CHAPTER 1

RESOURCE EFFICIENT BELIEF PROPAGATION ALGORITHMS

1.1 Introduction

Algorithms on graphs are important in many decision making, inference, and detection tasks. Belief Propagation (BP), one example being Sum-Product Belief Propagation, 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 [1]. 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 parallel-updates Sum-Product implementation of BP. One technique for increasing the efficiency of computation is Residual BP [2], 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 Belief Propagation Algorithm [3], which reduces both the computational cost of an iteration of the algorithm and the communication overhead at the expense of convergence rate. In [4], 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 [5,6], the error-resilient Markov random field message passing architecture for stereo matching [7], and analysis of belief propagation subject to certain types of messaging errors [8].

Another class of graph algorithms is gossip and consensus algorithms [9], where the typical application involves computing the average of observations taken at the nodes in a graph. In contrast to 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 a number of articles [10–19]. Consensus in networks with unreliable links has been studied by several researchers [18–23]. 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 [24].

In this chapter, we use some of the tools from the consensus research developed for lower computational and communication complexity in order to create belief propagation algorithms that are more appropriate to applications with strict resource constraints. In Section 1.2, we give background on the Sum-Product Belief Propagation algorithm and discuss potential areas for increased efficiency. In Section 1.3, we look at the Stochastic Belief Propagation algorithm [3], and proceed to both generalize and simplify the algorithm. In Section 1.4, we address some of the drawbacks of Stochastic BP, especially the slow convergence rate, by applying a technique that is reminiscent to Residual BP on a high level. We call the resulting algorithm Projected BP. We preceed to prove some properties of the fixed points and convergence of the algorithm. In Section 1.5, we consider more severely constrained communications between nodes in the graph, and propose a method of belief propagation with coarsely quantized messages. In Section 1.6, we present experimental results comparing our algorithms to other belief propagation algorithms, and explore the reasons behind the computational benefits

of our methods. Finally, we give concluding remarks in Section 1.7, and discuss some potential topics for future investigation.

1.2 Basics of the Sum-Product Algorithm

In this chapter, 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 the form of 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 the (strictly) positive real numbers. 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

$$\Psi(\mathcal{V}) = \Psi(X) = \prod_{f\in\mathcal{F}} \psi_f(\mathcal{N}_f), \tag{1.1}$$

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

Algorithm 1.1: 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 $v \in \mathcal{V}$

1 Initialize $\mu^0_{v\to f}(X_v) = \frac{1}{D}$ for each $(v, f) \in \mathcal{E}$;
2 Initialize $t = 0$;
3 **repeat**
4 $\quad t \leftarrow t + 1$;
$\quad$ /* Update function to variable messages */
5 $\quad$ **foreach** $(v, f) \in \mathcal{E}$ **do**
6 $\quad\quad$ **if** $|\mathcal{N}_f| = 1$ **then**
7 $\quad\quad\quad \theta^t_{f\to v}(X_v) = \psi_f(X_v)$;
8 $\quad\quad$ **else**
9 $\quad\quad\quad \theta^t_{f\to v}(X_v) = \mathrm{Marginal}(f \to v)$;
10 $\quad\quad$ **end**
11 $\quad$ **end**
$\quad$ /* Update variable to function messages */
12 $\quad$ **foreach** $(v, f) \in \mathcal{E}$ **do**
13 $\quad\quad$ **if** $|\mathcal{N}_v| = 1$ **then**
14 $\quad\quad\quad \mu^t_{v\to f}(X_v) = \frac{1}{D}$;
15 $\quad\quad$ **else**
16 $\quad\quad\quad \mu^t_{v\to f}(X_v) = \frac{1}{Z}\prod_{g\in\mathcal{N}_v\setminus f} \theta^t_{g\to v}(X_v)$;
17 $\quad\quad$ **end**
18 $\quad$ **end**
19 **until** some stopping condition;
20 **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 $f \in \mathcal{N}_v$

marginal distributions

$$\mathrm{P}_{X_v}(i) = \sum_{x:x_v=i} \mathrm{P}_X(X = x). \tag{1.2}$$

Approximating these marginal distributions is the objective of the Sum-Product belief propagation algorithm, and this is the problem we are concerned with throughout this chapter.

The Sum-Product algorithm iteratively updates messages on 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 proceeds as in Algorithm 1.1. The scaling factors $\frac{1}{Z}$ on Lines 16 and 20 are chosen so that the respective messages and marginal probability estimates sum to 1. Such

normalization may not be necessary in a factor tree, i.e., a factor graph that is a tree. We also have that

$$\text{Marginal}(f \to v) = \sum_{X_u : u \in \mathcal{N}_f \setminus v} \psi_f(\mathcal{N}_f) \prod_{w \in \mathcal{N}_f \setminus v} \mu_{w \to f}^{t-1}(X_w). \tag{1.3}$$

Note that the computational complexity of the function to variable message update in Line 9, via Equation (1.3), is $\mathcal{O}(D^{|\mathcal{N}_f|})$ as a function of D. In some applications, the function node kernels $\psi_f(\cdot)$ have structure that allow simplifications that lead to computational savings in this step. 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 belief propagation is for a flooding messaging schedule. Of course, other schedules for updating messages in the graph are possible, and this has been extensively studied [2, 25–27].

1.3 Stochastic Belief Propagation

The first alternative to the Sum-Product algorithm that we will explore is called Stochastic Belief Propagation. Proposed by Noorshams and Wainwright in [3], the intended goal of the work is to provide an alternative to Sum-Product that has 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 the original Stochastic BP as given in [3], and proceed to both generalize their method and simplify it in order to overcome some drawbacks of the original algorithm.

1.3.1 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 (1.2). We maintain 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 essentially consists of a randomized low complexity update of the function to variable messages, conditional upon the current variable to function messages, such that the expected value of the update is equal to a step in the same direction as a Sum-Product update. Equivalently, the expectation of the update is equivalent to a damped version of Sum-Product. Interesting to note, the nature of the variable to function messages, being samples from the sets $\mathcal{X}_v$, is reminiscent of the types of messages exchanged in the Social Sampling distributed consensus algorithm presented in [28].

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, which proceeds as in Algorithm 1.2. Again, we have that $\frac{1}{Z}$ is a normalization factor to ensure that the message or distribution sums to 1. Furthermore, note that there is a decaying step size parameter λ^t. In our work, we always use $\lambda^t = \frac{2}{t+1}$. The algorithm begins with function to variable messages $\theta^0_{f\to v}(X_v)$, which are initialized as in Lines 6 and 8. 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 message $\theta^0_{f\to v}(X_v)$. Rather than computing an update like Equation (1.3), the update is chosen randomly such that, in expectation, $\theta^0_{f\to v}(X_v)$ moves in the direction of the update indicated by Equation (1.3).

A number of theoretical results are given in the original Stochastic BP paper [3], 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

Algorithm 1.2: Original Stochastic Belief Propagation – SBP0.

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 $v \in \mathcal{V}$

Precompute $\beta_{f\to v}(X_w)$ for each $(v,f) \in \mathcal{E}$;
Precompute $\Gamma_{f\to v}(X_v, X_w)$ for each $(v,f) \in \mathcal{E}$;
Initialize $t = 0$;
foreach $(v,f) \in \mathcal{E}$ **do**
  if $|\mathcal{N}_f| = 1$ **then**
    Initialize $\theta^0_{f\to v}(X_v) = \psi_f(X_v)$; /* $|\mathcal{N}_f| = 1$ */
  else
    Initialize $\theta^0_{f\to v}(X_v) = \frac{1}{D}$; /* $|\mathcal{N}_f| = 2$ */
  end
end
repeat
  $t \leftarrow t + 1$;
  /* Update variable to function 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 function to variable 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
      pick w as the only element of $\mathcal{N}_f \setminus v$;
      Generate $J^t_{f\to v} \in \mathcal{X}_w$:
        $J^t_{f\to v} \sim \mathrm{P}^t_{f\to v}(X_w) \propto \mu^t_{w\to f}(X_w)\beta_{f\to v}(X_w)$;
      /* We use $\lambda^t = \frac{2}{t+1}$. Other choices are possible. */
      $\theta^t_{f\to v}(X_v) = (1-\lambda^t)\theta^{t-1}_{f\to v}(X_v) + \lambda^t \Gamma_{f\to v}(X_v, J^t_{f\to v})$;
    end
  end
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 $f \in \mathcal{N}_v$

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 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, at best, like $\frac{1}{\sqrt{t}}$. This holds true for both tree graphs, as well as graphs with cycles that satisfy the stated contraction property.

1.3.2 Generalization to Higher Degree Interactions (SBP1)

In the original Stochastic BP paper [3], no method is given for extending the algorithm to graphs that have function nodes of degree larger than 2. We will now show how the method can be extended to such graphs.

We begin by generalizing the definitions of the precomputed values $\beta_{f\to v}()$ and $\Gamma_{f\to v}()$, as follows:

$$\beta_{f\to v}(\mathcal{N}_f \setminus v) = \sum_{\mathcal{X}_v} \psi_f(\mathcal{N}_f),$$

and

$$\Gamma_{f\to v}(\mathcal{N}_f) = \frac{\psi_f(\mathcal{N}_f)}{\beta_{f\to v}(\mathcal{N}_f \setminus v)}.$$

In words, for each fixed $(X_{v_2}, ..., X_{v_n})$ with $\{v_2, ..., v_n\} = \mathcal{N}_f \setminus v$, the function $\Gamma_{f\to v}(X_v, X_{v_2}, ..., X_{v_n})$ takes the form of a conditional probability distribution $\mathrm{P}_{f\to v}(X_v | X_{v_2}, ..., X_{v_n})$ over the values $\mathcal{X}_v$, and is obtained by taking the values from $\psi_f(X_v, X_{v_2}, ..., X_{v_n})$ and applying a normalization factor $\beta_{f\to v}(X_{v_2}, ..., X_{v_n}) = \sum_{X_v} \psi_f(X_v, X_{v_2}, ..., X_{v_n})$. Note that we can think of $\beta_{f\to v}(X_{v_2}, ..., X_{v_n})$ as proportional to a joint probability distribution over the variables $X_{v_2}, ..., X_{v_n}$, which is derived from a joint distribution $\mathrm{P}_f(\mathcal{N}_f)$ that is proportional to the function kernel $\psi_f(\mathcal{N}_f)$.

Once $\beta_{f\to v}()$ and $\Gamma_{f\to v}()$ have been computed, the generalized Stochastic BP algorithm, which we will refer to as SBP1, then proceeds as in Algorithm 1.3. The value Z and the step sizes λ^t are as described for SBP0. Note that

$$\begin{aligned}\mathbb{E}_{J^t_{f\to v}}[\Gamma_{f\to v}(X_v, J^t_{f\to v})] &= \sum_{\mathcal{N}_f \setminus v} \Gamma_{f\to v}(\mathcal{N}_f) \mathrm{P}^t_{f\to v}(\mathcal{N}_f \setminus v) \\ &= \sum_{\mathcal{N}_f \setminus v} \psi_f(\mathcal{N}_f) \frac{1}{Z} \prod_{w \in \mathcal{N}_f \setminus v} \mu^t_{w\to f}(X_w),\end{aligned}$$

which shows that the function to variable message update is proportional, in expectation, to that of Sum-Product. Therefore, in expectation, this generalized Stochastic BP is also equivalent to a damped version of Sum-Product

Algorithm 1.3: Generalized Stochastic Belief Propagation – SBP1.

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 $v \in \mathcal{V}$

Precompute $\beta_{f\to v}()$ for each $(v,f) \in \mathcal{E}$;
Precompute $\Gamma_{f\to v}()$ for each $(v,f) \in \mathcal{E}$;
Initialize $t = 0$;
foreach $(v,f) \in \mathcal{E}$ **do**
  if $|\mathcal{N}_f| = 1$ **then**
    Initialize $\theta^0_{f\to v}(X_v) = \psi_f(X_v)$; /* $|\mathcal{N}_f| = 1$ */
  else
    Initialize $\theta^0_{f\to v}(X_v) = \frac{1}{D}$; /* $|\mathcal{N}_f| > 1$ */
  end
end
repeat
  $t \leftarrow t + 1$;
  /* Update variable to function 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 function to variable 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
      Generate $J^t_{f\to v} \in \prod_{w\in\mathcal{N}_f\setminus v} \mathcal{X}_w$:
        $J^t_{f\to v} \sim \mathrm{P}^t_{f\to v}(\mathcal{N}_f\setminus v) \propto \beta_{f\to v}(\mathcal{N}_f\setminus v)\prod_{w\in\mathcal{N}_f\setminus v}\mu^t_{w\to f}(X_w)$;
      /* We use $\lambda^t = \frac{2}{t+1}$. Other choices are possible. */
      $\theta^t_{f\to v}(X_v) = (1-\lambda^t)\theta^{t-1}_{f\to v}(X_v) + \lambda^t\Gamma_{f\to v}(X_v, J^t_{f\to v})$;
    end
  end
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 $f \in \mathcal{N}_v$

BP. Furthermore, note that if we restrict the graph to have function nodes of degree at most equal to 2, then SBP1 reduces to the SBP0 algorithm from [3].

As a final point, we comment on how to generate $J^t_{f\to v}$, which comes from a potentially high dimensional distribution. One way to deal with this, which is

the approach we take later in our computational experiments, is an instance of *rejection sampling*, where we first sample each variable $X_u : u \in \mathcal{N}_f \setminus v$ independently according to $\mu^t_{w\to f}(X_u)$ to get $\hat{J}^t_{f\to v}$. We then accept or reject $\hat{J}^t_{f\to v}$ according to $\beta_{f\to v}(\hat{J}^t_{f\to v})$ by drawing a value U uniformly from 0 to $\max_J \beta_{f\to v}(J)$. We let $J^t_{f\to v} = \hat{J}^t_{f\to v}$ if $U < \beta_{f\to v}(\hat{J}^t_{f\to v})$. Otherwise, try again with a newly generated $\hat{J}^t_{f\to v}$. A downside is that this reduces the decentralized nature of the algorithm, as it requires repeated coordination between the function and variable nodes to give as many random samples as necessary to get one accepted. Alternatively, if $\hat{J}^t_{f\to v}$ is rejected, we may simply skip the update for that iteration. Then, the only difference is that the step has some probability of having size zero, but the update still results in a damped Sum-Product in expectation.

1.3.3 Simplification with Reduced Setup Time (SBP2)

With SBP1, we have overcome the degree-2 function node limitation of SBP0, but there are still some significant drawbacks that we would like to address. For example, the precomputation of $\beta_{f\to v}()$ and $\Gamma_{f\to v}()$ requires $\mathcal{O}(D^{|\mathcal{N}_f|})$ computations and $\mathcal{O}(D^{|\mathcal{N}_f|})$ storage for *every edge* in the graph connected to a function node of degree greater than 1. In addition to this, the procedure to generate $J^t_{f\to v}$ could potentially involve a high probability of sample rejection, which leads to a loss of efficiency. For these reasons, we have developed a simplified Stochastic BP, which we will refer to as SBP2. This algorithm proceeds as in Algorithm 1.4

Importantly, this algorithm completely avoids the computation and storage requirements of SBP0 and SBP1 for $\beta_{f\to v}()$ and $\Gamma_{f\to v}()$. It also avoids the complications with generating samples of $J^t_{f\to v}$, because we simply need independent samples of each X_w for $w \in \mathcal{N}_f \setminus v$, thus avoiding complications with sampling from high dimensional joint distributions from which sampling may be difficult. Furthermore, note that

$$\begin{aligned}\mathbb{E}_{J^t_{f\to v}}[\psi_f(X_v, J^t_{f\to v})] &= \sum_{\mathcal{N}_f\setminus v} \psi_f(\mathcal{N}_f)\mathrm{P}^t_{f\to v}(\mathcal{N}_f \setminus v) \\ &= \sum_{\mathcal{N}_f\setminus v} \psi_f(\mathcal{N}_f)\frac{1}{Z}\prod_{w\in\mathcal{N}_f\setminus v} \mu^t_{w\to f}(X_w),\end{aligned}$$

Algorithm 1.4: Simplified Stochastic Belief Propagation – SBP2.

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 $\theta^0_{f\to v}(X_v) = \psi_f(X_v)$; /* $|\mathcal{N}_f| = 1$ */
  else
    Initialize $\theta^0_{f\to v}(X_v) = \frac{1}{D}$; /* $|\mathcal{N}_f| > 1$ */
  end
end
repeat
  $t \leftarrow t + 1$;
  /* Update variable to function 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 function to variable 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 $J^t_{f\to v} \in \prod_{w\in\mathcal{N}_f\setminus v} \mathcal{X}_w$:
        $J^t_{f\to v} \sim \mathrm{P}^t_{f\to v}(\mathcal{N}_f \setminus v) \propto \prod_{w\in\mathcal{N}_f\setminus v} \mu^t_{w\to f}(X_w)$;
      /* We use $\lambda^t = \frac{2}{t+1}$. Other choices are possible. */
      $\theta^t_{f\to v}(X_v) = (1-\lambda^t)\theta^{t-1}_{f\to v}(X_v) + \lambda^t \psi_f(X_v, J^t_{f\to v})$;
    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 $f \in \mathcal{N}_v$

1.3.4 Advantages and Shortcomings

As we have seen, these stochastic belief propagation algorithms have advantages over Sum-Product BP in resource usage efficiency per iteration. With respect to computations, we see that each iteration has only complexity of

$\mathcal{O}(D)$ (after setup of the algorithm). For communications overhead, we have only $\mathcal{O}(\log(D))$ overhead for each variable to function message, since we send only samples from each $\mathcal{X}_v$. Unfortunately, the convergence rate suffers, as it is reduced from an exponential rate of convergence, as seen in the contractivity assumption, to a rate of $\mathcal{O}(\frac{1}{\sqrt{t}})$. Due to this convergence rate disadvantage, in the following sections we will examine methods for reduced compexity belief propagation without giving up exponential convergence rates in the cases where Sum-Product converges exponentially quickly.

1.4 Projected Belief Propagation

The next set of 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 [2]. This is because Residual BP, as well as the algorithms to be presented in this section, 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 our Projected BP algorithms, 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.

Algorithm 1.5: Projected 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 $v \in \mathcal{V}$

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

1.4.1 Algorithm Description

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 1.5. In Line 11 of Algorithm 1.5, the expression Marginal($f \to v$) differs from Equation (1.3) only in that $\mu^{t-1}_{w\to f}(X_w)$ is re-

placed with $\hat{\mu}^{t-1}_{w\to f}(X_w)$. Specifically, we have that

$$\text{Marginal}(f \to v) = \sum_{X_u : u \in \mathcal{N}_f \setminus v} \psi_f(\mathcal{N}_f) \prod_{w \in \mathcal{N}_f \setminus v} \hat{\mu}^{t-1}_{w\to f}(X_w). \tag{1.4}$$

Again, Z is a normalizing constant to ensure that the respective results sum to 1.

There are two important differences between Projected BP and Sum-Product BP. First, we point out Lines 21-24 of Algorithm 1.5. 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 correspondingly 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 [3], with the most important difference being that our careful deterministic choice of information to send in the message is 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 11 of Algorithm 1.5. 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 available. Plus, instead of simply computing the update $\text{Marginal}(f \to v)$ 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 $\mathcal{U}^{t-1}_{w\to f}$. This is why we specify that we *enforce* the equality in Line 11.

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

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

Algorithm 1.6: Reduced Complexity Update of $\theta^t_{f\to v}(X_v)$, i.e., "Enforce $\theta^t_{f\to v}(X_v) = \text{Marginal}(f \to v)$."

Data: t,(v, f), $\mathcal{N}_f$, $\{\hat{\mu}^{t-1}_{w\to f}, \hat{\mu}^{t-2}_{w\to f}, \mathcal{U}^{t-1}_{w\to f}\}$ for each $w \in \mathcal{N}_f$
Result: $\theta^t_{f\to v}$

/* If not a flooding schedule, $\mathcal{N}_{\text{to\_update}}$ may be a smaller subset. */
Initialize $\mathcal{N}_{\text{to\_update}} = \mathcal{N}_f \setminus v$;
Initialize $\mathcal{N}_{\text{updated}} = (\mathcal{N}_f \setminus v) \setminus \mathcal{N}_{\text{to\_update}}$; /* {} if flooding. */
Initialize $\theta^t_{f\to v} = \theta^{t-1}_{f\to v}$;
foreach $w \in \mathcal{N}_{\text{to\_update}}$ **do**
  $t \leftarrow t + 1$;
  /* Incorporate changes in $\hat{\mu}^{t-1}_{w\to f}$ from $\hat{\mu}^{t-2}_{w\to f}$ */
  $\tilde{\mu}_{w\to f}(X_w) = \hat{\mu}^{t-1}_{w\to f}(X_w) - \hat{\mu}^{t-2}_{w\to f}(X_w)$; /* $= 0$ for $X_w \notin \mathcal{U}^{t-1}_{w\to f}$ */
  $$\Delta\theta_{f\to v}(X_v) = \sum_{X_w \in \mathcal{U}^{t-1}_{w\to f}} \sum_{X_u : u \in \mathcal{N}_f \setminus \{v,w\}} \Bigg(\psi_f(\mathcal{N}_f)\, \tilde{\mu}_{w\to f}(X_w) \times \prod_{m \in \mathcal{N}_{\text{updated}}} \hat{\mu}^{t-1}_{m\to f}(X_m) \prod_{m \in (\mathcal{N}_f \setminus \{v,w\}) \setminus \mathcal{N}_{\text{updated}}} \hat{\mu}^{t-2}_{m\to f}(X_m) \Bigg);$$
  $\theta^t_{f\to v}(X_v) \leftarrow \theta^t_{f\to v}(X_v) + \Delta\theta_{f\to v}(X_v)$;
  $\mathcal{N}_{\text{updated}} \leftarrow \mathcal{N}_{\text{updated}} \cup \{w\}$;
end
return $\theta^t_{f\to v}(X_v)$

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 $X_{v_1} \notin \mathcal{U}^{t-1}_{v_1\to f}$. Therefore, we have that

$$\Delta\theta^t_{f\to v_2}(X_{v_2}) = \sum_{X_{v_1} \in \mathcal{U}^{t-1}_{v_1\to f}} \psi_f(X_{v_1}, X_{v_2})(\Delta\hat{\mu}^{t-1}_{v_1\to f}(X_{v_1})), \tag{1.5}$$

where we have that

$$\Delta\theta^t_{f\to v_2}(X_{v_2}) = \theta^t_{f\to v_2}(X_{v_2}) - \theta^{t-1}_{f\to v_2}(X_{v_2})$$

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 $|\mathcal{U}^{t-1}_{v_1\to f}|/D$ of the full Sum-Product update complexity. The generalization for this update for function nodes that have degree greater than 2 is given in Algorithm 1.6. 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$(f \to v)$ as in Equation (1.4). This is hinted at by the fact that we initialize $\mathcal{U}^0_{v\to f} = \mathcal{X}_v$. Of course, this is equivalent to a Sum-Product update of $\theta^t_{f\to v}(X_v)$, and is therefore not a disadvantage of Projected BP compared to Sum-Product.

1.4.2 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 $K \leq D$ 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

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

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

$$\mathcal{U}_{v\to f}^{t} = \arg\min_{\mathcal{U}} |\mathcal{U}|$$

$$\text{subject to} \tag{1.6}$$

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

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

1.4.3 Computation and Communication Complexity

We now examine more carefully the complexity of our Projected BP algorithms, in terms of both computation and communication. We will also examine the complexity of the Sum-Product algorithm in order to make a comparison.

First, consider the computational complexity of the updates of the variable to function beliefs $\mu_{v\to f}^{t}(X_v)$, which is the same for both Sum-Product and both versions of Projected BP. The update for a single edge simply involves $|\mathcal{N}_v| - 1$ multiplies for each of D belief elements, followed by $D-1$ additions and D multiplications or divisions for the normalization step. Therefore, the overall computational complexity for this portion of the computation, in terms of the variable cardinality D, is simply $\mathcal{O}(D)$.

Next, consider the computational complexity of the updates of the function to variable beliefs $\mu_{f\to v}^{t}(X_v)$ in the Sum-Product algorithm. Reiterating Equation (1.3), we have that

$$\underbrace{\theta_{f\to v}^{t}(X_v)}_{D \text{ elements}} = \overbrace{\sum_{X_u : u \in \mathcal{N}_f \setminus v} \underbrace{\left(\psi_f(\mathcal{N}_f) \prod_{w \in \mathcal{N}_f \setminus v} \mu_{w\to f}^{t-1}(X_w) \right)}_{|\mathcal{N}_f|-1 \text{ multiplies}}}^{D^{(|\mathcal{N}_f|-1)} \text{ terms}}.$$

Therefore, the number of multiplies, when naively computed, is $D \times D^{(|\mathcal{N}_f|-1)} \times$

$(|\mathcal{N}_f| - 1) = (|\mathcal{N}_f| - 1)D^{|\mathcal{N}_f|}$. The number of additions is comparable, coming to $D \times (D^{(|\mathcal{N}_f|-1)} - 1)$. Hence, we have that the overall computational complexity of the update of $\theta^t_{f\to v}(X_v)$ is $\mathcal{O}(D^{|\mathcal{N}_f|})$, with respect to D. We will mention briefly that we may employ some tricks to reduce slightly the number of computations, such as maintaining partial products in order to use fewer than $|\mathcal{N}_f| - 1$ multiplies per summation term, but the computational complexity remains $\mathcal{O}(D^{|\mathcal{N}_f|})$. Now, over the whole graph, we may conclude that the overall computational complexity of a 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. Of course, the complexity also scales with the size of the graph and the number of nodes with a particular degree, but the differences in computational complexity that we will observe between Sum-Product and our Projected BP algorithms are only in the parameter D and the parameter K of K-Projected BP.

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. This time, reiterating Lines 7 and 8 of Algorithm 1.6, we have that

$$\underbrace{\theta^t_{f\to v}(X_v)}_{D \text{ elements}} \leftarrow \overbrace{\sum_{X_w \in \mathcal{U}^{t-1}_{w\to f}} \overbrace{\sum_{X_u : u \in \mathcal{N}_f \setminus \{v,w\}} \underbrace{\psi_f(\mathcal{N}_f)\, \underbrace{\tilde{\mu}_{w\to f}(X_w)}_{K \text{subtractions}}}_{2 \text{ multiplies}} \underbrace{\prod_{m \in \mathcal{N}_f \setminus \{w,v\}} \hat{\mu}^{t[m]}_{m\to f}(X_m)}_{|\mathcal{N}_f|-3 \text{ multiplies}}}^{D^{(|\mathcal{N}_f|-2)} \text{ terms}}}^{K \text{ terms}} \underbrace{+\theta^t_{f\to v}(X_v)}_{D \text{ additions}}.$$

We have combined the products over the sets $(\mathcal{N}_f \setminus \{v, w\}) \setminus \mathcal{N}_{\text{updated}}$ and $\mathcal{N}_{\text{updated}}$ by defining $t[m] = t - 1$ if $m \in \mathcal{N}_{\text{updated}}$, otherwise $t[m] = t - 2$. Therefore, the number of multiplies for the full update of $\theta^t_{f\to v}(X_v)$ is $D \times K \times D^{(|\mathcal{N}_f|-2)} \times (|\mathcal{N}_f| - 1)$, which is $\mathcal{O}(KD^{(|\mathcal{N}_f|-1)})$ with respect to K and D. The number of additions and subtractions is on the same order, coming to $K(D^{(|\mathcal{N}_f|-1)} + 1)$. Of course, this also dominates the $\mathcal{O}(D)$ complexity of the variable to function message updates, so the overall computational complexity of an iteration of K-Projectd BP, with respect to K and D, is

$\mathcal{O}(KD^{(N_{\max}-1)})$. Furthermore, note that if we employ a flooding message passing schedule, we need to perform this update of $\theta^t_{f\to v}(X_v)$ separately for each updated message $\hat{\mu}^{t-1}_{w\to f}(X_w)$, $w \in \mathcal{N}_f \setminus v$, contributing an additional constant factor $|\mathcal{N}_f| - 1$ to the complexity. This does not change the computational complexity of the iterations with respect to the parameters K and D.

Note that the most significant scaling constant ignored in the order notation that is different between Sum-Product BP and K-Projected BP is the factor of $(|\mathcal{N}_f| - 1)$ resulting from separate updates for each variable to function message that $\theta^t_{f\to w}(X_w)$ depends on in a flooding message passing scheme. 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.

1.4.4 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 [2] and [3], 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 [3] and for Residual BP [2], 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.

Correspondence of Fixed Points

Our first theoretical results concern the fixed points of our Projected BP algorithms. Essentially, these results say that every fixed point of Sum-Product corresponds with a unique fixed point of Projected BP, and every

fixed point of Projected BP corresponds with a unique fixed point of Sum-Product.

To demonstrate this, we will consider flooding message passing schedules for both Sum-Product BP and Projected BP. In the case of Sum-Product, we will employ normalization on the variable to function messages, but not on the function to variable messages. Let $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}$ be the concatenation of all variable to function messages and let $\mathcal{M}_{\mathcal{F}\to\mathcal{V}}$ be the concatenation of all function to variable messages. 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 at the receiving end of the variable to function channels. 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 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 $\mathcal{M}^t_{\mathcal{V}\to\mathcal{F}}$. We can now state the following theorem.

Theorem 1.4.1. *A state space point $\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}$ is a fixed point of Sum-Product if and only if it is a fixed point of Projected BP.*

Proof. Assume that $\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}$ is a fixed point of Sum-Product. Then we must

have that

$$\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}} = G(F(\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}))$$

Then, applying the Projected BP update to the Sum-Product fixed point, we see that

$$\begin{aligned} U(G(F(\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}})), \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}) &= U(\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}, \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}) \\ &= \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}. \end{aligned}$$

This shows that $\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}$ is also a fixed point of Projected BP.

Conversely, suppose that $\mathcal{M}^\star_{\mathcal{V}\to\mathcal{F}}$ is a fixed point of Projected BP. Then we must have that

$$\mathcal{M}^\star_{\mathcal{V}\to\mathcal{F}} = U(G(F(\mathcal{M}^\star_{\mathcal{V}\to\mathcal{F}})), \mathcal{M}^\star_{\mathcal{V}\to\mathcal{F}}).$$

However, since the second argument of $U()$ is equal to the output of $U()$, we know that the arguments of $U()$ must be equal. Hence, we have that

$$\mathcal{M}^\star_{\mathcal{V}\to\mathcal{F}} = G(F(\mathcal{M}^\star_{\mathcal{V}\to\mathcal{F}})),$$

which implies that $\mathcal{M}^\star_{\mathcal{V}\to\mathcal{F}}$ is also a fixed point of Sum-Product. Thus, we have that Sum-Product and Projected BP have the same fixed points. ■

Conditions for Guaranteed Convergence to Fixed Points

We have established that Sum-Product and Projected BP have the same fixed points. We will now explore whether Projected BP will converge to these fixed points. We will now establish that under standard assumptions, similar to those of other related approaches [2, 3], β-Projected BP will converge to a fixed point, as long as Sum-Product is guaranteed to converge and the β parameter is small enough. Our methods are similar to the methods of analysis for Residual BP [2] in that we first establish some basic properties of iterative algorithms and then proceed to apply these results to Projected BP.

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 $\mathcal{B}_N(r, x) = \{x' : \|x - x'\|_N \leq r\}$, which is simply a ball of radius r around the point x with respect to the norm N.

Assumption 1.4.2 (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 1.4.2 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 for convergence analysis of Residual BP [2] and Stochastic BP [3], and results that hold under Assumption 1.4.2 will therefore also hold under a contractivity assumption.

Lemma 1.4.3. *Suppose the following are true:*

- *Iterative algorithm $f()$ satisfies Assumption 1.4.2.*
- *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

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

Proof. Using Assumption 1.4.2 and the triangle inequality, we have that

$$\begin{aligned}
\|f(x) - x\|_N &= \|f(x) - x^* + x^* - x\|_N \\
&\leq \|f(x) - x^*\|_N + \|x^* - x\|_N \\
&\leq \alpha\|x - x^*\|_N + \|x - x^*\|_N \\
&= (1+\alpha)\|x - x^*\|_N \\
\Rightarrow \|f(x) - x\|_N &\leq (1+\alpha)\|x - x^*\|_N.
\end{aligned}$$

This then allows us to show that

$$\begin{aligned}
\|g(x) - x^*\|_N &= \|g(x) - f(x) + f(x) - x^*\|_N \\
&\leq \|g(x) - f(x)\|_N + \|f(x) - x^*\|_N \\
&\leq \|e(x)\|_N + \alpha\|x - x^*\|_N \\
&\leq \beta\|f(x) - x\|_N + \alpha\|x - x^*\|_N \\
&\leq \beta(1+\alpha)\|x - x^*\|_N + \alpha\|x - x^*\|_N \\
&= \left(\beta\left(1+\alpha\right) + \alpha\right)\|x - x^*\|_N \\
\Rightarrow \|g(x) - x^*\|_N &\leq \left(\beta\left(1+\alpha\right) + \alpha\right)\|x - x^*\|_N.
\end{aligned}$$

But, we have that

$$\begin{aligned}
\beta\left(1+\alpha\right) + \alpha &< \left(\frac{1-\alpha}{1+\alpha}\right)(1+\alpha) + \alpha \\
&= (1-\alpha) + \alpha \\
&= 1 \\
\Rightarrow \beta\left(1+\alpha\right) + \alpha &< 1.
\end{aligned}$$

Hence, for $x^0 \in \mathcal{B}_N(r, x^*)$ and $x^t = g(x^{t-1})$, we have that

$$\|x^t - x^*\|_N \leq r(\beta\left(1+\alpha\right) + \alpha)^t. \qquad \blacksquare$$

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 1.4.3 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 1.4.4. Consider any two global state space points $\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}$ and $\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}$. 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 1.4.5. *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 1.4.2 (with $x^* = \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}$).*
- *the message and global norms on $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}$ satisfy Assumption 1.4.4.*
- *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

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

Proof. We simply need to verify that β-Projected BP can be written as

$$g(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) = f(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) + e(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}), \tag{1.7}$$

and that

$$\|e(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})\|_N \leq \beta\|f(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) - \mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_N \tag{1.8}$$

when $\beta \in \left[0, \left(\frac{1-\alpha}{1+\alpha}\right)\right)$, which allows us to apply Lemma 1.4.3. To this end, using the notation from above and writing the subset selections $\mathcal{U}_{v\to f}$ (a function of $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}$) as $\mathcal{U}_i$ where i is the index of edge (v, f), we have that

$$[g(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})]_{i,j} = \begin{cases} [f(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})]_{i,j} & \text{if } j \in \mathcal{U}_i \\ [\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_{i,j} & \text{if } j \notin \mathcal{U}_i. \end{cases}$$

Therefore, if we define

$$[e(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})]_{i,j} = \begin{cases} 0 & \text{if } j \in \mathcal{U}_i \\ [\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_{i,j} - [f(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})]_{i,j} & \text{if } j \notin \mathcal{U}_i, \end{cases}$$

this verifies that the algorithms satisfy Equation (1.7).

Now, to verify Equation (1.8), note that the subset selection of β-Projected BP in Equation (1.6) ensures that

$$\|[e(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})]_i\|_{N_i} \leq \beta\|[f(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) - \mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_i\|_{N_i}.$$

Assumption 1.4.4 then leads to verification of Equation (1.8). Finally, application of Lemma 1.4.3 concludes the proof. ■

We now briefly consider what global norms satisfy Assumption 1.4.4.

Claim 1.4.6. Let the individual message norms be the usual Euclidean norm, i.e.,

$$\| [\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_i \|_{N_i} = \| [\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_i \|_2 = \sqrt{\sum_j ([\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_{i,j})^2}.$$

The Euclidean norm for the global message, $\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_2$, defined as

$$\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_2 = \sqrt{\sum_{(i,j)} ([\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_{i,j})^2},$$

satisfies Assumption 1.4.4.

Proof. Consider any two global state space points $\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}$ and $\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}$, and suppose that

$$\|[\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}]_i\|_2 \leq \beta\|[\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}]_i\|_2$$

for every component message. Then we have that

$$\begin{aligned}
\left(\|[\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}]_i\|_2\right)^2 &\leq \left(\beta\|[\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}]_i\|_2\right)^2 \\
\Rightarrow \sum_i \left(\|[\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}]_i\|_2\right)^2 &\leq \sum_i \left(\beta\|[\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}]_i\|_2\right)^2 \\
\Rightarrow \sum_i \sum_j \left(\left[\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}\right]_{i,j}\right)^2 &\leq \sum_i \beta^2 \sum_j \left(\left[\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}\right]_{i,j}\right)^2 \\
&= \beta^2 \sum_i \sum_j \left(\left[\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}\right]_{i,j}\right)^2 \\
\Rightarrow \sqrt{\sum_{(i,j)} \left(\left[\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}\right]_{i,j}\right)^2} &\leq \beta\sqrt{\sum_{(i,j)} \left(\left[\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}\right]_{i,j}\right)^2} \\
\Rightarrow \|\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}\|_2 &\leq \beta\|\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}\|_2.
\end{aligned}$$

■

Claim 1.4.7. 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

$$\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_C = \max_i \|[\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_i\|_{N_i},$$

satisfies Assumption 1.4.4.

Proof. Consider any two global state space points $\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}$ and $\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}$, and suppose 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 component message. Then we have that

$$\begin{aligned}
\|\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}\|_C &= \max_i \left\|\left[\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}\right]_i\right\|_{N_i} \\
&\leq \max_i \beta \left\|\left[\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}\right]_i\right\|_{N_i} \\
&= \beta \max_i \left\|\left[\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}\right]_i\right\|_{N_i} \\
&= \beta\|\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}\|_C \\
\Rightarrow \|\mathcal{M}^1_{\mathcal{V}\to\mathcal{F}}\|_C &= \beta\|\mathcal{M}^2_{\mathcal{V}\to\mathcal{F}}\|_C.
\end{aligned}$$

■

Claim 1.4.8. 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 1.4.4.

Proof. We have that

$$\begin{aligned}
\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_\infty &= \max_{(i,j)} \left| [\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_{i,j} \right| \\
&= \max_i \max_j \left| [\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_{i,j} \right| \\
&= \max_i \| [\mathcal{M}_{\mathcal{V}\to\mathcal{F}}]_i \|_{N_i} \\
&= \|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_C.
\end{aligned}$$

Therefore, the global max norm is a special case of the composite norm of Claim 1.4.7. ■

Theorem 1.4.5 demonstrates that if the convergence property of Assumption 1.4.2 holds for Sum-Product with respect to any one of a broad class of norms satisfying Assumption 1.4.4, then there is a β such that β-Projected BP is also convergent. However, there remains the possibility that better guarantees could be given if we consider particular global norms N. In the following, similar to the convergence analysis of Stochastic BP [3], we will consider exponential convergence in the form of Assumption 1.4.2 with respect to the Euclidean norm $\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_2$. We will also consider Assumption 1.4.2 with respect to the max norm $\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_\infty$. Note that Theorem 1.4.5 already applies to these cases, as demonstrated by Claims 1.4.6 and 1.4.8. In addition to specializing the results with respect the the norms under consideration, we will also take advantage of the particular structure of the perturbation term $e(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})$ to widen the range of values of β for which convergence of β-Projected BP is guaranteed.

First, we consider the Euclidean global norm.

Lemma 1.4.9. *Suppose the following are true:*

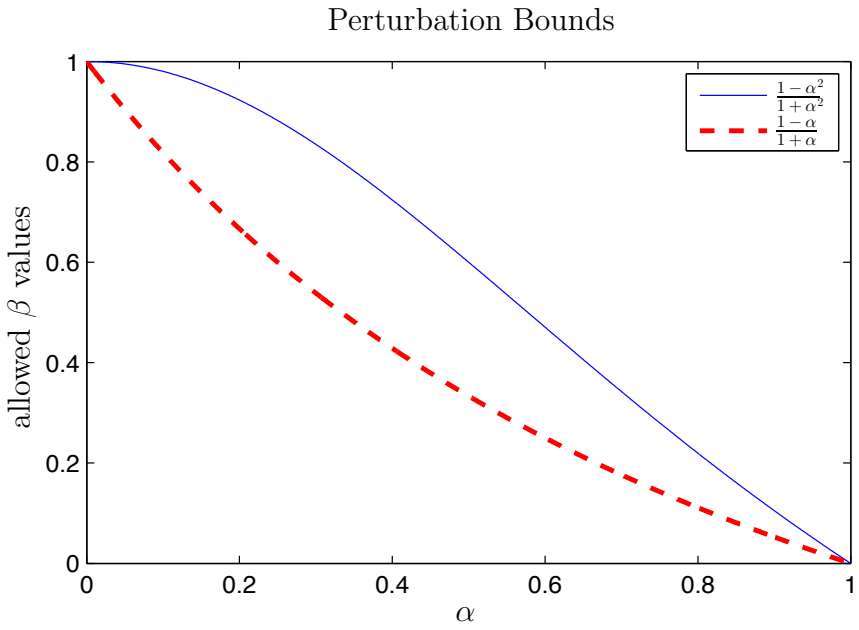


Figure 1.1: Different upper bounds on the relative perturbation size, β, for which convergence is guaranteed, as a function of α. Note that the bound specialized for the Euclidean norm includes a wider range of β values that guarantee convergence.

- *The norm N is the (global) Euclidian norm, as defined in Claim 1.4.6*
- *Iterative algorithm $f()$ satisfies Assumption 1.4.2 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 $\beta \in [0, \cos(2\tan^{-1}(\alpha))) = \left[0, \frac{1-\alpha^2}{1+\alpha^2}\right)$, 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

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

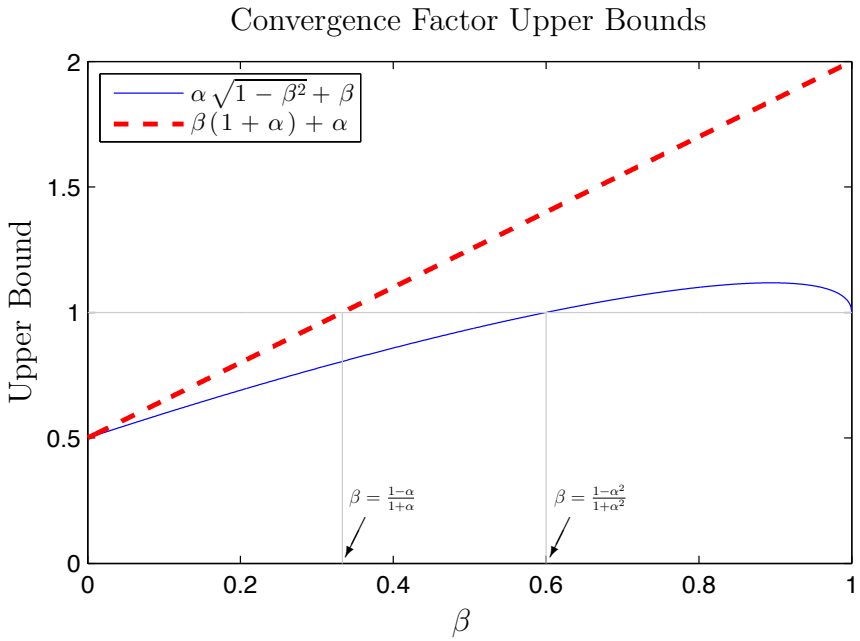


Figure 1.2: Different upper bounds on the convergence factor, where $\alpha = \frac{1}{2}$. A convergence factor less than 1 guarantees exponential convergence, with smaller factors corresponding with faster convergence. Values of β where the bounds transition to above 1 are labeled.

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 1.4.9 achieves a better bound on convergence rate than Lemma 1.4.3 by specifically accounting for 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 1.4.9 guarantees exponential convergence of β-Projected BP for a wider range of β values than Lemma 1.4.3.

Proof. We need to verify that algorithm g reduces the distance of the state from x^* by a factor at least as fast as $\alpha\sqrt{1-\beta^2} + \beta$, and that this factor is less than 1 for the specified range of β. First, let us define the notation $[x]_i$ as the i^{th} element of x. Next, note that the particular form of $e(x)$ guarantees that $g(x)$ lies on a sphere centered at $\frac{1}{2}(f(x)+x)$ with radius equal to $\left\|\frac{f(x)}{2} - \frac{x}{2}\right\|_2$. We will use the notation $\mathcal{S}_N(r,c)$ to indicate a sphere

of radius r centered at c under norm N, i.e.,

$$\mathcal{S}_N(r,c) = \{x : \|x-c\|_N = r\}.$$

To see that $g(x)$ is on the sphere $\mathcal{S}_2\left(\left\|\frac{f(x)}{2}-\frac{x}{2}\right\|_2, \frac{f(x)}{2}+\frac{x}{2}\right)$, first note that

$$[e(x)]_i = \begin{cases} 0 & \text{if } [b(x)]_i = 0 \\ [x-f(x)]_i & \text{if } [b(x)]_i = 1 \end{cases}$$

Therefore, we have that

$$\begin{aligned}\left[g(x)-\frac{1}{2}(f(x)+x)\right]_i &= \left[f(x)+e(x)-\frac{f(x)}{2}-\frac{x}{2}\right]_i \\ &= \left[\frac{f(x)}{2}-\frac{x}{2}+e(x)\right]_i \\ &= \begin{cases} \left[\frac{f(x)}{2}-\frac{x}{2}\right]_i & \text{if } [b(x)]_i = 0 \\ \left[\frac{x}{2}-\frac{f(x)}{2}\right]_i & \text{if } [b(x)]_i = 1 \end{cases}\end{aligned}$$

Hence, we have that

$$\begin{aligned}\sqrt{\sum_i \left[g(x)-\frac{1}{2}(f(x)+x)\right]_i^2} &= \sqrt{\sum_i \left[\frac{f(x)}{2}-\frac{x}{2}\right]_i^2} \\ &= \left\|\frac{f(x)}{2}-\frac{x}{2}\right\|_2.\end{aligned}$$

We continue the proof by examining how far $g(x)$ can be from x^* in the Euclidean norm. To this end, for a given α and β, consider

$$\tilde{\alpha} = \max_{x,x^*,f,g} \frac{\|g-x^*\|_2}{\|x-x^*\|_2} \text{ subject to } \begin{cases} f \in \mathcal{B}\left(\alpha\|x-x^*\|_2, x^*\right) \\ g \in \mathcal{S}\left(\frac{1}{2}\|f-x\|_2, \frac{1}{2}(f+x)\right) \\ g \in \mathcal{B}\left(\beta\|f-x\|_2, f\right) \end{cases}.$$

The first constraint comes from Assumption 1.4.2, the second from the assumption on the form of the perturbation, and the third constraint comes from the assumption on perturbation size. Thus, $\tilde{\alpha}\|x-x^*\|_2$ is an upper bound on $\|g-x^*\|_2$. Now, note that we may arbitrarily translate, scale, and rotate the coordinate system without affecting the value of $\tilde{\alpha}$ in this max-

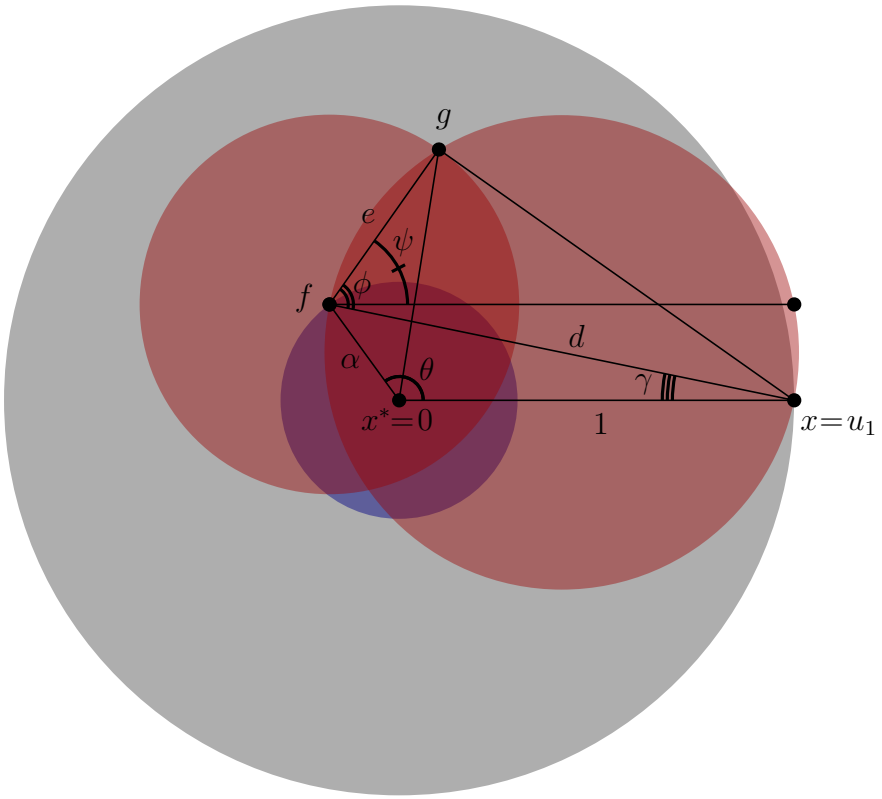


Figure 1.3: Geometry of x, f, and g.

imization. Furthermore, the maximization will force f and g to be on the boundaries of their respective ball constraints, restricting their values to the corresponding spheres. Finally, the maximizing value of g will be in the same plane as x, x^*, and f, such that we only need to consider the maximization in two dimensions. Therefore, we translate the coordinates so that $x^* = [0,0]^T$, and both scale and rotate the coordinates so that $x = [1,0]^T \triangleq u_1$, f is on the circle $\mathcal{S}(\alpha, 0)$, and g is on the circle $\mathcal{S}\left(\frac{1}{2}\|f-u_1\|_2, \frac{1}{2}(f+u_1)\right)$. Then our expression for $\tilde{\alpha}$ becomes

$$\tilde{\alpha} = \max_{f,g} \|g\|_2 \text{ subject to } \begin{cases} f \in \mathcal{S}(\alpha, 0) \\ g \in \mathcal{S}\left(\frac{1}{2}\|f-u_1\|_2, \frac{1}{2}(f+u_1)\right) \\ g \in \mathcal{S}(\beta\|f-u_1\|_2, f) \end{cases}.$$

This situation is depicted in Figure 1.3. In the figure, we have labeled the points x, x^*, f, and g, the angles θ, ψ, and ϕ, and the lengths α, e (the perturbation size), and d. We can now see that we may find an expression for g as a function of α, β, and the angle θ. From this, we maximize with

respect to θ. From the figure, we have that

$$f = [\alpha\cos(\theta), \alpha\sin(\theta)]^T.$$

We can also compute the angle γ, and find that

$$\gamma = \tan^{-1}\left(\frac{\alpha\sin(\theta)}{1-\alpha\cos(\theta)}\right).$$

The angle ϕ is found to be

$$\phi = \cos^{-1}\left(\frac{e}{d}\right) = \cos^{-1}(\beta).$$

The length e is found to be

$$\begin{aligned} e &= \beta d \\ &= \beta\sqrt{(\alpha\sin(\theta))^2 + (1+\alpha\cos(\theta))^2}. \end{aligned}$$

We then find that

$$\begin{aligned} g &= f + [e\cos(\psi), e\sin(\psi)]^T \\ &= f + [e\cos(\phi-\gamma), e\sin(\phi-\gamma)]^T \\ &= [\alpha\cos(\theta) + e\cos(\phi-\gamma), \alpha\sin(\theta) + e\sin(\phi-\gamma)]^T \end{aligned}$$

Hence, we have that

$$\begin{aligned} \|g\|_2^2 = &\left(\alpha\cos(\theta) + e\cos(\phi-\gamma)\right)^2 \\ &+ \left(\alpha\sin(\theta) + e\sin(\phi-\gamma)\right)^2. \end{aligned} \tag{1.9}$$

Substituting the expressions for e, ϕ, and γ leads to a complicated expression, but with the judicious, yet liberal, application of numerous trigonometric identities, Equation (1.9) may be transformed into

$$\|g\|_2^2 = \alpha^2 + \beta^2 - \alpha^2\beta^2 + 2\alpha\beta\sin(\theta)\sqrt{1-\beta^2}.$$

This is maximized for $\theta = \frac{\pi}{2}$, which leads to

$$\begin{aligned}\tilde{\alpha} &= \max_{f,g} \|g\|_2 \\ &= \sqrt{\alpha^2 + \beta^2 - \alpha^2\beta^2 + 2\alpha\beta\sqrt{1-\beta^2}} \\ &= \alpha\sqrt{1-\beta^2} + \beta.\end{aligned}$$

It remains to show that $\tilde{\alpha} < 1$ for $\beta \in \left[0, \frac{1-\alpha^2}{1+\alpha^2}\right)$. To this end, we will show the following:

1. $\alpha\sqrt{1-\beta^2} + \beta = \alpha$ for $\beta = 0$.
2. $\alpha\sqrt{1-\beta^2} + \beta = 1$ for $\beta = \frac{1-\alpha^2}{1+\alpha^2}$.
3. $\alpha\sqrt{1-\beta^2} + \beta$ is strictly increasing in β for each fixed value of α.

For $\beta = 0$, clearly we have that $\alpha\sqrt{1-0^2} + 0 = \alpha$. Next, for $\beta = \frac{1-\alpha^2}{1+\alpha^2}$, we have that

$$\begin{aligned}\alpha\sqrt{1-\beta^2} + \beta &= \alpha\sqrt{1 - \left(\frac{1-\alpha^2}{1+\alpha^2}\right)^2} + \frac{1-\alpha^2}{1+\alpha^2} \\ &= \alpha\sqrt{\frac{(1+\alpha^2)^2 - (1-\alpha^2)^2}{(1+\alpha^2)^2}} + \frac{1-\alpha^2}{1+\alpha^2} \\ &= \alpha\sqrt{\frac{4\alpha^2}{(1+\alpha^2)^2}} + \frac{1-\alpha^2}{1+\alpha^2} \\ &= \frac{2\alpha^2}{1+\alpha^2} + \frac{1-\alpha^2}{1+\alpha^2} \\ &= \frac{1+\alpha^2}{1+\alpha^2} \\ &= 1.\end{aligned}$$

Finally, we have that

$$\frac{d}{d\beta}\left(\alpha\sqrt{1-\beta^2} + \beta\right) = 1 - \frac{\alpha\beta}{\sqrt{1-\beta^2}},$$

which we must verify is positive. We begin with

$$1 \geq \frac{(1-\alpha^2)^2}{1+\alpha^2} = (1+\alpha^2)\left(\frac{1-\alpha^2}{1+\alpha^2}\right)^2.$$

Noting that $\beta < \frac{1-\alpha^2}{1+\alpha^2}$, we have that

$$1 > (1+\alpha^2)\beta^2.$$

Which then gives us

$$1 - \beta^2 > \alpha^2\beta^2.$$

Since the square root function is monotone increasing, we have that

$$\sqrt{1-\beta^2} > \alpha\beta.$$

We can now divide by $\sqrt{1-\beta^2}$ on both sides to get

$$1 > \frac{\alpha\beta}{\sqrt{1-\beta^2}}.$$

And finally we have that

$$1 - \frac{\alpha\beta}{\sqrt{1-\beta^2}} = \frac{d}{d\beta}\left(\alpha\sqrt{1-\beta^2} + \beta\right) > 0.$$

Therefore, $\alpha\sqrt{1-\beta^2} + \beta$ is strictly increasing in β from α to 1 for each α. Hence, we have that $\alpha\sqrt{1-\beta^2} + \beta < 1$ for $\beta \in \left[0, \frac{1-\alpha^2}{1+\alpha^2}\right)$, which gives us the exponential convergence of algorithm g. ■

We will now apply Lemma 1.4.9 to β-Projected BP.

Theorem 1.4.10. *Suppose the following are true:*

- *The global and message norms, N and N_i respectively, are the Euclidian norms, as defined in Claim 1.4.6*
- *The Sum-Product iteration $f(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) \triangleq G(F(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}))$ satisfies Assumption 1.4.2 (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*

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

The proof of this result involves essentially the same procedure as the proof of Theorem 1.4.5, but with application of Lemma 1.4.9 instead of Lemma 1.4.3. We note that the specified Euclidean norms satisfy Assumption 1.4.4, as demonstrated in Claim 1.4.6.

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 1.4.11. *Suppose the following are true:*

- *The norm N is the (global) max norm, as defined in Claim 1.4.8*
- *Iterative algorithm $f()$ satisfies Assumption 1.4.2 with respect to the particular norm N.*
- *We have that the iterative algorithm $g(x)$ is a perturbed version of $f(x)$, i.e., $g(x) = f(x) + e(x)$.*
- *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 [2]. In fact, the proof method for this lemma can be used to prove the mentioned corollary.

Proof. First, let us write the sequence of states for the iterations of algorithm g as x^t, where x^0 is some arbitrary initial state within $\mathcal{B}_{r,N}(x^*)$, and $x^{t+1} =$

$g(x^t)$. Now, note that

$$\left|\left[f(x^t) - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty, \forall i,$$

which follows directly from Assumption 1.4.2 under the max norm. We now note that

$$\left|\left[x^t - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty \text{ implies } \left|\left[x^{t+1} - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty. \quad (1.10)$$

This is true because, due to the form of the perturbation $e(x)$, we have that

$$\left[x^{t+1}\right]_i = \left[g(x^t)\right]_i = \begin{cases} [f(x^t)]_i & \text{if } [b(x^t)]_i = 1 \\ [x^t]_i & \text{if } [b(x^t)]_i = 0 \end{cases}.$$

Since we have that

$$\left|\left[f(x^t) - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty,$$

and

$$\left|\left[x^t - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty,$$

this proves the intermediate result in Equation (1.10). On the other hand, we have that

$$\|g(x) - x^*\|_\infty \leq \|x - x^*\|_\infty.$$

This is true because

$$\begin{aligned} \|g(x) - x^*\|_\infty &= \max_i |[g(x) - x^*]_i| \\ &= \max_i |[f(x) + e(x) - x^*]_i| \\ &\leq \max\left(\max_i |[f(x) - x^*]_i|, \max_i |[x - x^*]_i|\right) \\ &= \max_i |[x - x^*]_i| \\ &= \|x - x^*\|_\infty. \end{aligned}$$

Iterating this on the state sequence x^t, starting with state x^{t_0}, results in

$$\left\|x^t - x^*\right\|_\infty \leq \left\|x^{t_0} - x^*\right\|_\infty \text{ for } t \in \mathbb{Z}, t \geq t_0. \quad (1.11)$$

This allows us to iterate Equation (1.10) to obtain

$$\begin{aligned} &\left|\left[x^t - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty \\ \Rightarrow &\left|\left[x^k - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty \text{ for } k \in \mathbb{Z}, k \geq t. \end{aligned}$$

We show this by induction. The base case is Equation (1.10). Now, suppose, for $k > t$, we have that

$$\begin{aligned} &\left|\left[x^t - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty \\ \Rightarrow &\left|\left[x^k - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty . \end{aligned}$$

From Equation (1.11), we have that $\left\|x^k - x^*\right\|_\infty \leq \left\|x^t - x^*\right\|_\infty$, since $k \geq t$. Combining this with Equation (1.10), we have that

$$\begin{aligned} &\left|\left[x^k - x^*\right]_i\right| \leq \alpha \left\|x^k - x^*\right\|_\infty \leq \alpha \left\|x^t - x^*\right\|_\infty \\ \Rightarrow &\left|\left[x^{k+1} - x^*\right]_i\right| \leq \alpha \left\|x^k - x^*\right\|_\infty \leq \alpha \left\|x^t - x^*\right\|_\infty . \end{aligned}$$

Therefore, we have that

$$\begin{aligned} &\left|\left[x^t - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty \\ \Rightarrow &\left|\left[x^{k+1} - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty , \end{aligned}$$

which completes the induction step. One more application of Equation (1.11) results in

$$\begin{aligned} &\left|\left[x^t - x^*\right]_i\right| \leq \alpha \left\|x^{t_0} - x^*\right\|_\infty \\ \Rightarrow &\left|\left[x^k - x^*\right]_i\right| \leq \alpha \left\|x^{t_0} - x^*\right\|_\infty \text{ for } k \geq t \geq t_0. \end{aligned}$$

More specifically, we have that

$$\begin{aligned} &\left|\left[x^t - x^*\right]_i\right| \leq \alpha \left\|x^0 - x^*\right\|_\infty \\ \Rightarrow &\left|\left[x^k - x^*\right]_i\right| \leq \alpha \left\|x^0 - x^*\right\|_\infty \text{ for } k \geq t \geq 0. \end{aligned}$$

Essentially, this says that if g updates element i of the state to be at most $\alpha \left\|x^0 - x^*\right\|_\infty$ away from x^*, then it will forever stay within that distance of the fixed point. Since we have assumed that every element of the state gets

updated by the completion of iteration M, then we have that

$$\left|\left[x^M - x^*\right]_i\right| \leq \alpha \left\|x^0 - x^*\right\|_\infty \forall i.$$

Finally, this implies that

$$\left\|x^M - x^*\right\|_\infty \leq \alpha \left\|x^0 - x^*\right\|_\infty .$$

By applying this to every subsequence of M consecutive iterations of g, we are able to conclude the proof of Lemma 1.4.11. ■

At this point, we note that Lemma 1.4.11 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 on the 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 1.4.11 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 1.4.2 with respect to the max norm, then Lemma 1.4.11 implies that we stand to gain in convergence rate by breaking up the updates into fine-grained element-by-element updates, quite analogous to the arguments made in support of Residual BP in [2].

Convergence in Trees in Finite Iterations

We will now change direction a bit and 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 algorithm converges to a unique fixed point in a finite number of iterations for factor trees[1], i.e., has no cycles [29]. 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.

[1]Whenever we mention a tree, it shall be assumed that it is a finite tree.

For the following theorem, we will 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 1.4.12. *In a factor tree of finite diameter d and variable nodes of cardinality D, the K-Projected BP update,*

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

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.

Proof. We begin by defining some notation. Let $\mathbb{T}(v \to f)$ be the subtree rooted at node f through v, i.e., the subtree containing all nodes with paths to f that necessarily include the edge (v, f). Let $d_*(v \to f)$ be the depth of $\mathbb{T}(v \to f)$, i.e., the maximal distance from f to a leaf node of $\mathbb{T}(v \to f)$. We will say that a message or intermediate computation, such as $\hat{\mu}^t_{v\to f}$, is *stable* from iteration t if its value does not change with further iterations of the algorithm after iteration t. For example, if $\hat{\mu}^{t+k}_{v\to f} = \hat{\mu}^t_{v\to f}$ for $k \geq 0$, then $\hat{\mu}^t_{v\to f}$ is stable from iteration t. Note that if, from iteration t, message $[\mathcal{M}^t_{\mathcal{V}\to\mathcal{F}}]_i$ is stable for every component i, then $\mathcal{M}^t_{\mathcal{V}\to\mathcal{F}} = \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}$. Thus, the claim of the theorem is that $\left[\mathcal{M}^{t^*}_{\mathcal{V}\to\mathcal{F}}\right]_i$ is stable from t^* for every i, where we have that $t^* \leq \frac{d}{2}\lceil\frac{D}{K}\rceil$.

Before we proceed, we will define some intermediate computations implicit in $\mathcal{M}^{t+1}_{\mathcal{V}\to\mathcal{F}} = g(\mathcal{M}^t_{\mathcal{V}\to\mathcal{F}})$. First, let

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

which consists of all the function to variable messages $\left[\mathcal{M}^{t+1}_{\mathcal{F}\to\mathcal{V}}\right]_i = \mu^{t+1}_{f\to v}$. Also, let

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

which consists of all the (exact) variable to function messages $\left[\tilde{\mathcal{M}}^{t+1}_{\mathcal{V}\to\mathcal{F}}\right]_i = \mu^{t+1}_{v\to f}$. Finally, of course, we have that

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

which consists of all the (estimated) variable to function messages $\left[\mathcal{M}^{t+1}_{\mathcal{V}\to\mathcal{F}}\right]_i = \hat{\mu}^{t+1}_{v\to f}$.

Now, consider a particular variable to function message estimate $[\mathcal{M}^t_{\mathcal{V}\to\mathcal{F}}]_i = \hat{\mu}^t_{v\to f}$. We will first find an upper bound on the number of iterations t until $[\mathcal{M}^t_{\mathcal{V}\to\mathcal{F}}]_i = [\mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}]_i$. By induction on the factor tree, we will prove that $\hat{\mu}^t_{v\to f}$ is stable from $t^*_{v\to f}$, where $t^*_{v\to f} \leq \frac{d_*(v\to f)}{2}\lceil\frac{D}{K}\rceil$.

We begin with the base case. Suppose that $d_*(v \to f) = 2$, such that each node $f' \in \mathcal{N}_v \setminus f$ is a leaf node. Note that the message $\hat{\mu}^t_{v\to f}$ is a function of $\hat{\mu}^{t-1}_{v\to f}$ and $\mu^t_{v\to f}$. Furthermore, the message $\mu^t_{v\to f}$ is a function of the messages $\mu^t_{f'\to v}$ for $f' \in \mathcal{N}_v \setminus f$. Since the nodes f' are leaf nodes, the messages $\mu^t_{f'\to v}$ are stable from $t = 1$. This implies that the message $\mu^t_{v\to f}$ is also stable from $t = 1$. This furthermore implies that $\hat{\mu}^t_{v\to f}$ will be stable from $t^*_{v\to f}$, where $t^*_{v\to f} \leq \lceil\frac{D}{K}\rceil = \frac{d_*(v\to f)}{2}\lceil\frac{D}{K}\rceil$. This is because it will take at most $\lceil\frac{D}{K}\rceil$ iterations for the update function $U(\cdot)$ to fill in $\hat{\mu}_{v\to f}$ with the D elements of $\mu_{v\to f}$, K elements at a time.

Now, assume that, for any edge $(\tilde{v}, \tilde{f})$, if $d_*(\tilde{v} \to \tilde{f}) \leq \hat{d}$ then $\hat{\mu}^t_{\tilde{v}\to\tilde{f}}$ is stable from iteration $t^*_{\tilde{v}\to\tilde{f}}$, where $t^*_{\tilde{v}\to\tilde{f}} \leq \frac{d_*(\tilde{v}\to\tilde{f})}{2}\lceil\frac{D}{K}\rceil$. Now consider an edge (v, f) where, for $f' \in \mathcal{N}_v$, we have either that f' is a leaf node or that $\mathbb{T}(v' \to f')$ is a subtree of the factor tree for $v' \in \mathcal{N}'_f \setminus v$. Furthermore, assume that the maximal depth of these subtrees is $\hat{d} = d_*(v \to f) - 2$.

Note that the message $\hat{\mu}^t_{v\to f}$ is a function of $\hat{\mu}^{t-1}_{v\to f}$ and $\mu^t_{v\to f}$. We also have that the message $\mu^t_{v\to f}$ is a function of the messages $\mu^t_{f'\to v}$ for $f' \in \mathcal{N}_v \setminus f$. Finally, we have that, for each $f' \in \mathcal{N}_v \setminus f$, the message $\mu^t_{f'\to v}$ is either constant, i.e., stable from $t = 1$, if f' is a leaf node, or it is a function of each $\hat{\mu}^{t-1}_{v'\to f'}$ for $v' \in \mathcal{N}'_f \setminus v$. However, from our induction hypothesis and the assumption on subtree depth, we have that each $\hat{\mu}^t_{v'\to f'}$ is stable from $t \leq \frac{d_*(v'\to f')}{2}\lceil\frac{D}{K}\rceil \leq \frac{\hat{d}}{2}\lceil\frac{D}{K}\rceil$. Of course, we also have that $1 \leq \frac{\hat{d}}{2}\lceil\frac{D}{K}\rceil$. This makes $\mu^t_{f'\to v}$ stable for $t \leq \frac{\hat{d}}{2}\lceil\frac{D}{K}\rceil + 1$. This implies that $\mu^t_{v\to f}$ is stable for $t \leq \frac{\hat{d}}{2}\lceil\frac{D}{K}\rceil + 1$. Finally, filling in the elements of $\hat{\mu}^t_{v\to f}$ will take at most $\lceil\frac{D}{K}\rceil$ iterations, beginning no later than iteration $\frac{\hat{d}}{2}\lceil\frac{D}{K}\rceil + 1$, which implies $\hat{\mu}^t_{v\to f}$ will be stable from $t \leq \frac{\hat{d}}{2}\lceil\frac{D}{K}\rceil + \lceil\frac{D}{K}\rceil = \frac{\hat{d}+2}{2}\lceil\frac{D}{K}\rceil = \frac{d_*(v\to f)}{2}\lceil\frac{D}{K}\rceil$. This proves, from the induction hypothesis, that for any edge $(\tilde{v}, \tilde{f})$, if $d_*(\tilde{v} \to \tilde{f}) \leq \hat{d}+2$ then $\hat{\mu}^t_{\tilde{v}\to\tilde{f}}$ is stable from iteration $t^*_{\tilde{v}\to\tilde{f}}$, where $t^*_{\tilde{v}\to\tilde{f}} \leq \frac{d_*(\tilde{v}\to\tilde{f})}{2}\lceil\frac{D}{K}\rceil$. This, together with the base case, prove that for any edge (v, f) in the finite factor tree, we have that $\hat{\mu}^t_{v\to f}$ is stable from iteration $t^*_{v\to f}$, where $t^*_{v\to f} \leq \frac{d_*(v\to f)}{2}\lceil\frac{D}{K}\rceil$. Therefore, every message $\hat{\mu}^t_{v\to f}$ will be stable from some $t^*_{v\to f} \leq \max_{(v,f)} \frac{d_*(v\to f)}{2}\lceil\frac{D}{K}\rceil = \frac{d}{2}\lceil\frac{D}{K}\rceil$. Therefore, K-Projected BP has reach a fixed

point no later than itertion $\frac{d}{2}\left\lceil \frac{D}{K} \right\rceil$. Since this fixed point must correspond with a fixed point of Sum-Product, then it must correspond with the unique fixed point of Sum-Product. This concludes the proof of Theorem 1.4.12. ■

1.5 Quantized Coded Belief Propagation: Zoom BP

We will now present a variation of belief propagation that places even greater emphasis on the communication overhead than the Projected BP algorithms developed in Section 1.4. Essentially, this algorithm, which we call Zoom Belief Propagation, is an adaptation of Projected BP that incorporates ideas from [15] for the coding of messages of real values into a finite alphabet.

1.5.1 The Zoom Belief Propagation Algorithm

The algorithm is similar to the Projected BP algorithms, 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 $[\mu_{v\to f} - \hat{\mu}_{v\to f}]_i$. To do so, we employ separate encoder/decoder pairs as described in [15] 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 $\mathcal{A} = \{i : i \in \mathcal{Z}, |i| \leq \bar{Q}\}$ for some integer $\bar{Q} \geq 1$. 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.

Algorithm 1.7: Zoom Belief Propagation, quantized coded messages.

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 $\hat{\mu}^0_{v\to f}(X_v) = \frac{1}{D}$ for each $(v,f) \in \mathcal{E}$;

Initialize $\mathcal{U}^0_{v\to f} = \mathcal{X}_v$;

Initialize $\theta^0_{f\to v}(X_v) = \hat{\mu}^{-1}_{v\to f}(X_v) = 0$ for each $(v,f) \in \mathcal{E}$;

Initialize $a^0_{v\to f}(X_v) = a_0$;

Initialize $t = 0$;

repeat

  $t \leftarrow t+1$;

  /* Update function to variable 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

      Enforce $\theta^t_{f\to v}(X_v) = \mathrm{Marginal}(f \to v)$;

    end

  end

  /* Update variable to function 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

  foreach $(v,f) \in \mathcal{E}$ **do**

    Choose $\mathcal{U}^t_{v\to f}$ where $\mathcal{U}^t_{v\to f} \subseteq \mathcal{X}_v$;

    Transmit $\mathcal{U}^t_{v\to f}$;

    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 $i \in \mathcal{U}^t_{v\to f}$;

  end

  foreach $(v,f) \in \mathcal{E}$ **do**

    $$\hat{\mu}^t_{v\to f}(i) = \begin{cases} \hat{\mu}^{t-1}_{v\to f}(i) + H(a^{t-1}_{v\to f}(i), q^t_{v\to f}(i)) & \text{if } i \in \mathcal{U}^t_{v\to f} \\ \hat{\mu}^{t-1}_{v\to f}(i) & \text{if } i \notin \mathcal{U}^t_{v\to f} \end{cases};$$

    $$a^t_{v\to f}(i) = \begin{cases} V(a^{t-1}_{v\to f}(i), q^t_{v\to f}(i)) & \text{if } i \in \mathcal{U}^t_{v\to f} \\ a^{t-1}_{v\to f}(i) & \text{if } i \notin \mathcal{U}^t_{v\to f} \end{cases};$$

  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 $f \in \mathcal{N}_v$

Specifically, for the encoder function, we have that

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

For the decoder function, we have that

$$H(a, q) \triangleq aq.$$

Finally, for the encoder/decoder state update function, we have that

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

where we have that $0 < Z_{\text{in}} < 1$ 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 1.7. The marginal computation at Line 12 is identical to the same computation in Projected BP. We now note that the computational complexity of Zoom BP is essentially the same as that of Projected BP. The only real computational difference is the small amount of computation involved 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 $\left|\mathcal{U}_{v\to f}^t\right|$ of the update subset using $\lceil \log_2(D) \rceil$ bits, followed by the elements of $\mathcal{U}_{v\to f}^t$ using $\left|\mathcal{U}_{v\to f}^t\right| \lceil \log_2(D) \rceil$ bits, followed by the symbols $q_{v\to f}^t(i)$ for $i \in \mathcal{U}_{v\to f}^t$, using $\left|\mathcal{U}_{v\to f}^t\right| \lceil \log_2(2\bar{Q} + 1) \rceil$ bits. This represents potentially significant savings in communication rate compared to the transmission of 32- or 64-bit floating point values.

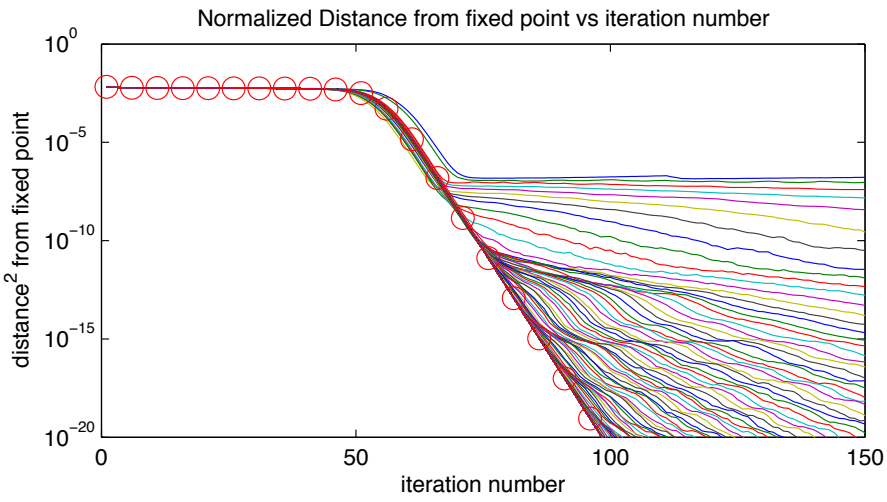


Figure 1.4: 10×10 square grid, $D = 4096$, Potts model. Normalized squared distance from the Sum-Product fixed point versus iteration number. Sum-Product is marked with circles. The other curves are K-Projections BP, where K=1 is the top curve, and larger values of K approaching the Sum-Product curve. Note that all instances of K-Projections and Sum-Product reach ∼1e-7 after about 70 iterations.

1.6 Simulations

In this section, we examine the performance of various belief propagation algorithms in a variety of scenarios. Specifically, we will be comparing Sum-Product BP, the Stochastic BP algorithms SBP1 and SBP2, K-Projected BP, and Zoom BP, all of which we have implemented in C++.

1.6.1 Convergence in a Simple Test Graph

The first set of experiments are with the simple square grid Potts model Markov random field (MRF) from [3]. In this graph, we have pairwise potential functions,

$$\psi_f(i,j) = \begin{cases} 1 & \text{if } i = j \\ \gamma & \text{if } i \neq j \end{cases}, \tag{1.12}$$

for each edge in the MRF grid, and single variable potentials,

$$\psi_f(i) = \begin{cases} 1 & \text{if } i = 1 \\ \mu + \sigma Z_{v,i} & \text{if } i \neq 1 \end{cases},$$

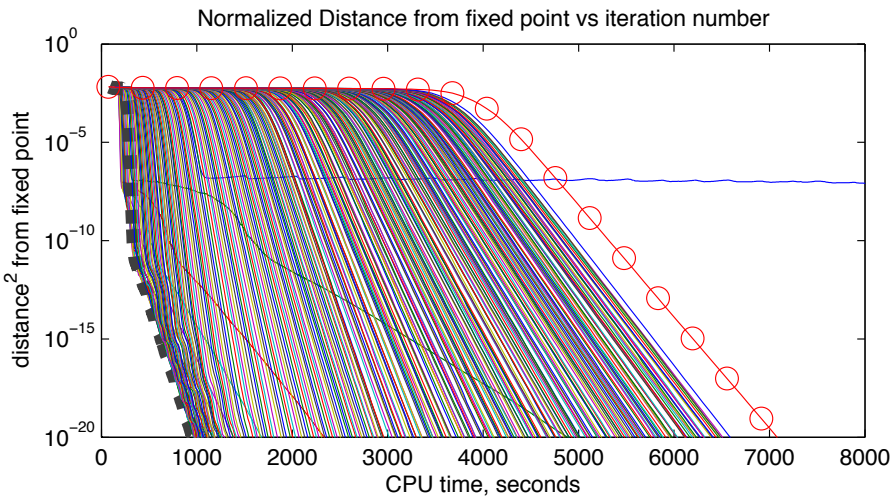


Figure 1.5: 10×10 square grid, $D = 4096$, Potts model. Normalized squared distance from the Sum-Product fixed point versus CPU time. Sum-Product is marked with circles. The other curves are K-Projections BP. For $K = 1$, convergence becomes quite slow after reaching $\sim$1e-7. For $K = \{6, 11, 16, 21\}$, the convergence rate per second CPU time improves. For K greater than approximately 65, the improvement in convergence per iteration no longer compensates the increase in computational complexity. The $K = 65$ curve is set in a thick dashed line.

connected to each MRF node, 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 this structure is not being exploited in the message updates in the algorithms.

Figures 1.4, 1.5, and 1.6 show some results of these experiments. In each of these graphs, we show results of the Sum-Product algorithm (marked by red circles), as well as K-Projected BP for $K = \{1, 6, 11, 16, 21, ...\}$. First, Figure 1.4 shows how the K-Projections BP algorithm converges to a fixed point in comparison to Sum-Product, as a function of iteration number. The vertical axis is the squared distance from the fixed point of the Sum-Product algorithm, normalized by the squared norm of the fixed point. In these experiments, we have taken the algorithm state to be the concatenation of all function to variable messages on all edges of the graph. (For K-Projections, the messages used are normalized versions of every $\theta^t_{f\to v}(\cdot)$.) What we see

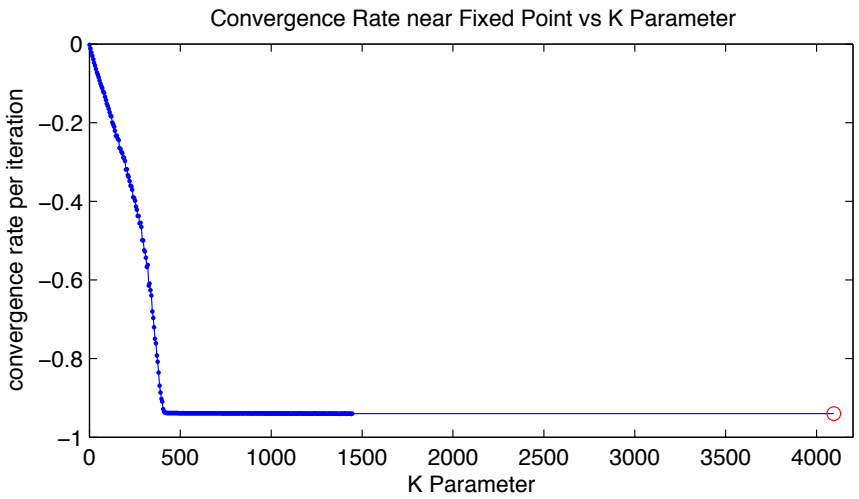


Figure 1.6: 10×10 square grid, $D = 4096$, Potts model. Convergence rate R of the normalized squared distance from the Sum-Product fixed point versus the parameter K of the K-Projections algorithm, such that $|\varepsilon_{t+1}|^2 \approx e^R |\varepsilon_t|^2$ near the fixed point. This is computed for $K = \{1, 6, 11, ..., 1446\}$ The rate of Sum-Product is marked with a circle, and corresponds with $K = 4096$.

is that after some number of iterations, the convergence per iteration of the K-Projections algorithm is slower than Sum-Product. This convergence is slowest for $K = 1$, and steadily improves as K is increased. However, it is quite interesting that all instances of K-Projections, even with $K = 1$, converge to a normalized squared distance of about 10^{-7} after about 70 iterations, which is essentially the same as Sum-Product. In fact, for many values of K, K-Projections reaches 10^{-7} sooner than Sum-Product.

In Figure 1.5, we show the same comparison between K-Projections and Sum-Product, with the same vertical axis, except that now the horizontal axis is in seconds of computation (wall) time. We see that for $K = 1$, convergence becomes quite slow after reaching ∼1e-7. For $K = \{6, 11, 16, 21\}$, the convergence rate per second CPU time improves. Hence, in this range the computational cost incurred per iteration by utilizing larger update subsets is more than compensated by the corresponding increase in convergence speed. For K greater than approximately 65, the improvement in convergence per iteration no longer compensates the increase in computational complexity. However, for essentially all instances of K-Projections in this simulation, convergence is faster than the standard Sum-Product update.

Finally, in Figure 1.6, we show how the convergence rate per iteration of

K-Projected BP in the vicinity of the belief propagation fixed point depends on the choice of parameter 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 $|\varepsilon_{t+1}|^2 \approx e^R|\varepsilon_t|^2$. Therefore, we hope for R to be as negative as possible. We see that up to $K \approx 250$, R decreases linearly as K increases, then the rate accelerates downward until it stops decreasing for $K > 410$. At this point, K-Projections has already achieved the convergence per iteration of Sum-Product BP, with a small fraction of the computational load.

1.6.2 Stereo Image Matching - Energy Minimization

The remainder of our simulations use the "Cones" stereo image pair from [30], as shown in Figures 1.7(a) and 1.7(b), and the "Tsukuba" stereo image pair from [31], as shown in Figures 1.7(c) and 1.7(d), in order to test the algorithms on a basic model for stereo image matching. We note that the model we use in these tests admit computational simplifications due 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 $(i - d, j)$. We also have pairwise potential functions composing a square grid Pott's model as in Equation (1.12), 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:

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

(a) "Cones" left source image.

(b) "Cones" right source image.

(c) "Tsukuba" left source image.

(d) "Tsukuba" right source image.

Figure 1.7: The "Cones" and "Tsukuba" stereo matching source images.

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 defined in Equation (1.1), and $\mathbf{x}$ is the vector of all hard decisions $x_{i,j}$.

We will now present the results of experiments where we evaluate Sum-Product BP and some of its 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 something different called the Bethe free energy [32, 33]. 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 (1.2). 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

$$\mathrm{P}_{X_v}(x_v) = \begin{cases} 1 & \text{if } x_v = \tilde{x}_v \\ 0 & \text{if } x_v \neq \tilde{x}_v \end{cases} . \tag{1.13}$$

Clearly, the decision rule applied to this would give

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

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 $\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \mathrm{P}_X(\mathbf{x})$.

In summary, this shows that if $\mathrm{P}_{X_v}(\cdot)$ is of the form given in Equation (1.13), 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 (1.13) 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 $\hat{\mathbf{x}}$.

In these experiments, we will be comparing the standard Sum-Product BP, K-Projected BP with $K = 1$, Zoom BP, Stochastic BP 0 (SBP0), and Stochastic BP 2 (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 Stochastic BP, note that we are not separately comparing with Stochastic BP 1, since the graphical model in which we are performing inference has function nodes of maximal degree equal to 2, which means that SBP1 is the same as SBP0. For the Cones data set, the disparities to be estimated have $D = 50$ possible values, whereas for the Tsukuba data set, the disparities to be estimated have $D = 17$ possible values. The square grid factor graph generated from the Cones images has dimensions 351×375, and that generated from the Tsukaba images has dimensions 367×288.

First, we have results on the Cones images in Figure 1.8. What we see is

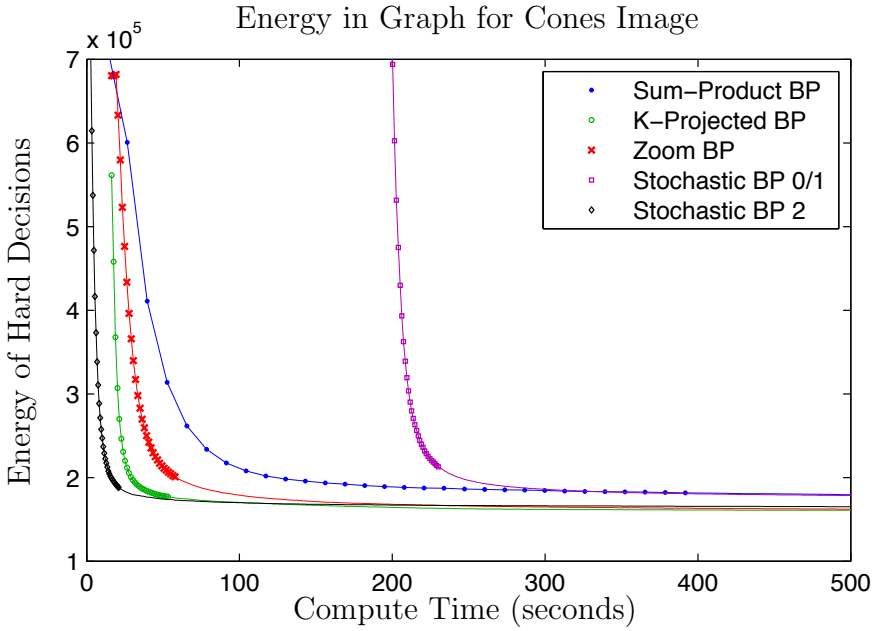


Figure 1.8: Cones Energy Graph. Markers indicate energy and compute time since the start of the initialization process for the first 30 iterations of each algorithm. Compute time includes any algorithm setup time and precomputations.

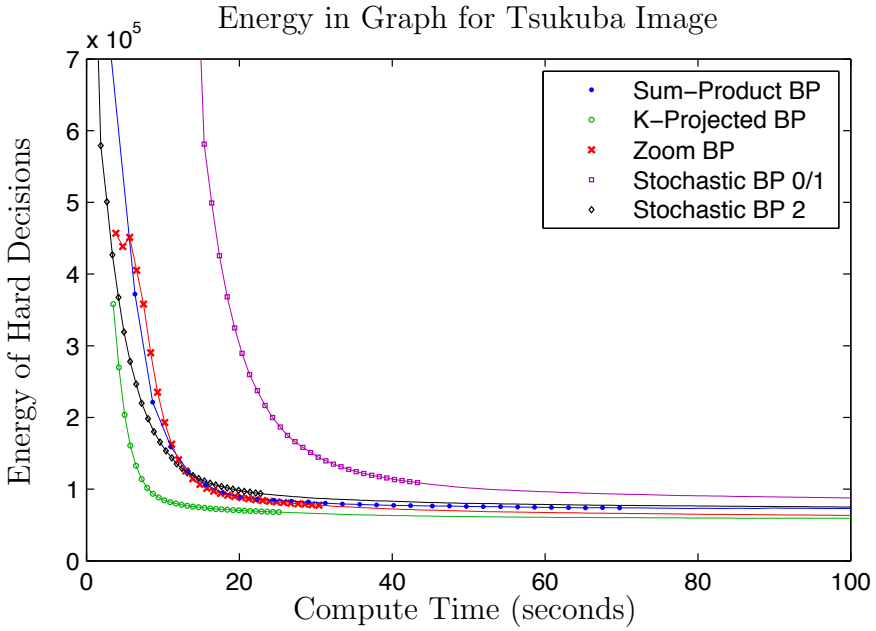
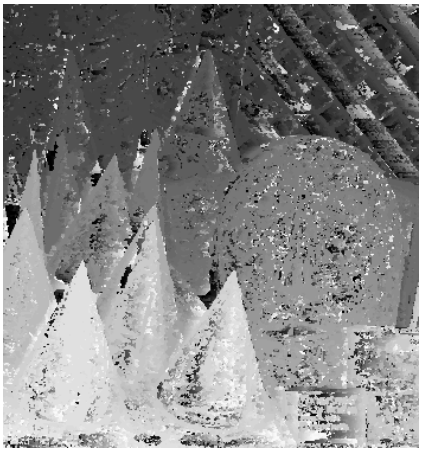


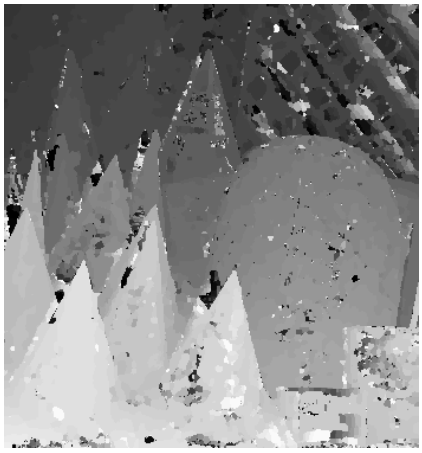
Figure 1.9: Tsukuba Energy Graph. Markers indicate energy and compute time since the start of the initialization process for the first 30 iterations of each algorithm. Compute time includes any algorithm setup time and precomputations.

that Sum-Product takes the largest steps in decreasing the energy, but the iterations each take much more compute time than the iterations of any other algorithm. The consequence is that in the first phase of convergence, after the initialization period, all of 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. Stochastic BP 0, on the other hand, takes a significant amount of time to precompute every $\beta_{f\to v}(X_w)$ and $\Gamma_{f\to v}(X_v, X_w)$. Of course, Stochastic BP 2 avoids this computation entirely, and suffers no comparative loss in convergence rate as a consequence. Stochastic BP 0 ends up converging to an energy level comparable to Sum-Product, which is higher than the level to which the other algorithms converge. Even Zoom BP outperforms Sum-Product BP in energy minimization, despite sending such a small amount of information in every message update.

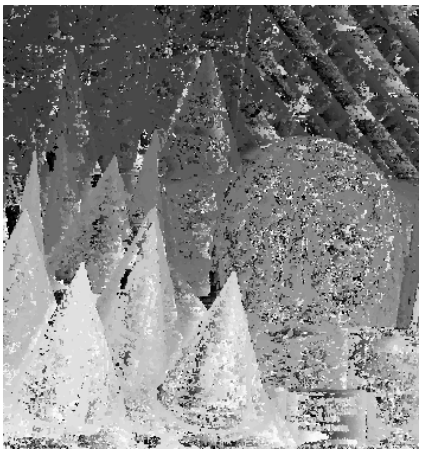
We present similar results for the Tsukaba images in Figure 1.9. Since D is smaller in this case, the per-iteration advantage of each algorithm over Sum-Product is less significant. We see that K-Projected BP converges faster and to a lower energy level than any other algorithm. Zoom BP is comparable to Sum-Product in convergence speed, though it approaches a slightly lower value than Sum-Product. With Stochastic BP 0, we see that 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 Stochastic BP 0 and Stochastic BP 2 seem to 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 graphs 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 Stochastic BP 0 provides no benefit over Stochastic BP 2 while increasing the setup time severely. Finally, the per-second convergence rate, after initialization, of Stochastic BP (either 0 or 2) has the potential to be



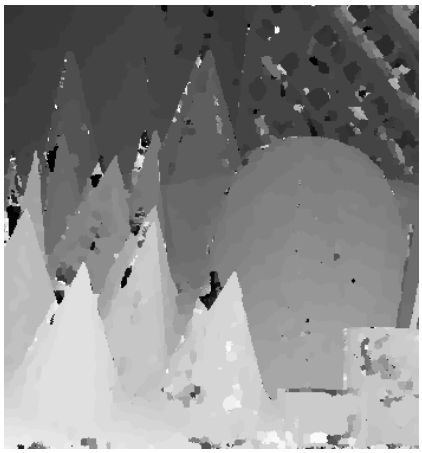
(a) Sum-Product BP. Iter=2.



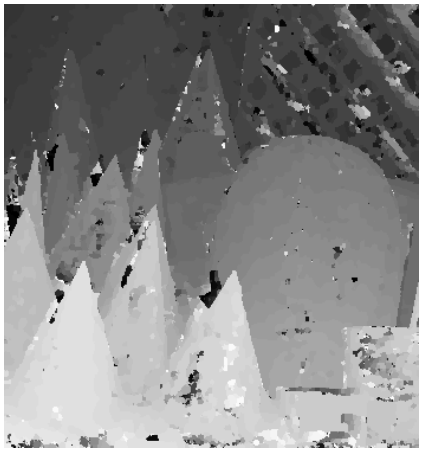
(b) K-Projected BP, $K = 1$. Iter=9.



(c) Zoom BP, $K = 1$. Iter=8.

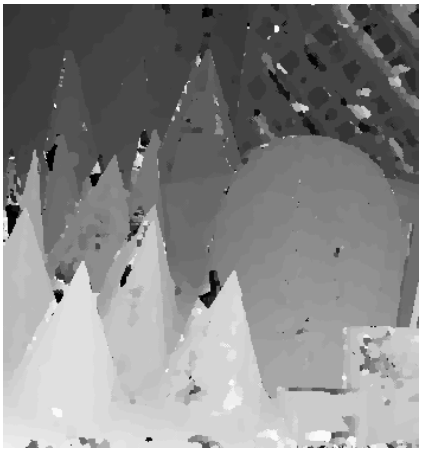


(d) Stochastic BP 2. Iter=38.

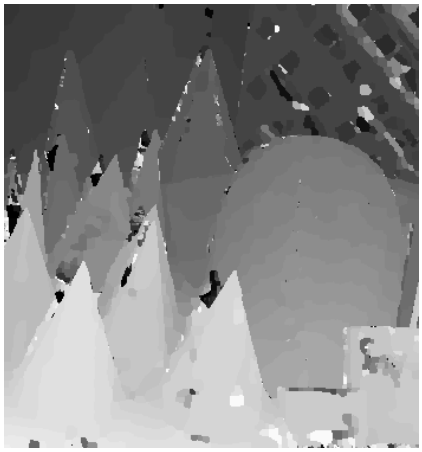


(e) Stochastic BP 0/1. Iter=38.

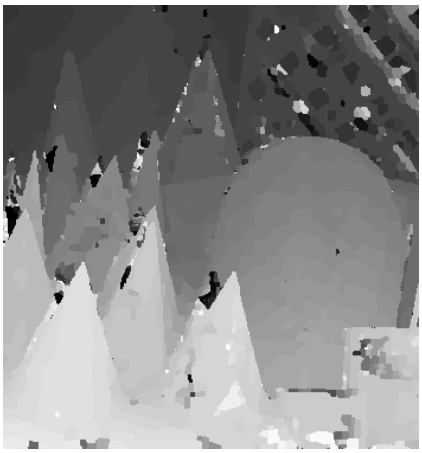
Figure 1.10: Comparison of results given same computation time, except for SBP0. Computation time is 26.6 seconds, equal to 2 iterations of Sum-Product. SBP0 did not complete its precomputation by this time. Also included in (e) is the SBP0 result after the same number of iterations as SBP2, at 238.5 seconds.



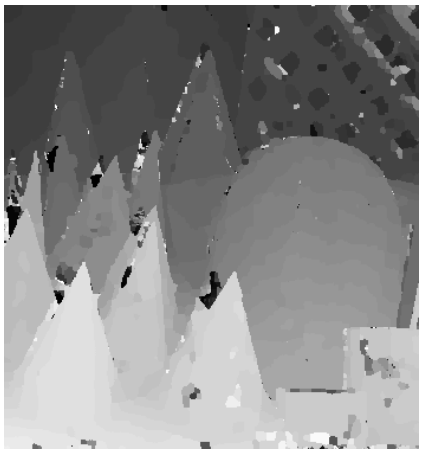
(a) Sum-Product BP. Iter=10.



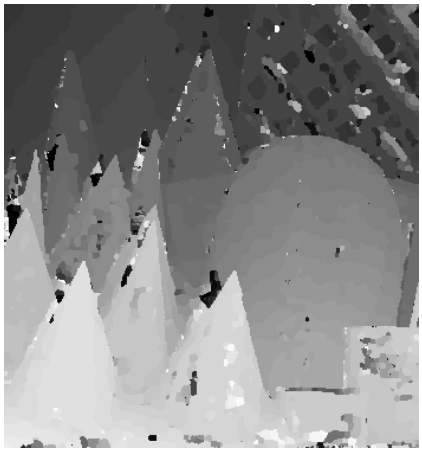
(b) K-Projected BP, $K = 1$. Iter=90.



(c) Zoom BP, $K = 1$. Iter=80.

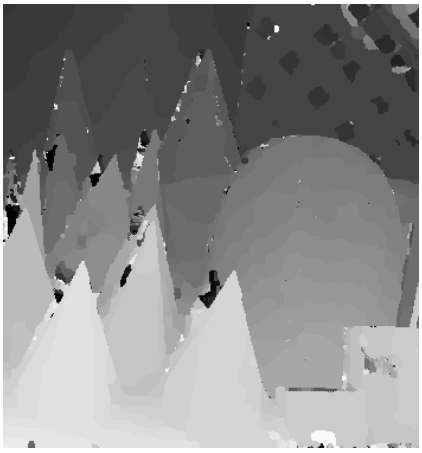


(d) Stochastic BP 2. Iter=194.

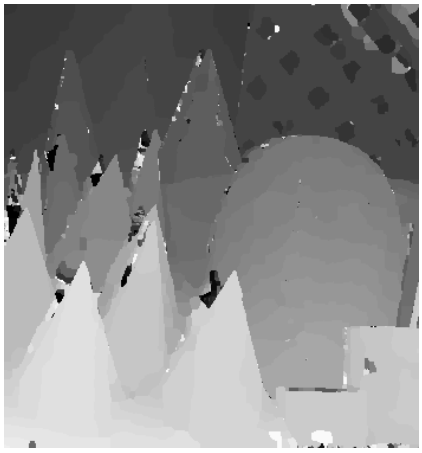


(e) Stochastic BP 0/1. Iter=194.

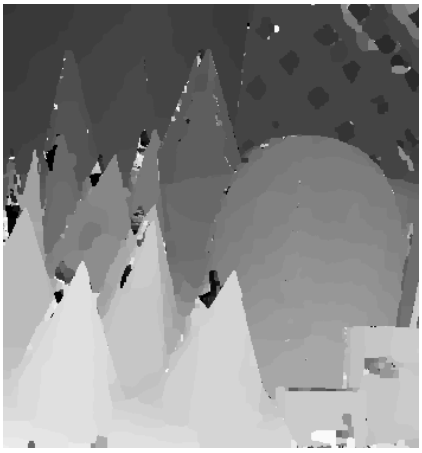
Figure 1.11: Comparison of results given same computation time, except for SBP0. Computation time is 130 seconds, equal to 10 iterations of Sum-Product. SBP0 did not complete its precomputation by this time. Also included in (e) is the SBP0 result after the same number of iterations as SBP2, at 402.8 seconds.



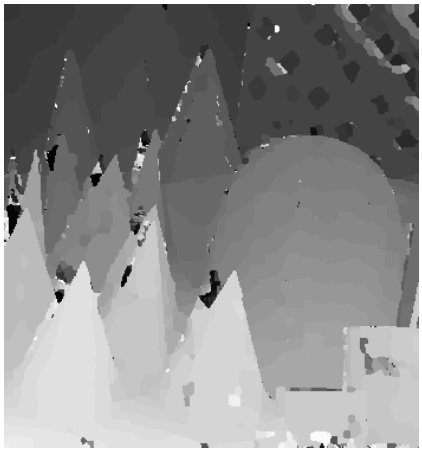
(a) Sum-Product BP.



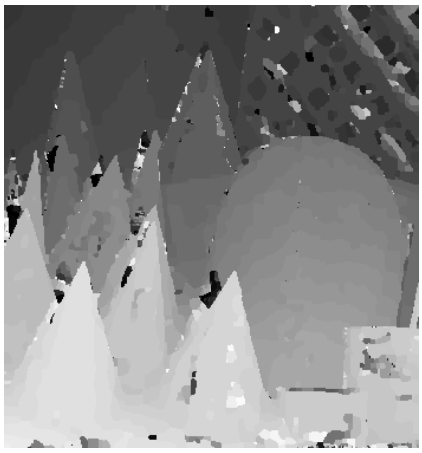
(b) K-Projected BP, $K = 1$.



(c) Zoom BP, $K = 1$.



(d) Stochastic BP 2.



(e) Stochastic BP 0/1.

Figure 1.12: Comparison of results after 1000 iterations.

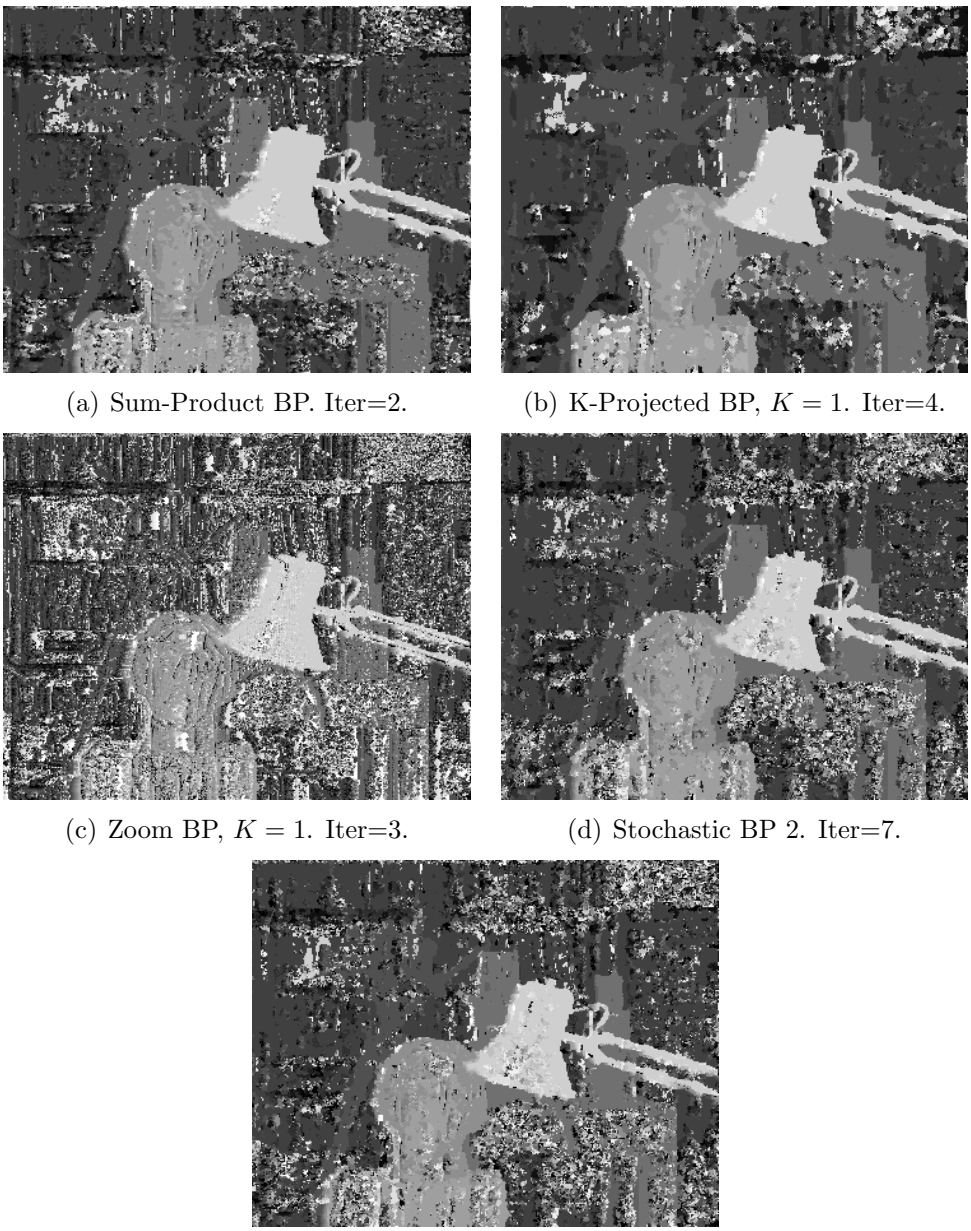
better than any of the other algorithms, but this is subject to the particular graph, as will see in the following.

1.6.3 Stereo Image Matching - Qualitative Results

We will now present some qualitative stereo matching results. Specifically we will 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. First, in Figure 1.10, we have the disparity maps that result after 26.6 seconds of compute time on the Cones data for the algorithms Sum-Product BP, K-Projected BP, Zoom BP, and Stochastic BP 2. This time is equivalent to the time it takes to perform 2 iterations of Sum-Product BP. In order of increasing apparent quality of results, we have Zoom BP, then Sum-Product BP appears slightly better, followed by K-Projected BP, which looks much better, and finally Stochastic BP 2. This is roughly in line with the algorithm performance indicated by the energy of the solutions, shown in Figure 1.8. Although Stochastic BP 0 had not completed its precomputation by this point, we show the disparity map at iteration 38 in Figure 1.10(e), which is the number of iterations of Stochastic BP 2 that had completed by 26.6 seconds. We see that the results of Stochastic BP 0 are comparable or slightly worse than the results of Stochastic BP 2.

In Figure 1.11, we have the disparity maps that result after 130 seconds of compute time on the same algorithms. This time is equivalent to the time it takes to perform 10 iterations of Sum-Product BP. In order of increasing apparent quality of results, we have Sum-Product BP, then Zoom BP is qualitatively almost the same, followed by K-Projected BP and Stochastic BP 2, which appear to have essentially the same quality. Finally, although Stochastic BP 0 had not completed its precomputation by this point, we show the disparity map at iteration 194 in Figure 1.11(e), which is the number of iterations of Stochastic BP 2 that had completed by 130 seconds. We see that the results of Stochastic BP 0 actually appear inferior to the results of Stochastic BP 2, especially in the regions of the fence at the back, and the mug in the front.

Finally, in Figure 1.12, we have the disparity maps that result after 1000 iterations of each of the 5 algorithms. Each of Sum-Product BP, K-Projected



(a) Sum-Product BP. Iter=2.

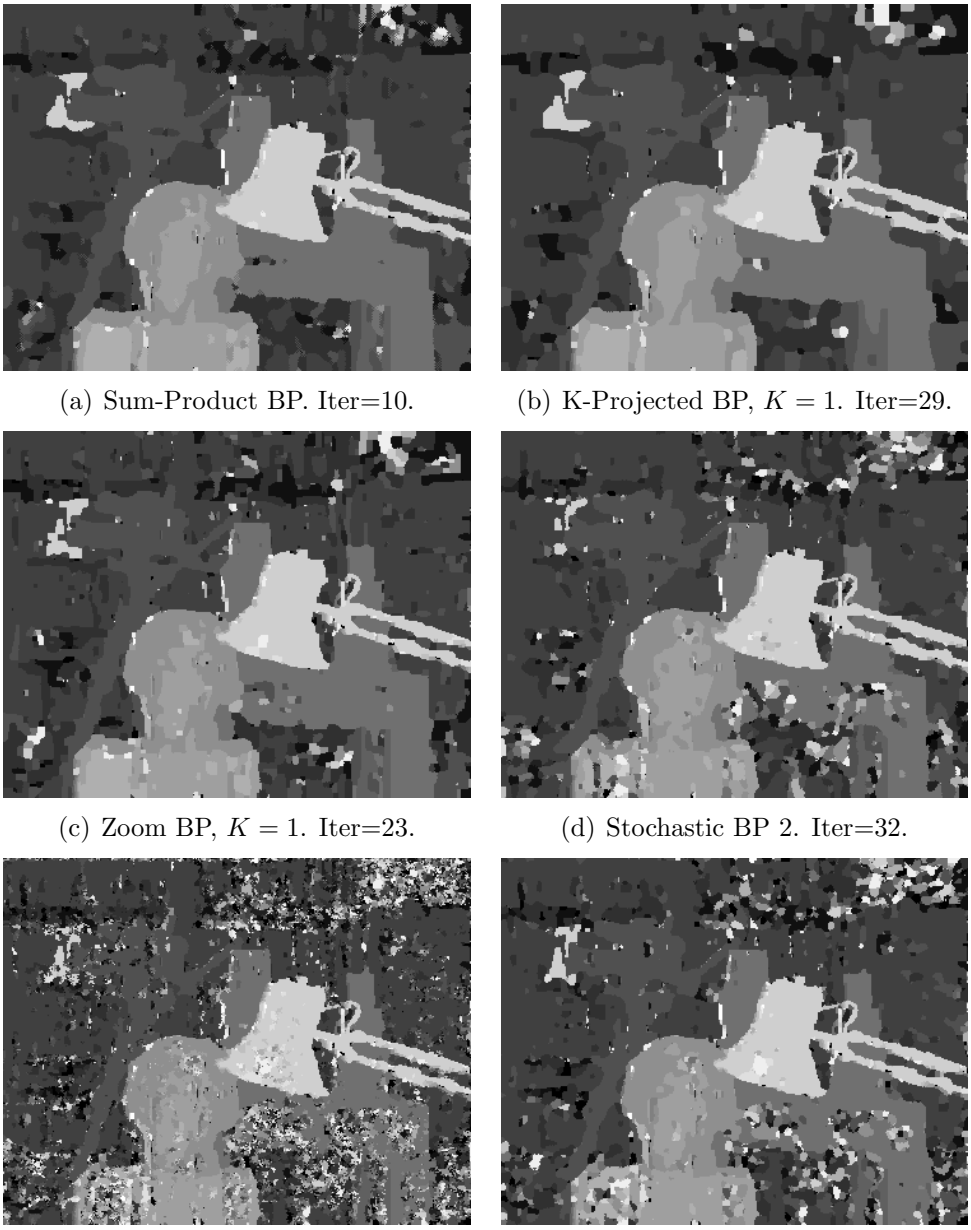
(b) K-Projected BP, $K = 1$. Iter=4.

(c) Zoom BP, $K = 1$. Iter=3.

(d) Stochastic BP 2. Iter=7.

(e) Stochastic BP 0/1. Iter=7.

Figure 1.13: Comparison of results given same computation time. Computation time is 6.3 seconds, equal to 2 iterations of Sum-Product. SBP0 did not complete its precomputation by this time. Also included in (e) is the SBP0 result after the same number of iterations as SBP2, at 20.3 seconds.



(a) Sum-Product BP. Iter=10.

(b) K-Projected BP, $K = 1$. Iter=29.

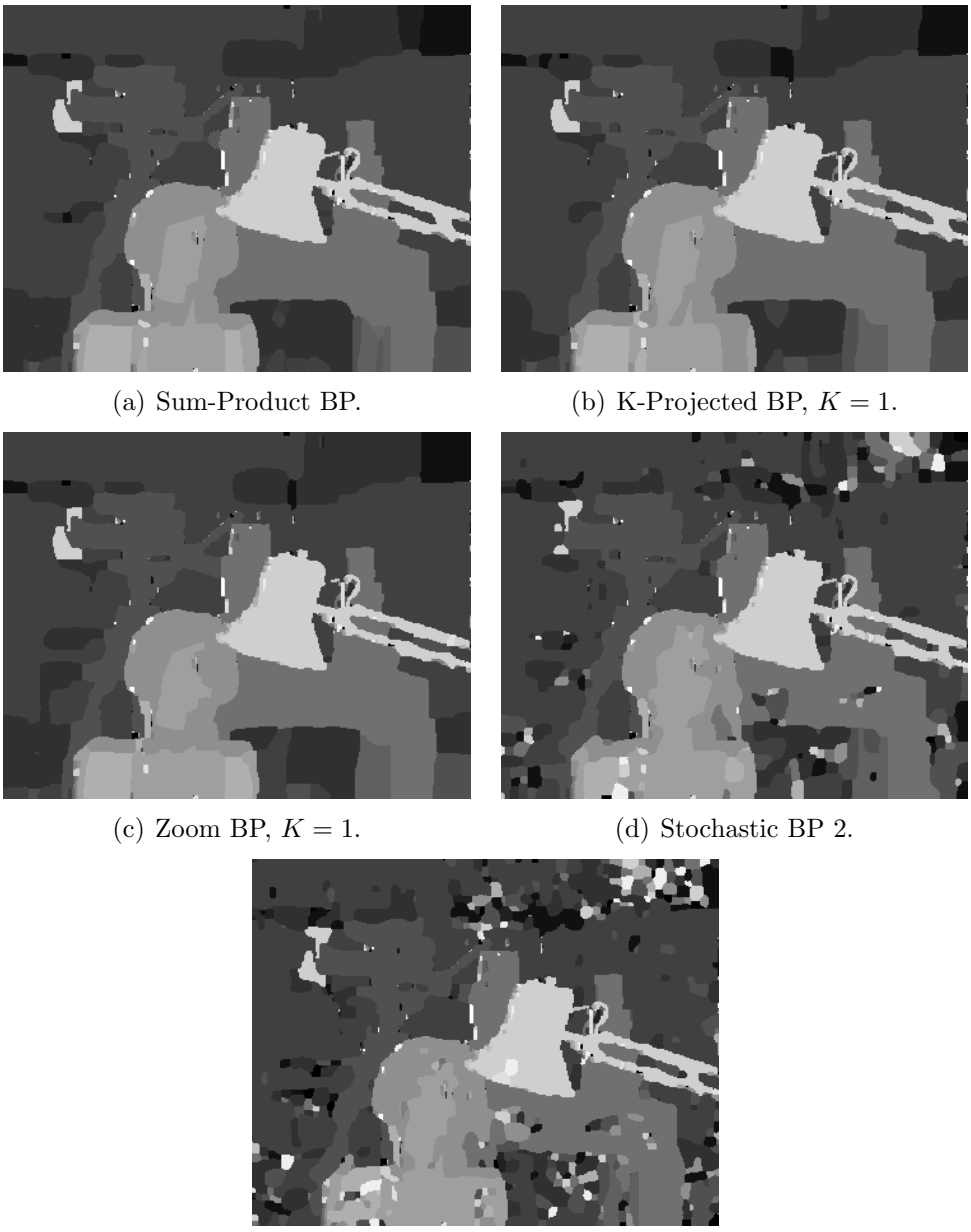
(c) Zoom BP, $K = 1$. Iter=23.

(d) Stochastic BP 2. Iter=32.

(e) Stochastic BP 0/1. Iter=11.

(f) Stochastic BP 0/1. Iter=32.

Figure 1.14: Comparison of results given same computation time. Computation time is 24.6 seconds, equal to 10 iterations of Sum-Product. Also included in (f) is the SBP0 result after the same number of iterations as SBP2, at 45.3 seconds.



(a) Sum-Product BP.

(b) K-Projected BP, $K = 1$.

(c) Zoom BP, $K = 1$.

(d) Stochastic BP 2.

(e) Stochastic BP 0/1.

Figure 1.15: Comparison of results after 1000 iterations.

BP, Zoom BP, and Stochastic BP 2 have results that are essentially the same, whereas Stochastic BP 0 appears to give inferior results. Again, the regions of concern are the fence at the back, and the mug in the front.

The disparity map results for the Cones images show that the alternatives to Sum-Product BP can give good results in a practical application sooner than Sum-Product. With the Tsukuba images, however, we will see the same, but we will also gain more intuition about the strengths or weaknesses of the Sum-Product alternatives.

In Figure 1.13, we have the disparity maps that result after 6.3 seconds of compute time for Sum-Product BP, K-Projected BP, Zoom BP, and Stochastic BP 2, the amount of time it took for 2 iterations of Sum-Product BP. We make some of the same conclusions as with the Cones data: Even with a smaller variable cardinality of $D = 17$, we can see by the result given by K-Projected BP that it is possible to improve on the efficiency of Sum-Product BP. We also see that there is essentially no advantage in Stochastic BP 0 over Stochastic BP 2, only the disadvantage of the large precompute time, which is evident because the results of iteration 7 of Stochastic BP 0 are comparable to those of iteration 7 of Stochastic BP 2, but the the results from SBP0 are achieved after significantly more computation time.

Figure 1.14, 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 1.14(d), 1.14(e) and 1.14(f), 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 shallow decreasing energy plots in Figure 1.9. 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 Figure 1.15, where all of the algorithms have run for 1000 iterations. Even after so many iterations, Stochastic BP 0 and Stochastic BP 2 are both unable to smooth out the homogeneous regions to a single disparity estimate, whereas Sum-Product, K-Projected BP, and

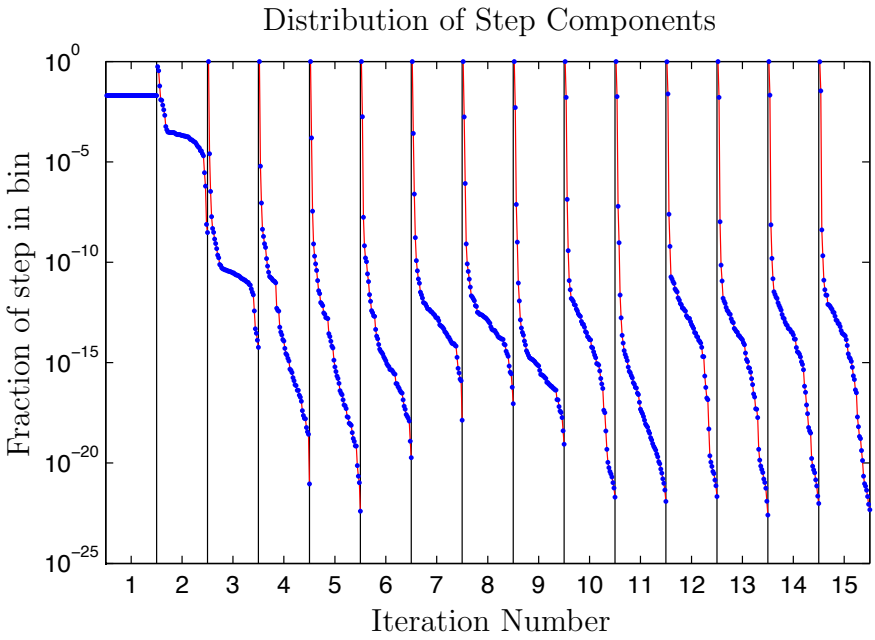


Figure 1.16: Fraction of the squared Euclidean norm of the step size $\tilde{\delta}^t(k)$ for the first 15 iterations of Sum-Product.

Zoom BP have all been able to accomplish this smoothing with far fewer iterations. This effect is much less apparent with the Cones images, because 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.

1.6.4 Experimental Step Sizes for Fixed Subset Size

In our final experiment, we explore how far the K-Projected BP update would be from a full Sum-Product BP update as a function of the subset size parameter K. Figure 1.16 gives an indication of how sufficient a given value of K is for approaching the full Sum-Product BP update. This experiment uses the graph defined for the Cones stereo images. Specifically, for $t \geq 1$, define

$$\delta^t(k) = \sum_{(v,f)} \left[\text{sort}_{\searrow}\left(\mu^t_{v\to f} - \mu^{t-1}_{v\to f}\right)\right]^2_k,$$

where $\text{sort}_{\searrow}(\cdot)$ sorts the elements of the argument in descending order. Note that $\delta^t(k)$ is decreasing in k. Then, for example, we have that

$$\left\|\mathcal{M}^t_{\mathcal{V}\to\mathcal{F}} - \mathcal{M}^{t-1}_{\mathcal{V}\to\mathcal{F}}\right\|_2^2 = \sum_k \delta^t(k),$$

where $\mathcal{M}^t_{\mathcal{V}\to\mathcal{F}}$ is the global state of Sum-Product at time t. Now, let's define

$${}_K\widehat{\mathcal{M}}^t_{\mathcal{V}\to\mathcal{F}} = U_K(G(F(\mathcal{M}^{t-1}_{\mathcal{V}\to\mathcal{F}})), \mathcal{M}^{t-1}_{\mathcal{V}\to\mathcal{F}}),$$

which is the result of applying an iteration of K-Projected BP with parameter K to the Sum-Product state $\mathcal{M}^{t-1}_{\mathcal{V}\to\mathcal{F}}$. Then we have that

$$\left\|{}_K\widehat{\mathcal{M}}^t_{\mathcal{V}\to\mathcal{F}} - \mathcal{M}^{t-1}_{\mathcal{V}\to\mathcal{F}}\right\|_2^2 = \sum_{k\le K} \delta^t(k),$$

as well as

$$\left\|{}_K\widehat{\mathcal{M}}^t_{\mathcal{V}\to\mathcal{F}} - \mathcal{M}^t_{\mathcal{V}\to\mathcal{F}}\right\|_2^2 = \sum_{k>K} \delta^t(k).$$

Therefore, if $\delta^t(k)$ is small $k > K$, then we know that the K-Projected BP update must be close to the full Sum-Product BP update. We see this in Figure 1.16, where we plot

$$\tilde{\delta}^t(k) \triangleq \frac{\delta^t(k)}{\sum_i \delta^t(i)}$$

for iterations 1 through 15 of Sum-Product. In fact, it is clear that $K = 1$ is sufficient for K-Projected BP to be quite close to a Sum-Product BP iteration, since we see that, after the first iteration, $\tilde{\delta}^t(1)$ is very close to 1, but $\tilde{\delta}^t(k)$ is quite small for $k > 1$.

1.7 Conclusion

Motivated in part by the strict low power requirements of distributed sensor networks, in this chapter we have considered alternatives to Sum-Product belief propagation, with special consideration for increased efficiency of both computation and communication. We began our development with a generalization and simplification of the Stochastic BP algorithm from [3]. We have

also proposed Projected belief propagation algorithms that provide gains in computational and communications efficiency, while avoiding the slow convergence rate drawback of Stochastic BP. We provide theoretical results proving that Projected BP converges exponentially quickly under certain conditions in loopy factor graphs, and that it converges in a factor tree to the unique Sum-Product fixed point in a finite number of iterations. We have also presented Zoom BP, which has even greater communications efficiency. Finally, we have given a number of experimental results demonstrating the strengths and weaknesses of the various algorithms in this Sum-Product family.

We have primarily remained focused on the distinct innovations developed for belief propagation. This includes the means of simplifying and generalizing Stochastic BP, the method for reducing message size and the corresponding reduction in computational complexity of the updates in Projected BP, and the method for utilizing discrete channels used by Zoom BP. However, it is certainly possible to mix these methods with other known methods, or come up with different ways of using our methods. For example, the dynamic coding and decoding schemes could be used selectively on a subset of edges and with varying levels of quantization in a distributed network where certain node links have higher bandwidth than others. Some edges could use the coding and decoding mechanisms, while others use Stochastic BP updates or Projected BP updates. These methods could also be mixed with alternative message passing schedules. For example, combining the methods of K-Projected BP with an update schedule like Residual BP [2] could lead to incredible increases in the performance of belief propagation in large loopy graphs with high variable cardinality.

It would be interesting to see the results of mixing methods in various ways for different applications. It would also be interesting to examine some other applications where the benefits of the methods might become much more evident. For example, image denoising and the estimation of optical flow in images are applications that potentially have high variable cardinalities in the hundreds or thousands. Stereo image matching in high resolution images would also have higher cardinality disparity variables.

REFERENCES

[1] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, February 2001.

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

[3] N. Noorshams and M. Wainwright, "Stochastic belief propagation: A low-complexity alternative to the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 59, no. 4, pp. 1981–2000, April 2013.

[4] M. Çetin, L. Chen, J. W. Fisher III, A. T. Ihler, R. L. Moses, M. J. Wainwright, and A. S. Willsky, "Distributed fusion in sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 42–55, July 2006.

[5] L. Varshney, "Performance of LDPC codes under faulty iterative decoding," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4427–4444, July 2011.

[6] S. M. S. Tabatabaei Yazdi, H. Cho, and L. Dolecek, "Gallager B decoder on noisy hardware," *IEEE Transactions on Communications*, vol. 61, no. 5, pp. 1660–1673, May 2013.

[7] J. Choi, E. P. Kim, R. A. Rutenbar, and N. R. Shanbhag, "Error resilient MRF message passing architecture for stereo matching," in *IEEE Workshop on Signal Processing Systems*, 2013.

[8] A. T. Ihler, J. W. Fisher III, and A. S. Willsky, "Loopy belief propagation: Convergence and effects of message errors," *Journal of Machine Learning Research*, vol. 6, pp. 905–936, May 2005.

[9] A. Dimakis, S. Kar, J. Moura, M. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, November 2010.

[10] A. Kashyap, T. Başar, and R. Srikant, "Quantized consensus," *Automatica*, vol. 43, no. 7, p. 11921203, 2007.

[11] J. Lavaei and R. M. Murray, "On quantized consensus by means of gossip algorithm - part I: Convergence proof," in *Proceedings of the American Control Conference*, June 2009, p. 394401.

[12] J. Lavaei and R. M. Murray, "On quantized consensus by means of gossip algorithm - part II: Convergence time," in *Proceedings of the American Control Conference*, June 2009, p. 29582965.

[13] F. Benezit, P. Thiran, and M. Vetterli, "Interval consensus: From quantized gossip to voting," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2009, pp. 3661–3664.

[14] R. Carli, F. Fagnani, P. Frasca, and S. Zampieri, "Gossip consensus algorithms via quantized communication," *Automatica*, vol. 46, pp. 70–80, 2010.

[15] R. Carli, F. Bullo, and S. Zampieri, "Quantized average consensus via dynamic coding/decoding schemes," *International Journal of Robust and Nonlinear Control*, vol. 20, pp. 156–175, 2010.

[16] T. C. Aysal, M. J. Coates, and M. G. Rabbat, "Distributed average consensus with dithered quantization," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4905–4918, October 2008.

[17] T. C. Aysal, M. Coates, and M. Rabbat, "Distributed average consensus using probabilistic quantization," in *Proceedings of the 14th IEEE Workshop on Statistical Signal Processing*, August 2007, pp. 640–644.

[18] R. Carli, G. Como, P. Frasca, and F. Garin, "Distributed averaging on digital noisy networks," in *Proceedings of the Information Theory and Applications Workshop (ITA)*, February 2011, pp. 1–9.

[19] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks: Quantized data and random link failures," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1383–1400, March 2010.

[20] V. Saligrama and D. A. Castanon, "Reliable distributed estimation with intermittent communications," in *Proceedings of the 45th IEEE Conference on Decision and Control*, December 2006, pp. 6763–6768.

[21] S. Patterson and B. Bamieh, "Distributed consensus with link failures as a structured stochastic uncertainty problem," in *Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computation*, 2008, pp. 623–627.

[22] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise," *IEEE Transactions on Signal Processing*, vol. 57, no. 1, pp. 355–369, January 2009.

[23] R. Carli, G. Como, P. Frasca, and F. Garin, "Distributed averaging on digital erasure networks," *Automatica*, vol. 47, pp. 115–121, 2011.

[24] C. C. Moallemi and B. V. Roy, "Consensus propagation," *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 4753–4766, November 2006.

[25] H. Kfir and I. Kanter, "Parallel versus sequential updating for belief propagation decoding," *Physica A*, vol. 330, pp. 259–270, November 2003.

[26] J. Goldberger and H. Kfir, "Serial schedules for belief-propagation: Analysis of convergence time," *IEEE Transactions on Information Theory*, vol. 54, no. 3, pp. 1316–1319, March 2008.

[27] C. Sutton and A. McCallum, "Improved dynamic schedules for belief propagation," in *Proceedings of the 23nd Conference on Uncertainty in Artificial Intelligence*, 2007.

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

[29] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Francisco, CA: Morgan Kaufman, 1988.

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

[31] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1/2/3, pp. 7–42, April-June 2002.

[32] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," in *Advances in Neural Information Processing Systems*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., vol. 13. Cambridge, MA: MIT Press, 2001.

[33] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2282–2312, July 2005.