcal{N}_v\setminus f} \theta^t_{g\to v}(X_v)$;
20 $\quad\quad$ **end**
21 $\quad$ **end**
22 $\quad$ **foreach** $(v,f) \in \mathcal{E}$ **do**
23 $\quad\quad$ Choose $\mathcal{U}^t_{v\to f}$ where $\mathcal{U}^t_{v\to f} \subseteq \mathcal{X}_v$;
24 $\quad\quad$ Transmit $\mathcal{U}^t_{v\to f}$;
25 $\quad\quad$ Transmit $q^t_{v\to f}(i) = Q(a^{t-1}_{v\to f}(i), \mu^t_{v\to f}(i) - \hat{\mu}^{t-1}_{v\to f}(i))$ for $i \in \mathcal{U}^t_{v\to f}$;
26 $\quad$ **end**
27 $\quad$ **foreach** $(v,f) \in \mathcal{E}$ **do**
28 $\quad\quad$ $\hat{\mu}^t_{v\to f}(i) =$
$$\begin{cases} \hat{\mu}^{t-1}_{v\to f}(i) \\ \quad +H(a^{t-1}_{v\to f}(i), q^t_{v\to f}(i)) & \text{if } i \in \mathcal{U}^t_{v\to f} \\ \hat{\mu}^{t-1}_{v\to f}(i) & \text{if } i \notin \mathcal{U}^t_{v\to f} \end{cases};$$
29 $\quad\quad$ $a^t_{v\to f}(i) =$
$$\begin{cases} V(a^{t-1}_{v\to f}(i), q^t_{v\to f}(i)) & \text{if } i \in \mathcal{U}^t_{v\to f} \\ a^{t-1}_{v\to f}(i) & \text{if } i \notin \mathcal{U}^t_{v\to f} \end{cases};$$
30 $\quad$ **end**
31 **until** some stopping condition;
32 **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 $f \in \mathcal{N}_v$

encoder/decoder state update function is

$$V(a,q) \triangleq \begin{cases} Z_{\text{in}}a & \text{if } |q| < \bar{Q} \\ Z_{\text{out}}a & \text{if } |q| = \bar{Q} \end{cases},$$

where $0 < Z_{\text{in}} < 1$ and $Z_{\text{out}} > 1$. In the course of the algorithm, the sending node will compute a value μ, which it wishes to transmit to a neighboring node. Rather than sending μ, it may (if this element is in the update set) then compute the symbol $q = Q(a, \mu - \hat{\mu})$ and send it to the neighboring node. At this point, both the sending and receiving nodes compute the update $\hat{\mu} \leftarrow \hat{\mu} + H(a,q)$, and follow this with the encoder/decoder state update $a \leftarrow V(a,q)$. Note that the encoder and decoder ensure that $\hat{\mu} + H(a,q)$ is not outside the range between μ and $\hat{\mu}$. This ensures that all message estimate values $\hat{\mu}$ remain in the range $[0,1]$, since we have such a condition on the message values μ.

The details of the algorithm are given in Algorithm 4. The subset selection in Line 23 could take a number of forms, but here we consider choosing the K elements where $\mu^t_{v\to f}$ and $\hat{\mu}^{t-1}_{v\to f}$ differ the most. The marginal computation at Line 12 is identical to the same computation in Projected BP. This is less than the full computation of Marginal$(f \to v)$, since typically only a few (or even just 1) elements of $\hat{\mu}^{t-1}_{v\to f}$ are different from $\hat{\mu}^{t-2}_{v\to f}$. Hence, we "enforce" equality with a low complexity update, rather than performing the computation from scratch. The details of this update are given in (Bean & Singer, 2015). In fact, the full computational complexity of Zoom BP is essentially the same as that of Projected BP. The only computational difference is in the encoder and decoder operations. On the other hand, with Zoom BP, we are able to send much smaller messages than would be involved with Sum-Product BP or even Projected BP, since we do not need to send entire floating point values. Instead, for each edge (v,f), we may send the size $\left|\mathcal{U}^t_{v\to f}\right|$ of the update subset using $\lceil \log_2(D) \rceil$ bits, followed by the elements of $\mathcal{U}^t_{v\to f}$ using $\left|\mathcal{U}^t_{v\to f}\right| \lceil \log_2(D) \rceil$ bits, followed by the symbols $q^t_{v\to f}(i)$ for $i \in \mathcal{U}^t_{v\to f}$, using $\left|\mathcal{U}^t_{v\to f}\right| \lceil \log_2(2\bar{Q}+1) \rceil$ bits. This represents potentially significant savings in communication rate compared to the transmission of 32- or 64-bit floating point values.

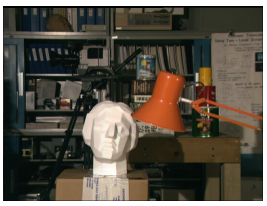
6 Simulations

6.1 Stereo Image Matching - Energy Minimization

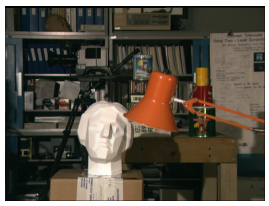
In this section, we examine BP performance in a computer vision application. Specifically, we will be comparing Sum-Product BP, the Stochastic BP algorithms SBP0 and SBP2, K-Projected BP (Bean & Singer, 2015), and Zoom BP, all of which we have implemented in C++. These experiments use the "Cones" stereo image pair from (Scharstein & Szeliski, 2003), as shown in Figures 1(a) and 1(b), and the "Tsukuba" stereo image pair from (Scharstein & Szeliski, 2002), as shown in Figures 1(c) and 1(d), in order to test the algorithms on a basic model for stereo image matching. Note that the goal here is not to improve on stereo matching performance, but rather to draw a comparison between the algorithms. Also note that the model we use in these tests admit computational simplifications due

(a) "Cones" left image.

(b) "Cones" right image.



(c) "Tsukuba" left image.



(d) "Tsukuba" right image.

Figure 1: "Cones" and "Tsukuba" source images.

to the special form of the potential functions, but none of this structure is being exploited, allowing us to focus our attention only on the virtues and drawbacks of each of the algorithms. This model consists of the following: First, let variables $X_{i,j}$ represent disparities of the right image from the left image. Specifically, if $X_{i,j} = d$, then the content of pixel (i, j) of the left image, indexed from the bottom left of the image, lines up with content of pixel $(i - d, j)$. Note that higher d indicates closer to camera. We also have pairwise potential functions composing a square grid Pott's model, where we have that

$$\psi_{v_1,v_2}(i,j) = \begin{cases} 1 & \text{if } i = j \\ \gamma & \text{if } i \neq j \end{cases},$$

for $v_1 = X_{x,y}$ and v_2 is one of the four neighboring variables $X_{x+1,y}$, $X_{x-1,y}$, $X_{x,y+1}$, or $X_{x,y-1}$. In our experiments, we use $\gamma = \frac{1}{20}$. Finally, we have single variable data potentials, whose values depend on the source images. These are defined as follows:

$$\psi_{i,j}(d) = \max\left(\varepsilon, \exp\left(-\lambda \left\|C_L(i,j) - C_R(i-d,j)\right\|_1\right)\right),$$

where we have that $C_L(i, j)$ and $C_R(i, j)$ are vectors of red, green, and blue color components of pixel (i, j), taking values 0 to 255, for the left and right source images, respectively, and $\|\cdot\|_1$ is the L^1-norm. In our experiments, we use $\varepsilon = \frac{1}{1000}$ and $\lambda = \frac{1}{10}$. Note that if the colors of pixels (i, j) and $(i - d, j)$ are similar, then $\psi_{i,j}(d)$ will have a value closer to 1, whereas if the colors are less similar, then $\psi_{i,j}(d)$ will be smaller, with a minimal value of ε.

We will also define an "energy," which is a function of hard decisions $x_{i,j}$ on the value of each variable $X_{i,j}$. This energy is simply defined as $E(\mathbf{x}) = -\log(\Psi(\mathbf{x}))$, where $\Psi(\cdot)$ is the global function represented by the factor graph, as

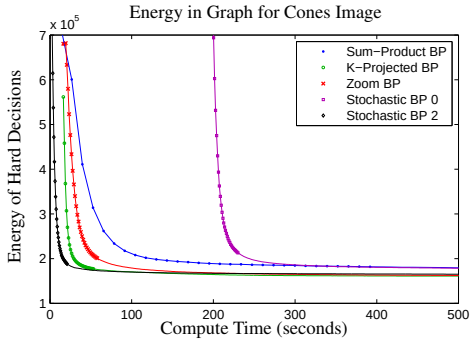


Figure 2: Cones data. Compute time includes setup time and precomputations. First 30 iterations have markers.

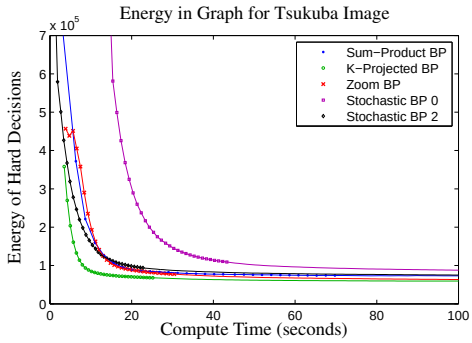


Figure 3: Tsukuba data. Compute time includes setup time and precomputations. First 30 iterations have markers.

defined in Equation (1), and $\mathbf{x}$ is the vector of all hard decisions $x_{i,j}$.

We will first evaluate Sum-Product BP and the mentioned alternatives with respect to this energy function $E(\mathbf{x})$. However, this requires some justification. Note that when the global function $\Psi(\cdot)$ is considered proportional to a probability distribution $\mathrm{P}(\cdot)$ modeling the joint distribution among the variables at the variable nodes, minimizing $E(\mathbf{x})$ with respect to $\mathbf{x}$ is equivalent to finding a maximum a posteriori probability (MAP) estimate of $\mathbf{x}$, i.e., $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} E(\mathbf{x})$. This is often approximated by the use of Max-Product belief propagation (on the factor graph of $\Psi(\mathbf{x})$) or, equivalently, Min-Sum belief propagation (on a graphical representation of $E(\mathbf{x})$). Sum-Product belief propagation, on the other hand, does not move toward minimizing the energy function $E(\mathbf{x})$, but instead minimizes a function on message space called the Bethe free energy (Yedidia et al., 2001, 2005). Ultimately, however, we use Sum-Product or one of its alternatives in order to approximate the marginal probabilities $\mathrm{P}_{X_v}(x_v)$ as in Equation (2). From this, the decision rule is to choose $\hat{x}_v = \arg\max_{x_v} \mathrm{P}_{X_v}(x_v)$. We will use the notation $v^{\sim}$ to indicate "all other variables in the graph other than v."

In order to justify evaluating Sum-Product and its relatives

according to $E(\mathbf{x})$, suppose, for some variable node v, we have that

$$\mathrm{P}_{X_v}(x_v) = \begin{cases} 1 & \text{if } x_v = \tilde{x}_v \\ 0 & \text{if } x_v \neq \tilde{x}_v \end{cases} . \quad (4)$$

Clearly, the decision rule applied to this would give

$$\hat{x}_v = \arg\max_{x_v} \mathrm{P}_{X_v}(x_v) = \tilde{x}_v.$$

On the other hand, we have that $\max_{\mathbf{x}} \mathrm{P}_X(\mathbf{x}) > 0$, since $\sum_{\mathbf{x}} \mathrm{P}_X(\mathbf{x}) = 1$. However, we have that

$$\begin{aligned} \mathrm{P}_{X_v}(x_v) &= \sum_{\bar{\mathbf{x}}:\bar{x}_v = x_v} \mathrm{P}_X(\bar{\mathbf{x}}) \\ &= \sum_{\bar{\mathbf{x}}:\bar{x}_v = x_v} \mathrm{P}_{X_v|X_{v\sim}}(\bar{x}_v|\bar{\mathbf{x}}_{v\sim})\mathrm{P}_{X_{v\sim}}(\bar{\mathbf{x}}_{v\sim}), \end{aligned}$$

which implies that $\mathrm{P}_{X_{v\sim}}(\mathbf{x}_{v\sim}) = 0$ whenever $\mathrm{P}_{X_v|X_{v\sim}}(\cdot|\mathbf{x}_{v\sim}) \neq \mathrm{P}_{X_v}(\cdot)$. Now, consider some $\check{\mathbf{x}}$ such that $\check{x}_v \neq \tilde{x}_v$. Then we have that

$$\mathrm{P}_X(\check{\mathbf{x}}) = \mathrm{P}_{X_v|X_{v\sim}}(\check{x}_v|\check{\mathbf{x}}_{v\sim})\mathrm{P}_{X_{v\sim}}(\check{\mathbf{x}}_{v\sim}).$$

If $\mathrm{P}_{X_v|X_{v\sim}}(\cdot|\check{\mathbf{x}}_{v\sim}) = \mathrm{P}_{X_v}(\cdot)$, then we have that

$$\begin{aligned} \mathrm{P}_X(\check{\mathbf{x}}) &= \mathrm{P}_{X_v|X_{v\sim}}(\check{x}_v|\check{\mathbf{x}}_{v\sim})\mathrm{P}_{X_{v\sim}}(\check{\mathbf{x}}_{v\sim}) \\ &= \mathrm{P}_{X_v}(\check{x}_v)\mathrm{P}_{X_{v\sim}}(\check{\mathbf{x}}_{v\sim}) = 0, \end{aligned}$$

since $\check{x}_v \neq \tilde{x}_v$. But, if $\mathrm{P}_{X_v|X_{v\sim}}(\cdot|\check{\mathbf{x}}_{v\sim}) \neq \mathrm{P}_{X_v}(\cdot)$, then we have that

$$\begin{aligned} \mathrm{P}_X(\check{\mathbf{x}}) &= \mathrm{P}_{X_v|X_{v\sim}}(\check{x}_v|\check{\mathbf{x}}_{v\sim})\mathrm{P}_{X_{v\sim}}(\check{\mathbf{x}}_{v\sim}) \\ &= \mathrm{P}_{X_v|X_{v\sim}}(\check{x}_v|\check{\mathbf{x}}_{v\sim}) \times 0 = 0. \end{aligned}$$

Since we know that $\max_{\mathbf{x}} \mathrm{P}_X(\mathbf{x}) > 0$, we must have that $\check{\mathbf{x}} \neq \arg\max_{\mathbf{x}} \mathrm{P}_X(\mathbf{x})$, which implies that $\hat{x}_v = \tilde{x}_v$ when $\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \mathrm{P}_X(\mathbf{x})$.

Thus, if $\mathrm{P}_{X_v}(\cdot)$ is of the form given in Equation (4), then the decision rule from an algorithm of the Sum-Product family is attempting to find the same x_v as the MAP rule, which minimizes the energy function $E(\mathbf{x})$. Thus, if we believe that $\mathrm{P}_{X_v}(\cdot)$ is close to the form given in Equation (4) for most of the variables in the graphical model, which seems to be the case in many applications including our stereo matching example, then it seems that $E(\hat{\mathbf{x}})$ is a reasonable way to evaluate the quality of the estimate $\hat{\mathbf{x}}$.

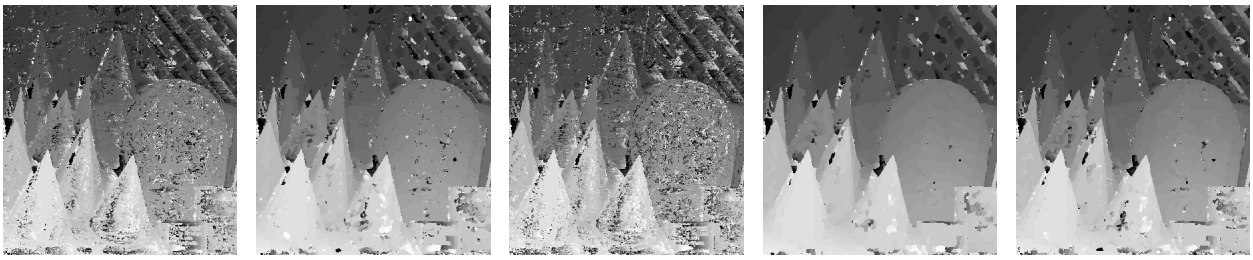
We will be comparing Sum-Product, K-Projected BP ($K = 1$), Zoom BP ($K = 1$), SBP0, and SBP2. For Zoom BP, we are using $\bar{Q} = 7$, $Z_{\text{in}} = 0.763$, $Z_{\text{out}} = 225$, and $a_0 = \frac{1}{1000}$. Furthermore, we apply the encoding and decoding procedure to both the variable to function messages and the function to variable messages. For the Cones data set, the disparities to be estimated have $D = 50$ possible values, whereas $D = 17$ for the Tsukuba data set. The square grid factor graph generated from the Cones images has dimensions 351×375, and is 367×288 for Tsukaba.

Figure 2 shows results on the Cones images. Sum-Product takes the largest steps in decreasing the energy, but the iterations each take more compute time than any other algorithm. Thus, in the first phase of convergence, after the initialization period, the other algorithms are decreasing the energy faster than Sum-Product, albeit with more iterations along the way. Taking into account the initialization time, we see that all of K-Projected BP, Zoom BP, and SBP2 reach lower levels of energy than Sum-Product, and come close to this level in a fraction of the time that it takes Sum-Product. SBP0, on the other hand, takes a significant amount of time to precompute every $\beta_{f\to v}(X_w)$ and $\Gamma_{f\to v}(X_v, X_w)$. SBP2 needs no such computation, yet suffers no comparative loss in convergence rate. SBP0 ends up converging to an energy level comparable to Sum-Product, which is a higher than achieved by the other algorithms. Even Zoom BP outperforms Sum-Product BP in energy minimization, despite sending such a small amount of information in every message update.

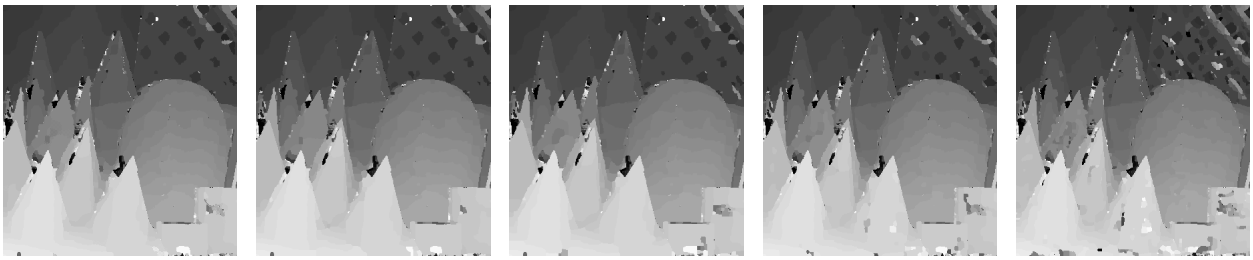
Similar results for the Tsukaba images are in Figure 3. Since D is smaller in this case, the per-iteration advantage of each algorithm over Sum-Product is less significant. K-Projected BP converges faster and to a lower energy level than any other algorithm. Zoom BP is comparable to Sum-Product in convergence speed (per second), though it approaches a slightly lower value than Sum-Product. With SBP0, the precompute time is less significant than with Cones, due to the reduced cardinality of the variables and the smaller size of the source images. Interestingly, both SBP0 and SBP2 suffer in convergence speed, with the energy vs. time decreasing less steeply than the other algorithms. We will revisit this observation when we examine the qualitative results seen in the disparity maps. In brief, this is a result of the Stochastic BP algorithms having difficult propagating beliefs over long distances in the graph. In summary, the energy plots for Cones and Tsukaba seem to indicate that K-Projected BP has the most reliable advantage over Sum-Product, Zoom BP can be comparable or better than Sum-Product at greatly reduced data transfer requirements (relevant to distributed networks), and the precompute phase of SBP0 provides no benefit over SBP2 while increasing the setup time severely. Finally, the per-second convergence rate, after initialization, of SBP (0 or 2) has the potential to be better than any of the other algorithms, but this is subject to the particular graph, as will see in the following.

6.2 Stereo Image Matching - Qualitative Results

We now examine the disparity maps generated by applying the decision rule $\hat{x}_v = \arg\max_{x_v} \hat{\mathrm{P}}_{X_v}(x_v)$ for each variable node. In Figures 4(a)-4(d), we have the disparity maps that result after 26.6 seconds of compute time on the Cones data for Sum-Product, K-Projected BP, Zoom BP, and SBP2. This is equivalent to the time taken to perform 2



(a) Sum-Product. It=2. (b) K-Projected BP. It=9. (c) Zoom BP. It=8. (d) SBP2. It=38. (e) SBP0. It=38.



(f) Sum-Product. (g) K-Projected BP. (h) Zoom BP. (i) SBP2. (j) SBP0.

Figure 4: (a)-(d): 26.6 seconds compute time (2 Sum-Product iterations). (e): SBP0 at 238.5 seconds (38 iterations). (f)-(j): 1000 iterations.

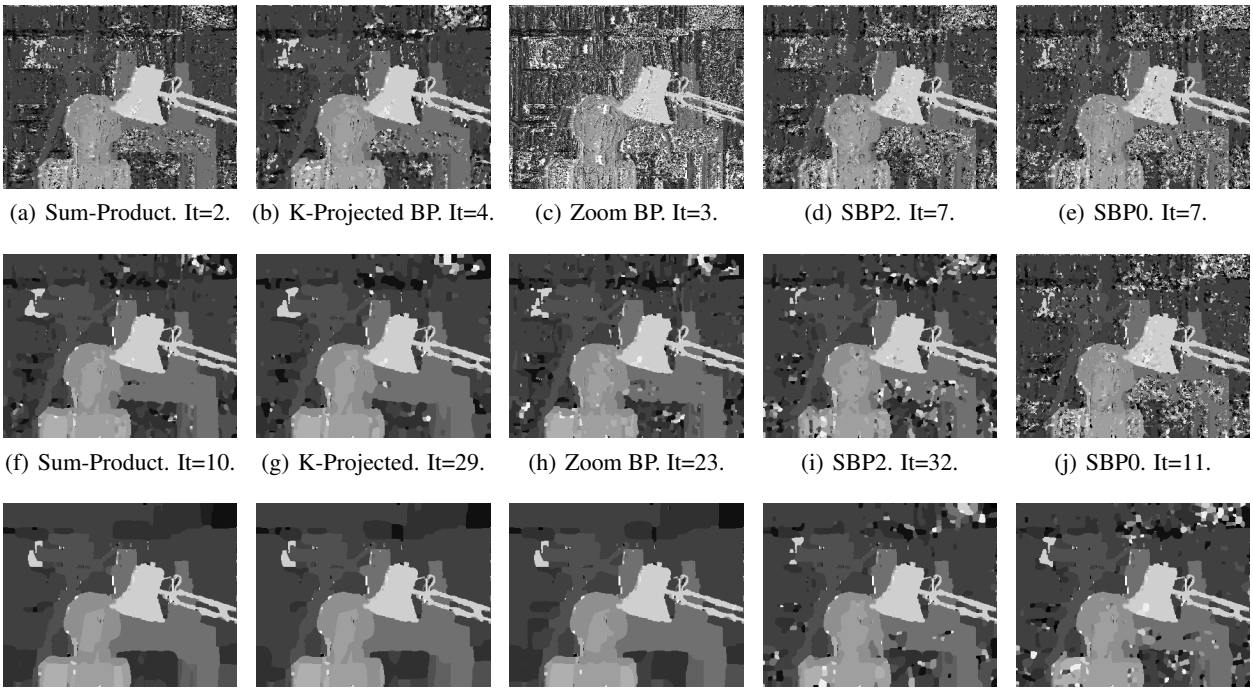
iterations of Sum-Product. In order of increasing apparent quality of results, we have Zoom BP, Sum-Product (slightly better), K-Projected BP (much better), and finally SBP2. This is roughly in line with the performance indicated by Figure 2. SBP0 had not completed its precomputation by this time, but we show the disparity map at iteration 38 in Figure 4(e), which is the number of iterations shown for SBP2 in Figure 4(d). The results of SBP0 are comparable or slightly worse than the results of SBP2.

Figures 4(f)-4(j) shows the disparity maps that result after 1000 iterations for every algorithm. All but SBP0 have results that are essentially the same, whereas SBP0 appears to give inferior results, particularly for the fence at the back, and the mug in the front.

These Cones images results show that the Sum-Product alternatives can give good results in a practical application sooner than Sum-Product. Results for Tsukuba are similar, but we will gain more intuition about the strengths or weaknesses of the algorithms.

In Figures 5(a)-5(d), we have the disparity maps that result after 6.3 seconds of compute time for Sum-Product, K-Projected BP, Zoom BP, and SBP2, the amount of time for 2 iterations of Sum-Product. We make some of the same conclusions as with the Cones data: Even with smaller variable cardinality $D = 17$, the results of K-Projected BP show that it is possible to improve on the efficiency of Sum-Product. We again note no advantage in SBP0 over SBP2, only the disadvantage of the large precompute time, which is evident because the results of iteration 7 of SBP0 and SBP2 are comparable, but the results from SBP0 are achieved after significantly more computation time.

Figures 5(f)-5(j), where we have the disparity maps that result after 24.6 seconds of compute time (10 iterations of Stochastic BP), is where we begin to see one of the drawbacks of Stochastic BP (SBP0 and SBP2). Specifically, focusing on the parts of the disparity maps corrsponding with the table, as well as the top right dard background (among other areas) in Figures 5(i) and 5(j), we see that Stochastic BP is having difficulty deciding how to assign disparities in these regions, producing small blobs of random disparities rather than one uniform disparity estimate throughout. This is the difficulty hinted at by the energy plots decreasing more slowly in Figure 3. Looking back at the Tsukuba source images, we see that these regions correspond with broad homogeneous regions that lack significant texture. Essentially, due to the low information content of randomly sampled messages, Stochastic BP has difficulty propagating information over significant distances in the graph. This effect is particularly evident in Figures 5(k)-5(o), where all of the algorithms have run for 1000 iterations. Even after so many iterations, SBP0 and SBP2 are both unable to smooth out the homogeneous regions to a single disparity estimate, whereas Sum-Product, K-Projected BP, and Zoom BP have all been able to accomplish this smoothing with far fewer iterations. This effect is much less apparent with the Cones images, as there is sufficient texture throughout the entirety of the scene, allowing the algorithms to settle on a choice of disparity based only on the immediately surrounding image data.



(a) Sum-Product. It=2. (b) K-Projected BP. It=4. (c) Zoom BP. It=3. (d) SBP2. It=7. (e) SBP0. It=7.

(f) Sum-Product. It=10. (g) K-Projected. It=29. (h) Zoom BP. It=23. (i) SBP2. It=32. (j) SBP0. It=11.

(k) Sum-Product. (l) K-Projected BP. (m) Zoom BP. (n) SBP2. (o) SBP0.

Figure 5: (a)-(d): 6.3 seconds compute time (2 Sum-Product iterations). (e): SBP0 at 20.3 seconds (7 iterations). (f)-(j): 24.6 seconds compute time (10 Sum-Product iterations). (k)-(o): 1000 iterations.

7 Conclusion

Motivated in part by the strict low power requirements and high cost of communication in distributed sensor networks, we proposed an improvement to Stochastic BP (Noorshams & Wainwright, 2013), as well as Zoom BP. Both algorithms utilize low bandwidth discrete channels. The new Stochastic BP (SBP2) avoids the large precompute of SBP0, while suffering no loss in performance versus SBP0. Experiments with stereo image matching indicate that SBP performs well only if information need not travel far in the factor graph. Projected BP and Zoom BP show promising advantages over Sum-Product, with algorithm choice depending on communication constraints. Projected BP could be justified in both distributed and centralized computing applications.

Acknowledgments

This work was supported in part by Systems on Nanoscale Information fabriCs (SONIC), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA.

References

Aysal, Tuncer C., Coates, Mark, and Rabbat, Michael. Distributed average consensus using probabilistic quantization. In *Proceedings of the 14th IEEE Workshop on Statistical Signal Processing*, pp. 640–644, August 2007.

Aysal, Tuncer Can, Coates, Mark J., and Rabbat, Michael G. Distributed average consensus with dithered quantization. *IEEE Transactions on Signal Processing*, 56 (10):4905–4918, October 2008.

Bean, Andrew J. and Singer, Andrew C. Projected belief propagation: A sum-product alternative for severely resource constrained applications. In *Submitted to the 31st International Conference on Machine Learning (See Supplement)*, 2015.

Benezit, Florence, Thiran, Patrick, and Vetterli, Martin. Interval consensus: From quantized gossip to voting. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 3661–3664, 2009.

Carli, Ruggero, Bullo, Francesco, and Zampieri, Sandro. Quantized average consensus via dynamic coding/decoding schemes. *International Journal of Robust and Nonlinear Control*, 20:156–175, 2010a.

Carli, Ruggero, Fagnani, Fabio, Frasca, Paolo, and Zampieri, Sandro. Gossip consensus algorithms via quantized communication. *Automatica*, 46:70–80, 2010b.

Carli, Ruggero, Como, Giacomo, Frasca, Paolo, and Garin, Federica. Distributed averaging on digital erasure networks. *Automatica*, 47:115–121, 2011a.

Carli, Ruggero, Como, Giacomo, Frasca, Paolo, and Garin, Federica. Distributed averaging on digital noisy networks. In *Proceedings of the Information Theory and Applications Workshop (ITA)*, pp. 1–9, February 2011b.

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.

Dimakis, A.G., Kar, S., Moura, J.M.F., Rabbat, M.G., and Scaglione, A. Gossip algorithms for distributed signal processing. *Proceedings of the IEEE*, 98(11):1847–1864, November 2010. ISSN 0018-9219. doi: 10.1109/JPROC.2010.2052531.

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.

Kar, Soummya and Moura, Jose M. F. Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise. *IEEE Transactions on Signal Processing*, 57(1):355–369, January 2009.

Kar, Soummya and Moura, Jose M. F. Distributed consensus algorithms in sensor networks: Quantized data and random link failures. *IEEE Transactions on Signal Processing*, 58(3):1383–1400, March 2010.

Kashyap, Akshay, Başar, Tamer, and Srikant, R. Quantized consensus. *Automatica*, 43(7):11921203, 2007.

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.

Lavaei, Javad and Murray, Richard M. On quantized consensus by means of gossip algorithm - part I: Convergence proof. In *Proceedings of the American Control Conference*, pp. 394401, June 2009a.

Lavaei, Javad and Murray, Richard M. On quantized consensus by means of gossip algorithm - part II: Convergence time. In *Proceedings of the American Control Conference*, pp. 29582965, June 2009b.

Moallemi, Ciamac C. and Roy, Benjamin Van. Consensus propagation. *IEEE Transactions on Information Theory*, 52 (11):4753–4766, November 2006.

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.

Patterson, Stacy and Bamieh, Bassam. Distributed consensus with link failures as a structured stochastic uncertainty problem. In *Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computation*, pp. 623–627, 2008.

Saligrama, Venkatesh and Castanon, David A. Reliable distributed estimation with intermittent communications. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 6763–6768, December 2006.

Sarwate, A. D. and Javidi, T. Opinion dynamics and distributed learning of distributions. In *Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computation*, September 2011.

Scharstein, D. and Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7–42, April-June 2002.

Scharstein, D. and Szeliski, R. High-accuracy stereo depth maps using structured light. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, volume 1, pp. 195–202, Madison, WI, June 2003.

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.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. Generalized belief propagation. In Leen, T. K., Dietterich, T. G., and Tresp, V. (eds.), *Advances in Neural Information Processing Systems*, volume 13, Cambridge, MA, 2001. MIT Press.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, July 2005.