© 2015 Andrew Joseph Bean

MESSAGE PASSING ALGORITHMS – METHODS AND APPLICATIONS

BY

ANDREW JOSEPH BEAN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Doctoral Committee:

Professor Andrew Singer, Chair
Assistant Professor Pulkit Grover, Carnegie Melon University
Assistant Professor Maxim Raginsky
Professor Naresh Shanbhag

Abstract

Algorithms on graphs are used extensively in many applications and research areas. Such applications include machine learning, artificial intelligence, communications, image processing, state tracking, sensor networks, sensor fusion, distributed cooperative estimation, and distributed computation. Among the types of algorithms that employ some kind of message passing over the connections in a graph, the work in this dissertation will consider belief propagation and gossip consensus algorithms.

We begin by considering the marginalization problem on factor graphs, which is often solved or approximated with Sum-Product belief propagation (BP) over the edges of the factor graph. For the case of sensor networks, where the conservation of energy is of critical importance and communication overhead can quickly drain this valuable resource, we present techniques for specifically addressing the needs of this low power scenario. We create a number of alternatives to Sum-Product BP. The first of these is a generalization of Stochastic BP with reduced setup time. We then present Projected BP, where a subset of elements from each message is transmitted between nodes, and computational savings are realized in proportion to the reduction in size of the transmitted messages. Zoom BP is a derivative of Projected BP that focuses particularly on utilizing low bandwidth discrete channels. We give the results of experiments that show the practical advantages of our alternatives to Sum-Product BP.

We then proceed with an application of Sum-Product BP in sequential investment. We combine various insights from universal portfolios research in order to construct more sophisticated algorithms that take into account transaction costs. In particular, we use the insights of Blum and Kalai's transaction costs algorithm to take these costs into account in Cover and Ordentlich's side information portfolio and Kozat and Singer's switching portfolio. This involves carefully designing a set of causal portfolio strategies and

computing a convex combination of these according to a carefully designed distribution. Universal (sublinear regret) performance bounds for each of these portfolios show that the algorithms asymptotically achieve the wealth of the best strategy from the corresponding portfolio strategy set, to first order in the exponent. The Sum-Product algorithm on factor graph representations of the universal investment algorithms provides computationally tractable approximations to the investment strategies. Finally, we present results of simulations of our algorithms and compare them to other portfolios.

We then turn our attention to gossip consensus and distributed estimation algorithms. Specifically, we consider the problem of estimating the parameters in a model of an agent's observations when it is known that the population as a whole is partitioned into a number of subpopulations, each of which has model parameters that are common among the member agents. We develop a method for determining the beneficial communication links in the network, which involves maintaining non-cooperative parameter estimates at each agent, and the distance of this estimate is compared with those of the neighbors to determine time-varying connectivity. We also study the expected squared estimation error of our algorithm, showing that estimates are asymptotically as good as centralized estimation, and we study the short term error convergence behavior.

Finally, we examine the metrics used to guide the design of data converters in the setting of digital communications. The usual analog to digital converters (ADC) performance metrics—effective number of bits (ENOB), total harmonic distortion (THD), signal to noise and distortion ratio (SNDR), and spurious free dynamic range (SFDR)—are all focused on the faithful reproduction of observed waveforms, which is not of fundamental concern if the data converter is to be used in a digital communications system. Therefore, we propose other information-centric rather than waveform-centric metrics that are better aligned with the goal of communications. We provide computational methods for calculating the values of these metrics, some of which are derived from Sum-Product BP or related algorithms. We also propose Statistics Gathering Converters (SGCs), which represent a change in perspective on data conversion for communications applications away from signal representation and towards the collection of relevant statistics for the purposes of decision making and detection. We show how to develop algorithms for the detection of transmitted data when the transmitted signal is received by

an SGC. Finally, we provide evidence for the benefits of using system-level metrics and statistics gathering converters in communications applications.

For Sophia

Acknowledgments

I would like to express my appreciation to Professor Andrew Singer, first of all for his support and guidance. I also want to thank him for the opportunity to present my research in so many incredible places all over the world. Even when I was struggling to maintain interest in a topic, the encouragement to submit at numerous international conferences provided much needed motivation to keep pushing forward. Finally, I especially want to thank him for his willingness to allow me to search for the research I could be passionate about, for his considerable patience as I dabbled in a number of different topics throughout this process, and for his understanding of my struggles.

I am grateful for many discussions, both academic in nature and not, with many people throughout the years. This includes my office mates Thomas Riedl and Peter Kairouz; past roommates John Kolinski, Praveen Bommannavar, and Michael Cason; and fellow ECE students Zuofu Cheng, Sundeep Kartan, Roger Serwy, Daniel Barron, Alex Duda, and Jon Weissman.

I am also grateful for the many people with whom I have been able to share my passion for rock climbing, and who have helped me find a way to get away from research a bit and reboot my mind. This includes Ricardo Mejia-Alvarez, Juan Saenz, James Fran, Rich Weston, Rob Werner, Praveen Bommannavar, Tzlil Perahia, Georg Zeitler, Andrew Maginniss, Kevin Sierzega, Mirella Bajric, Anthony Ho, Jon Weissman, Michael Cason, Michael Schubert, and many others.

Finally, the importance of my family to helping me achieve my academic goals cannot be overstated. I am most indebted to my parents Vicky and Kevin Bean, and my brother Jonathan Bean, for years of support, encouragement, and confidence in my potential. I similarly have a most heartfelt gratitude toward Flora Zhou for these same things, as well as for her understanding of my experiences as I have gone through the highs and lows of my graduate research, especially in the time leading up to the completion of

this dissertation. I truly appreciate all of the effort she has put in to help me make it through the process as smoothly as possible. Finally, I want to thank my daughter Sophia Bean for helping me to appreciate every moment and for being my motivation to plan for a bright future ahead of us.

Portions of this research were performed under an appointment to the U.S. Department of Homeland Security (DHS) Scholarship and Fellowship Program, administered by the Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy (DOE) and DHS. ORISE is managed by Oak Ridge Associated Universities (ORAU) under DOE contract number DE-AC05-06OR23100. All opinions expressed in this dissertation are the author's and do not necessarily reflect the policies and views of DHS, DOE, or ORAU/ORISE. This work was also supported by the department of the Navy, Office of Naval Research, under grants ONR MURI N00014-07-1-0738 and ONR N00014-07-1-0311 and by the National Science Foundation under grant NSF CCF 07-29092. Further support came from the Gigascale System Research Center (GSRC), one of five research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation program. Finally, we have received support from Systems on Nanoscale Information fabriCs (SONIC), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Algorithms on graphs are used extensively in many applications and research areas. Such applications include machine learning, artificial intelligence, communications, image processing, state tracking, sensor networks, sensor fusion, distributed cooperative estimation, and distributed computation. Among the types of algorithms that employ some kind of message passing over the connections in a graph, the work in this dissertation will consider belief propagation and gossip consensus algorithms.

In belief propagation (BP), typically we have a graphical representation of the joint probability distribution relating a number of variables to one another. The goal of the belief propagation algorithm might be, for example, to find the most likely state of all of the variables or to approximate marginal probability distributions of subsets of variables. The canonical problem of gossip consensus algorithms, on the other hand, is simply to compute the average of a number of observations in a distributed fashion in a network. Examples where this may be useful are distributed data fusion in a sensor network, or distributed computing in a computer cluster. While these two kinds of message passing algorithms have distinct fundamental differences, it is sometimes possible to adapt techniques developed in one domain for use in the other. This is the first goal of the research we present here. In addition to this, we will explore a number of applications for message passing algorithms, as well as improvements to existing uses of message passing algorithms.

We begin in Chapter 2 by considering the Sum-Product belief propagation algorithm in a general setting. The Sum-Product algorithm represents a generalized framework for unifying the understanding of a wide diversity of algorithms developed in engineering and science. It can be applied both in distributed low resource sensor networks as well as in centralized high performance computing hardware. For the case of sensor networks, where the conservation of energy is of critical importance and communication overhead

can quickly drain this valuable resource, we present techniques for specifically addressing the needs of this low power scenario by creating an algorithm that improves both the computational complexity of message passing as well as the communications overhead with a corresponding increase in convergence rate per unit time as compared to the standard Sum-Product belief propagation. In part, we accomplish this by adapting techniques from gossip consensus research for efficient communication over the connections in a graph.

In Chapter 3, we examine the sequential investment problem under transaction costs. In particular, we make the observation that the computations of "universal" portfolios that do not account for transaction costs can be thought of as filtering algorithms (in the sense of Kalman filtering, particle filtering, and related algorithms) that have implementations equivalent to message passing in a factor graph. This realization allows us to generalize universal portfolios to the situation with transaction costs, and furthermore provides techniques for the computational approximation of these generalized algorithms.

In Chapter 4, we turn our attention to gossip consensus and distributed estimation algorithms. Specifically, we consider the problem of estimating the parameters in a model of an agent's observations when it is known that the population as a whole is partitioned into a number of subpopulations, each of which has model parameters that are common among the member agents. We develop a method for determining the beneficial communication links in the network and study the expected squared estimation error of our algorithm.

Finally, in Chapter 5, we examine the metrics used to guide the design of data converters in the setting of digital communications. Analog to digital converters (ADCs) are typically viewed as a generic component for sensing signals. It is usually not considered whether the end use of the ADC is audio, digitization of sensor measurements, oscilloscopes, digital communications, or any of a number of other applications. Instead, it is assumed that the performance requirements of the application can be stated in terms of a small number of generic ADC performance metrics. These include effective number of bits (ENOB), total harmonic distortion (THD), signal to noise and distortion ratio (SNDR), and spurious free dynamic range (SFDR). However, these performance metrics are all focused on the faithful reproduction of observed waveforms, which is not of fundamental concern if the data

converter is to be used in a digital communications system. Therefore, we propose other information-centric rather than waveform-centric metrics that are better aligned with the goal of communications. We provide computational methods for calculating the values of these metrics, some of which are derived from Sum-Product BP or related algorithms. We also propose Statistics Gathering Converters (SGCs), which represent a change in perspective on data conversion for communications applications away from signal representation and towards the collection of relevant statistics for the purposes of decision making and detection. We show how to develop algorithms for the detection of transmitted data when the transmitted signal is received by an SGC. Finally, we provide evidence for the benefits of using system-level metrics and statistics gathering converters in communications applications.

CHAPTER 2

RESOURCE EFFICIENT BELIEF PROPAGATION ALGORITHMS

2.1 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 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 2.2, we give background on the Sum-Product Belief Propagation algorithm and discuss potential areas for increased efficiency. In Section 2.3, we look at the Stochastic Belief Propagation algorithm [3], and proceed to both generalize and simplify the algorithm. In Section 2.4, 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 proceed to prove some properties of the fixed points and convergence of the algorithm. In Section 2.5, we consider more severely constrained communications between nodes in the graph, and propose a method of belief propagation with coarsely quantized messages. In Section 2.6, 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 2.7, and discuss some potential topics for future investigation.

2.2 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

$$\Psi(\mathcal{V}) = \Psi(X) = \prod_{f\in\mathcal{F}} \psi_f(\mathcal{N}_f), \tag{2.1}$$

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 2.1: 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 $t \leftarrow t + 1$;
  /* Update function to variable messages */
5 **foreach** $(v, f) \in \mathcal{E}$ **do**
6 **if** $|\mathcal{N}_f| = 1$ **then**
7 $\theta^t_{f\to v}(X_v) = \psi_f(X_v)$;
8 **else**
9 $\theta^t_{f\to v}(X_v) = \text{Marginal}(f \to v)$;
10 **end**
11 **end**
  /* Update variable to function messages */
12 **foreach** $(v, f) \in \mathcal{E}$ **do**
13 **if** $|\mathcal{N}_v| = 1$ **then**
14 $\mu^t_{v\to f}(X_v) = \frac{1}{D}$;
15 **else**
16 $\mu^t_{v\to f}(X_v) = \frac{1}{Z} \prod_{g\in\mathcal{N}_v\setminus f} \theta^t_{g\to v}(X_v)$;
17 **end**
18 **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

$$\mathrm{P}_{X_v}(i) = \sum_{x:x_v=i} \mathrm{P}_X(X = x). \tag{2.2}$$

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 2.1. 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

$$\text{Marginal}(f \to v) = \sum_{X_u : u \in \mathcal{N}_f \setminus v} \psi_f(\mathcal{N}_f) \prod_{w \in \mathcal{N}_f \setminus v} \mu_{w \to f}^{t-1}(X_w). \tag{2.3}$$

Note that the computational complexity of the function to variable message update in Line 9, via Equation (2.3), 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 allows 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].

2.3 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.

2.3.1 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 (2.2). 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 2.2. 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 (2.3), 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 (2.3).

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 2.2: 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.

2.3.2 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 2.3. 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 2.3: 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.

2.3.3 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 2.4

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 2.4: 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 $f \in \mathcal{N}_v$

2.3.4 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.

2.4 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 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 2.5: 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 $f \in \mathcal{N}_v$

2.4.1 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 2.5. In Line 11 of Algorithm 2.5, the expression Marginal($f \to v$) differs from Equation (2.3) 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

$$\text{Marginal}(f \to v) = \sum_{X_u : u \in \mathcal{N}_f \setminus v} \psi_f(\mathcal{N}_f) \prod_{w \in \mathcal{N}_f \setminus v} \hat{\mu}^{t-1}_{w\to f}(X_w). \tag{2.4}$$

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 2.5. 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 2.5. 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 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 2.6: 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

$$\Delta\theta^t_{f\to v_2}(X_{v_2}) = \sum_{X_{v_1} \in \mathcal{U}^{t-1}_{v_1\to f}} \psi_f(X_{v_1}, X_{v_2})(\Delta\hat{\mu}^{t-1}_{v_1\to f}(X_{v_1})), \tag{2.5}$$

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 2.6. 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 (2.4). 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.

2.4.2 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

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

$$\text{subject to} \tag{2.6}$$

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

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.

2.4.3 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 (2.3), 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 2.6, 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-Projected 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.

2.4.4 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 2.4.1. *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 2.4.2 (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 2.4.2 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 2.4.2 will therefore also hold under a contractivity assumption.

Lemma 2.4.3. *Suppose the following are true:*

- *Iterative algorithm $f()$ satisfies Assumption 2.4.2.*
- *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 2.4.2 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 2.4.3 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 2.4.4. 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 2.4.5. *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 2.4.2 (with $x^* = \mathcal{M}^*_{\mathcal{V}\to\mathcal{F}}$).*
- *the message and global norms on $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}$ satisfy Assumption 2.4.4.*
- *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

$$g(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) = f(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) + e(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}), \tag{2.7}$$

and that

$$\|e(\mathcal{M}_{\mathcal{V}\to\mathcal{F}})\|_N \leq \beta\|f(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) - \mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_N \tag{2.8}$$

when $\beta \in \left[0, \left(\frac{1-\alpha}{1+\alpha}\right)\right)$, which allows us to apply Lemma 2.4.3. 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 (2.7).

Now, to verify Equation (2.8), note that the subset selection of β-Projected BP in Equation (2.6) 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 2.4.4 then leads to verification of Equation (2.8). Finally, application of Lemma 2.4.3 concludes the proof. ■

We now briefly consider what global norms satisfy Assumption 2.4.4.

Claim 2.4.6. 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 2.4.4.

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 2.4.7. 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 2.4.4.

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 2.4.8. 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 2.4.4.

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 2.4.7. ■

Theorem 2.4.5 demonstrates that if the convergence property of Assumption 2.4.2 holds for Sum-Product with respect to any one of a broad class of norms satisfying Assumption 2.4.4, 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 2.4.2 with respect to the Euclidean norm $\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_2$. We will also consider Assumption 2.4.2 with respect to the max norm $\|\mathcal{M}_{\mathcal{V}\to\mathcal{F}}\|_\infty$. Note that Theorem 2.4.5 already applies to these cases, as demonstrated by Claims 2.4.6 and 2.4.8. In addition to specializing the results with respect to 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 2.4.9. *Suppose the following are true:*

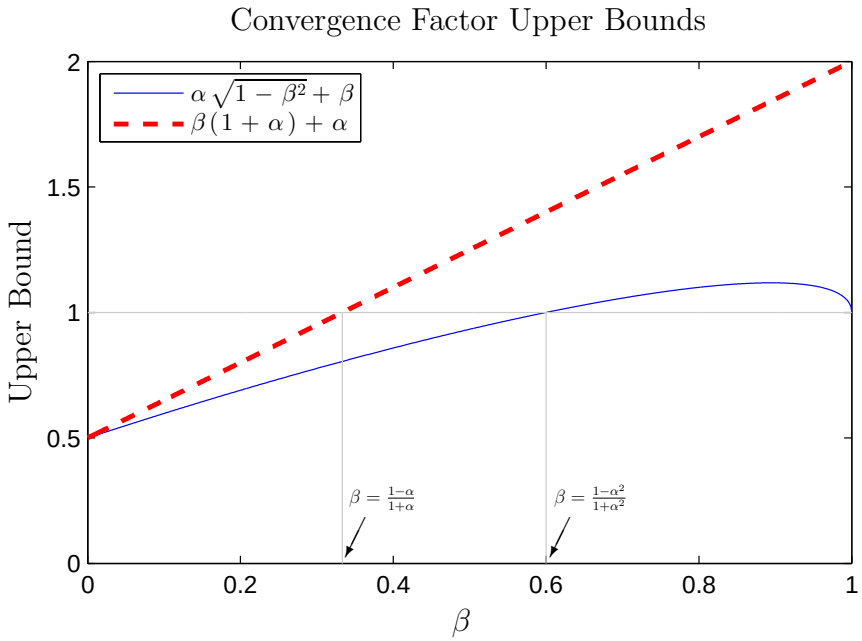


Figure 2.1: Different upper bounds on the convergence factor, where $\alpha = \frac{1}{2}$. A convergence factor less than 1 guarantees exponential convergence, with smaller factors corresponding with faster convergence. Values of β where the bounds transition to above 1 are labeled.

- *The norm N is the (global) Euclidian norm, as defined in Claim 2.4.6*
- *Iterative algorithm $f()$ satisfies Assumption 2.4.2 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

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

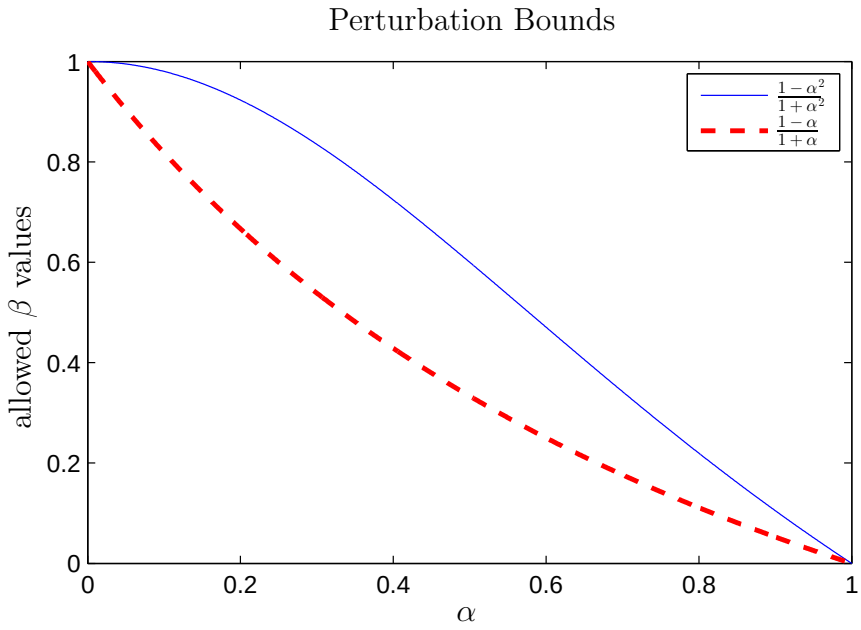


Figure 2.2: Different upper bounds on the relative perturbation size, β, for which convergence is guaranteed, as a function of α. Note that the bound specialized for the Euclidean norm includes a wider range of β values that guarantee convergence.

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 2.4.9 achieves a better bound on convergence rate than Lemma 2.4.3 by specifically accounting for the form of the global norm and the type of perturbation. This is shown in Figure 2.1. Furthermore, as shown in Figure 2.2, note that

$$\frac{1-\alpha^2}{1+\alpha^2} \geq \frac{1-\alpha}{1+\alpha},$$

and therefore Lemma 2.4.9 guarantees exponential convergence of β-Projected BP for a wider range of β values than Lemma 2.4.3.

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 2.4.2, 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

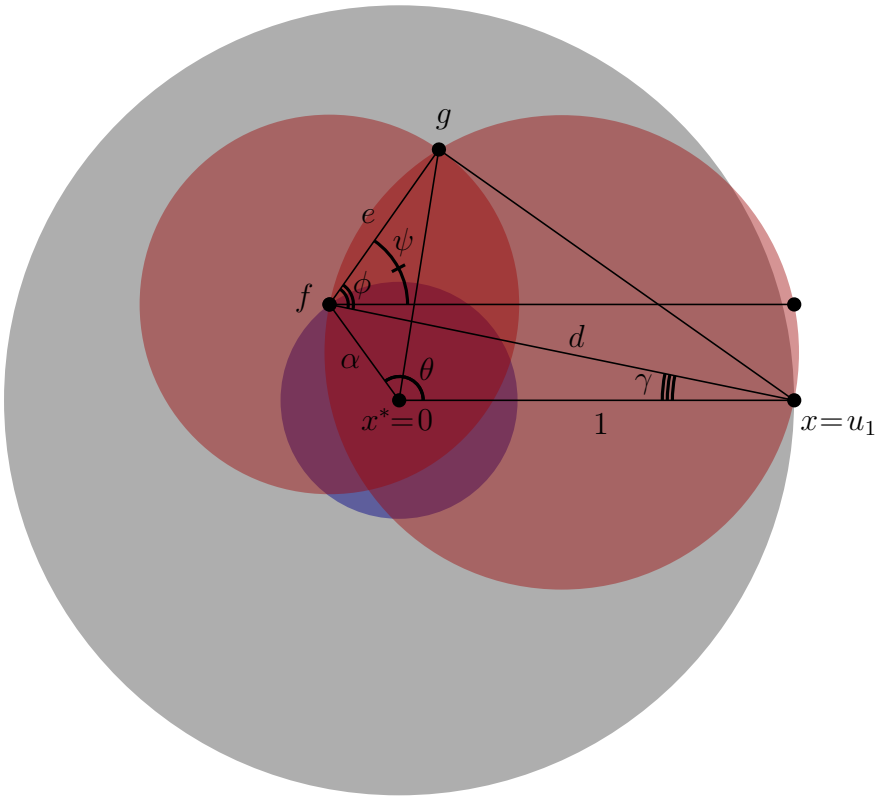


Figure 2.3: Geometry of x, f, and g.

rotate the coordinate system without affecting the value of $\tilde{\alpha}$ in this maximization. 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 2.3. 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

$$\begin{aligned} \|g\|_2^2 = &\left(\alpha\cos(\theta) + e\cos(\phi-\gamma)\right)^2 \\ &+ \left(\alpha\sin(\theta) + e\sin(\phi-\gamma)\right)^2. \end{aligned} \tag{2.9}$$

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 (2.9) 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 2.4.9 to β-Projected BP.

Theorem 2.4.10. *Suppose the following are true:*

- *The global and message norms, N and N_i respectively, are the Euclidian norms, as defined in Claim 2.4.6*
- *The Sum-Product iteration $f(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}) \triangleq G(F(\mathcal{M}_{\mathcal{V}\to\mathcal{F}}))$ satisfies Assumption 2.4.2 (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 2.4.5, but with application of Lemma 2.4.9 instead of Lemma 2.4.3. We note that the specified Euclidean norms satisfy Assumption 2.4.4, as demonstrated in Claim 2.4.6.

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 2.4.11. *Suppose the following are true:*

- *The norm N is the (global) max norm, as defined in Claim 2.4.8.*
- *Iterative algorithm $f()$ satisfies Assumption 2.4.2 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 2.4.2 under the max norm. We now note that

$$\left|\left[x^t - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty \text{ implies } \left|\left[x^{t+1} - x^*\right]_i\right| \leq \alpha \left\|x^t - x^*\right\|_\infty. \quad (2.10)$$

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 (2.10). On the other hand, we have that

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

This is true because

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

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

$$\left\|x^t - x^*\right\|_\infty \leq \left\|x^{t_0} - x^*\right\|_\infty \text{ for } t \in \mathbb{Z}, t \geq t_0. \quad (2.11)$$

This allows us to iterate Equation (2.10) 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 (2.10). 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 (2.11), we have that $\left\|x^k - x^*\right\|_\infty \leq \left\|x^t - x^*\right\|_\infty$, since $k \geq t$. Combining this with Equation (2.10), 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 (2.11) 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 2.4.11. ■

At this point, we note that Lemma 2.4.11 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 2.4.11 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 2.4.2 with respect to the max norm, then Lemma 2.4.11 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., for a factor graph that 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 marginals with respect to 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. Furthermore, if $p(u, v)$ is the number of edges from node u to node v in the tree, then the diameter d of the tree equals $\max_{(u,v)} p(u, v)$.

Theorem 2.4.12. *In a factor tree of finite diameter d and variable nodes of cardinality D, the K-Projected BP update,*

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

will converge to the unique fixed point $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}^{*}$ in at most $\frac{d}{2}\left\lceil\frac{D}{K}\right\rceil$ 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 $\hat{\mu}_{v\to f}^{t}$, is *stable* from iteration t if its value does not change with further iterations of the algorithm after iteration t. For example, if $\hat{\mu}_{v\to f}^{t+k} = \hat{\mu}_{v\to f}^{t}$ for $k \geq 0$, then $\hat{\mu}_{v\to f}^{t}$ is stable from iteration t. Note that if, from iteration t, message $[\mathcal{M}_{\mathcal{V}\to\mathcal{F}}^{t}]_i$ is stable for every component i, then $\mathcal{M}_{\mathcal{V}\to\mathcal{F}}^{t} = \mathcal{M}_{\mathcal{V}\to\mathcal{F}}^{*}$. Thus, the claim of the theorem is that there exists $t^* \leq \frac{d}{2}\left\lceil\frac{D}{K}\right\rceil$ such that $\left[\mathcal{M}_{\mathcal{V}\to\mathcal{F}}^{t^*}\right]_i$ is stable from t^* for every i.

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

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

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

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

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

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

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 $t^*_{v\to f} \leq \frac{d_*(v\to f)}{2}\left\lceil\frac{D}{K}\right\rceil$.

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 $t^*_{v\to f} \leq \left\lceil\frac{D}{K}\right\rceil = \frac{d_*(v\to f)}{2}\left\lceil\frac{D}{K}\right\rceil$. This is because it will take at most $\left\lceil\frac{D}{K}\right\rceil$ 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.

We will now perform induction on subtree depth $\hat{d}$. 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 $t^*_{\tilde{v}\to\tilde{f}} \leq \frac{d_*(\tilde{v}\to\tilde{f})}{2}\left\lceil\frac{D}{K}\right\rceil$. Now consider an edge (v, f) where, for $f' \in \mathcal{N}_v \setminus f$, 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 $\hat{d}$. Note that this implies that $d_*(v\to f) = \hat{d} + 2$.

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 $t \leq \frac{d_*(v'\to f')}{2}\left\lceil\frac{D}{K}\right\rceil \leq \frac{\hat{d}}{2}\left\lceil\frac{D}{K}\right\rceil$. Of course, we also have that $1 \leq \frac{\hat{d}}{2}\left\lceil\frac{D}{K}\right\rceil$. This makes $\mu^t_{f'\to v}$ stable for $t \leq \frac{\hat{d}}{2}\left\lceil\frac{D}{K}\right\rceil + 1$. This implies that $\mu^t_{v\to f}$ is stable for $t \leq \frac{\hat{d}}{2}\left\lceil\frac{D}{K}\right\rceil + 1$. Finally, filling in the elements of $\hat{\mu}^t_{v\to f}$ will take at most $\left\lceil\frac{D}{K}\right\rceil$ iterations, beginning no later than iteration $\frac{\hat{d}}{2}\left\lceil\frac{D}{K}\right\rceil + 1$, which implies $\hat{\mu}^t_{v\to f}$ will be stable from $t \leq \frac{\hat{d}}{2}\left\lceil\frac{D}{K}\right\rceil + \left\lceil\frac{D}{K}\right\rceil = \frac{\hat{d}+2}{2}\left\lceil\frac{D}{K}\right\rceil = \frac{d_*(v\to f)}{2}\left\lceil\frac{D}{K}\right\rceil$. 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 $t^*_{\tilde{v}\to\tilde{f}} \leq \frac{d_*(\tilde{v}\to\tilde{f})}{2}\left\lceil\frac{D}{K}\right\rceil$. 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 $t^*_{v\to f} \leq \frac{d_*(v\to f)}{2}\left\lceil \frac{D}{K} \right\rceil$. Therefore, every message $\hat{\mu}^t_{v\to f}$ will be stable from some $t^*_{v\to f} \leq \max_{(v,f)} \frac{d_*(v\to f)}{2}\left\lceil \frac{D}{K} \right\rceil = \frac{d}{2}\left\lceil \frac{D}{K} \right\rceil$. Therefore, K-Projected BP has reach a fixed point no later than iteration $\frac{d}{2}\left\lceil \frac{D}{K} \right\rceil$. 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 2.4.12. ■

2.5 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 2.4. 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.

2.5.1 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 encoding 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.

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 2.7. 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

Algorithm 2.7: 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 $f \in \mathcal{N}_v$

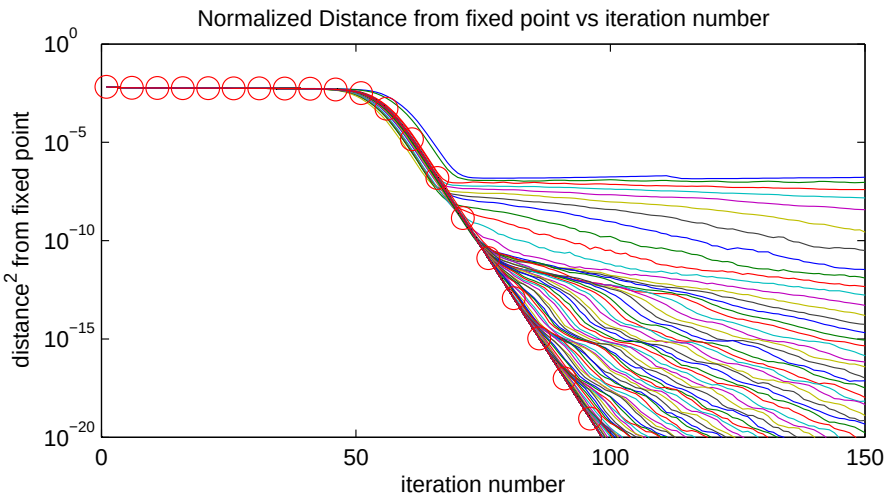


Figure 2.4: 10×10 square grid, $D = 4096$, Potts model. Normalized squared distance from the Sum-Product fixed point versus iteration number. Sum-Product is marked with circles. The other curves are K-Projections BP, where K=1 is the top curve, and larger values of K approaching the Sum-Product curve. Note that all instances of K-Projections and Sum-Product reach ∼1e-7 after about 70 iterations.

represents potentially significant savings in communication rate compared to the transmission of 32- or 64-bit floating point values.

2.6 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++.

2.6.1 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,

$$\psi_f(i,j) = \begin{cases} 1 & \text{if } i = j \\ \gamma & \text{if } i \neq j \end{cases}, \tag{2.12}$$

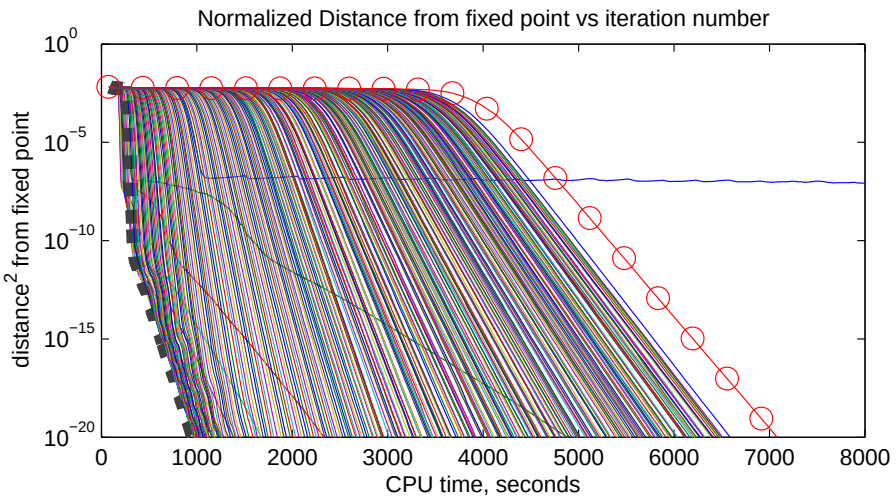


Figure 2.5: 10×10 square grid, $D = 4096$, Potts model. Normalized squared distance from the Sum-Product fixed point versus CPU time. Sum-Product is marked with circles. The other curves are K-Projections BP. For $K = 1$, convergence becomes quite slow after reaching $\sim$1e-7. For $K = \{6, 11, 16, 21\}$, 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.

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

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

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 2.4, 2.5, and 2.6 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 2.4 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

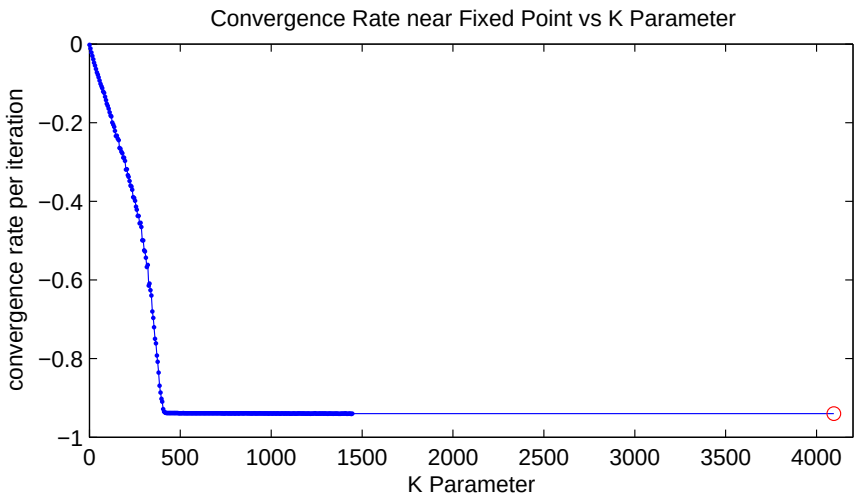


Figure 2.6: 10×10 square grid, $D = 4096$, Potts model. Convergence rate R of the normalized squared distance from the Sum-Product fixed point versus the parameter K of the K-Projections algorithm, such that $|\varepsilon_{t+1}|^2 \approx e^R |\varepsilon_t|^2$ near the fixed point. This is computed for $K = \{1, 6, 11, ..., 1446\}$. The rate of Sum-Product is marked with a circle, and corresponds with $K = 4096$.

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 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 2.5, 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 2.6, 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. Unfortunately, we were unable to determine the cause of this feature of our algorithm, but we suspect it is a result of the messages converging to sparse distributions.

2.6.2 Stereo Image Matching - Energy Minimization

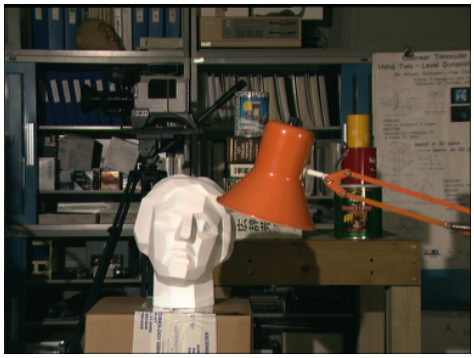
The remainder of our simulations use the "Cones" stereo image pair from [30], as shown in Figures 2.7(a) and 2.7(b), and the "Tsukuba" stereo image pair from [31], as shown in Figures 2.7(c) and 2.7(d), in order to test the algorithms on a basic model for stereo image matching. We note that the model 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 (2.12), where we have that

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

(a) "Cones" left source image.

(b) "Cones" right source image.

(c) "Tsukuba" left source image.



(d) "Tsukuba" right source image.

Figure 2.7: The "Cones" and "Tsukuba" stereo matching source images.

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

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

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

We will also define an "energy," which is a function of specific realiza-

tions $x_{i,j}$ of the variables $X_{i,j}$. 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 (2.1), and $\mathbf{x}$ is the vector of all values $x_{i,j}$.

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 (2.2). 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

$$\mathrm{P}_{X_v}(x_v) = \begin{cases} 1 & \text{if } x_v = \tilde{x}_v \\ 0 & \text{if } x_v \neq \tilde{x}_v \end{cases} . \tag{2.13}$$

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 (2.13), 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 (2.13) 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

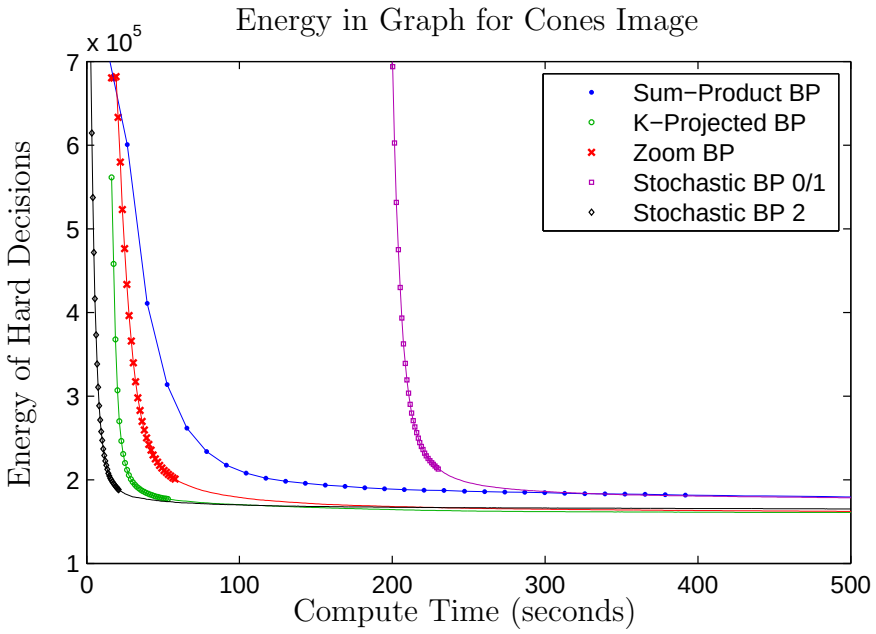


Figure 2.8: Cones Energy Graph. Markers indicate energy and compute time since the start of the initialization process for the first 30 iterations of each algorithm. Compute time includes any algorithm setup time and precomputations.

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 2.8. What we see is 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

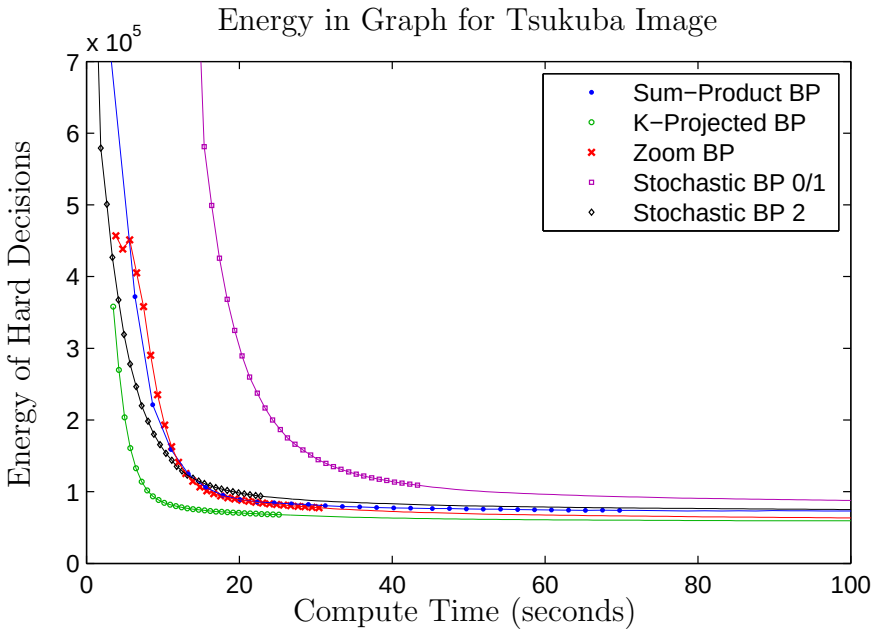
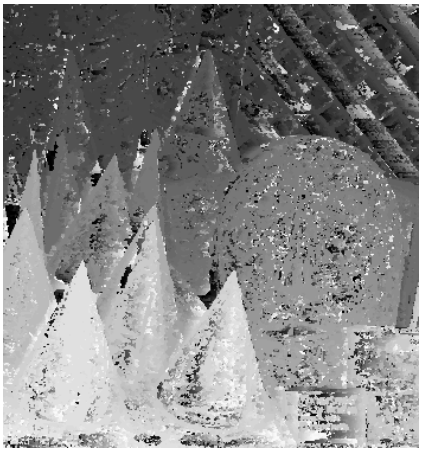


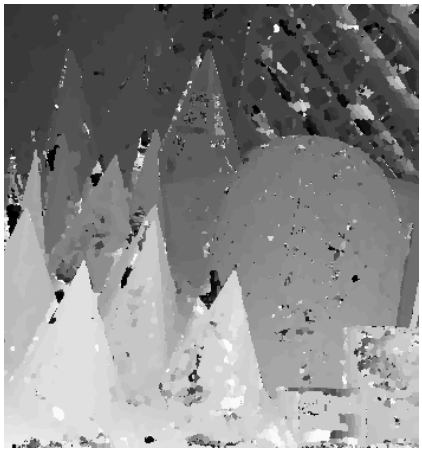
Figure 2.9: Tsukuba Energy Graph. Markers indicate energy and compute time since the start of the initialization process for the first 30 iterations of each algorithm. Compute time includes any algorithm setup time and precomputations.

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.

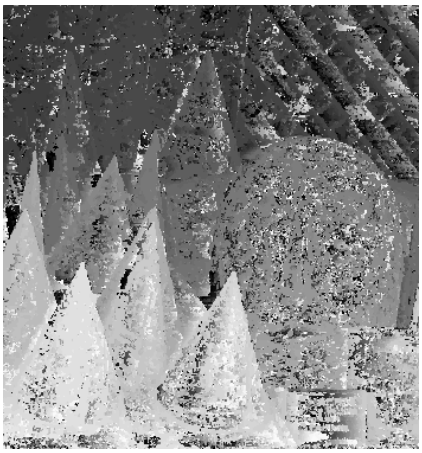
We present similar results for the Tsukaba images in Figure 2.9. Since D is smaller in this case, the per-iteration advantage of each algorithm over Sum-Product is less significant. We see that K-Projected BP converges faster and to a lower energy level than any other algorithm. Zoom BP is comparable 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.



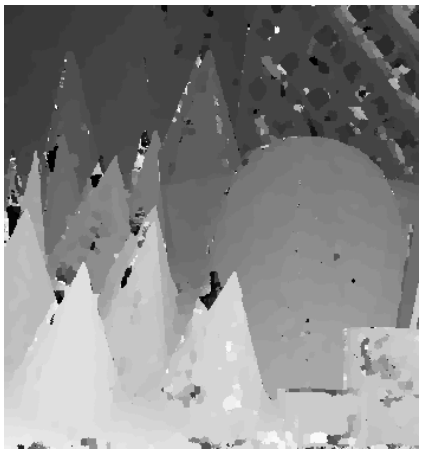
(a) Sum-Product BP. Iter=2.



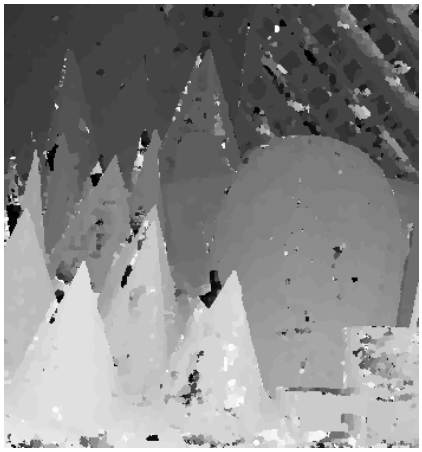
(b) K-Projected BP, $K = 1$. Iter=9.



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

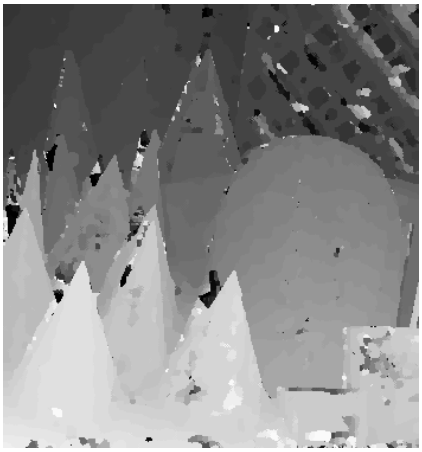


(d) Stochastic BP 2. Iter=38.

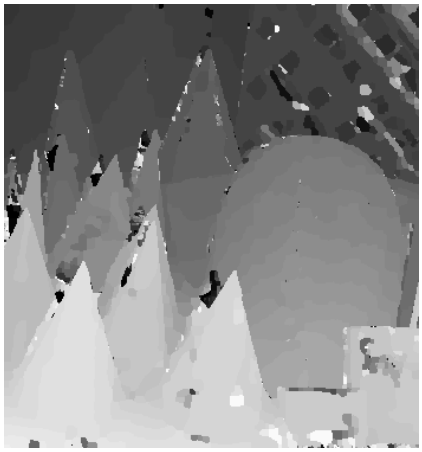


(e) Stochastic BP 0/1. Iter=38.

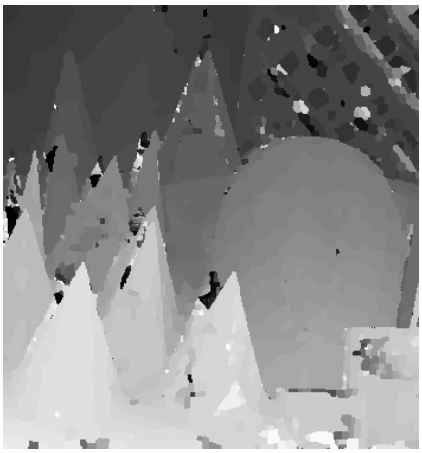
Figure 2.10: Comparison of results given same computation time, except for SBP0. Computation time is 26.6 seconds, equal to 2 iterations of Sum-Product. SBP0 did not complete its precomputation by this time. Also included in (e) is the SBP0 result after the same number of iterations as SBP2, at 238.5 seconds.



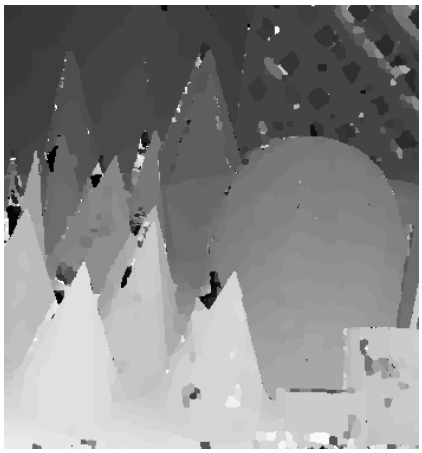
(a) Sum-Product BP. Iter=10.



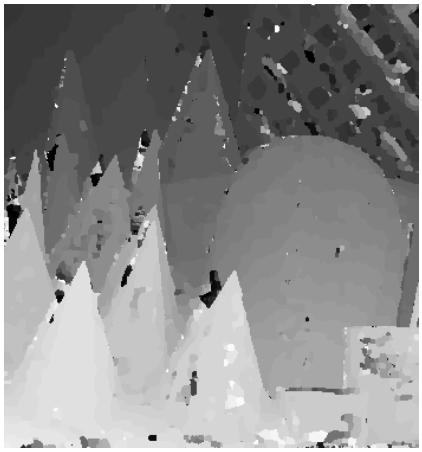
(b) K-Projected BP, $K = 1$. Iter=90.



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

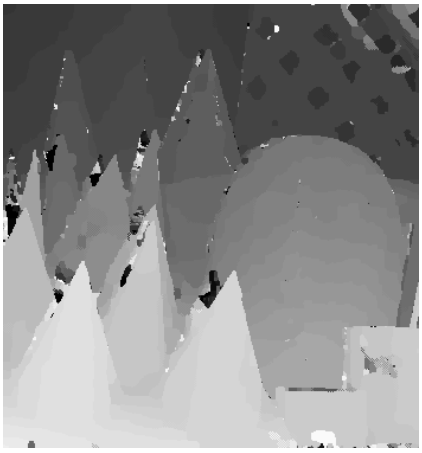


(d) Stochastic BP 2. Iter=194.

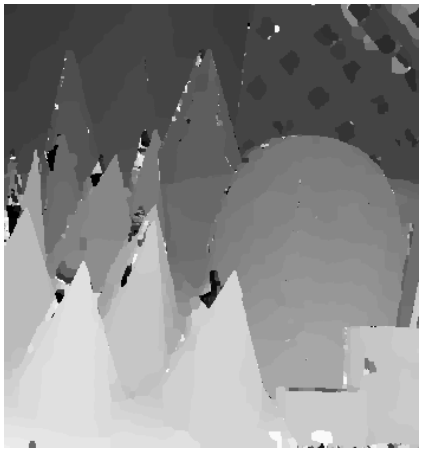


(e) Stochastic BP 0/1. Iter=194.

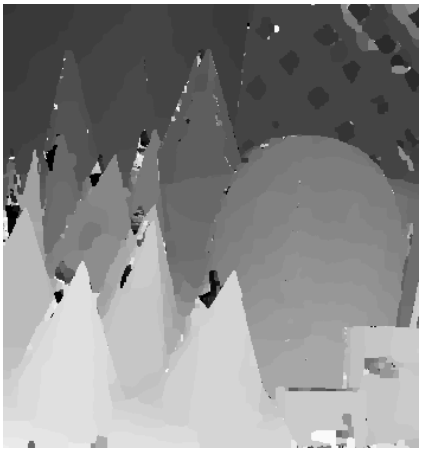
Figure 2.11: Comparison of results given same computation time, except for SBP0. Computation time is 130 seconds, equal to 10 iterations of Sum-Product. SBP0 did not complete its precomputation by this time. Also included in (e) is the SBP0 result after the same number of iterations as SBP2, at 402.8 seconds.



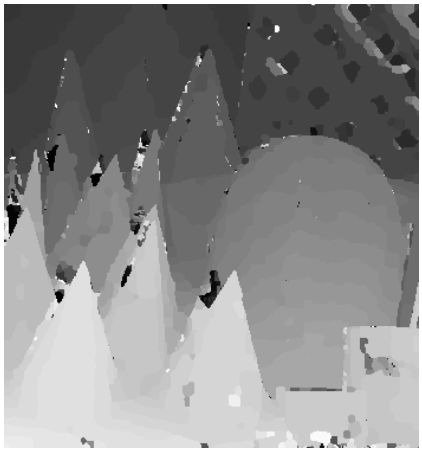
(a) Sum-Product BP.



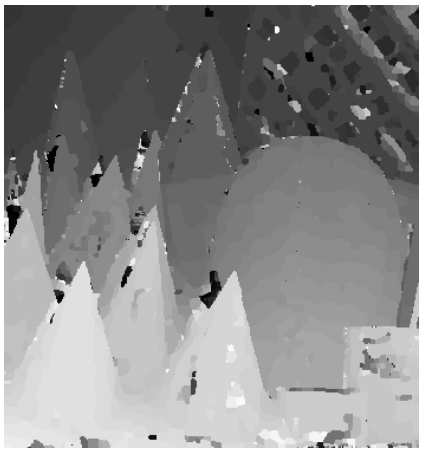
(b) K-Projected BP, $K = 1$.



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



(d) Stochastic BP 2.



(e) Stochastic BP 0/1.

Figure 2.12: Comparison of results after 1000 iterations.

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

2.6.3 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 2.10, 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 2.8. Although Stochastic BP 0 had not completed its precomputation by this point, we show the disparity map at iteration 38 in Figure 2.10(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 to or slightly worse than the results of Stochastic BP 2.

In Figure 2.11, 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

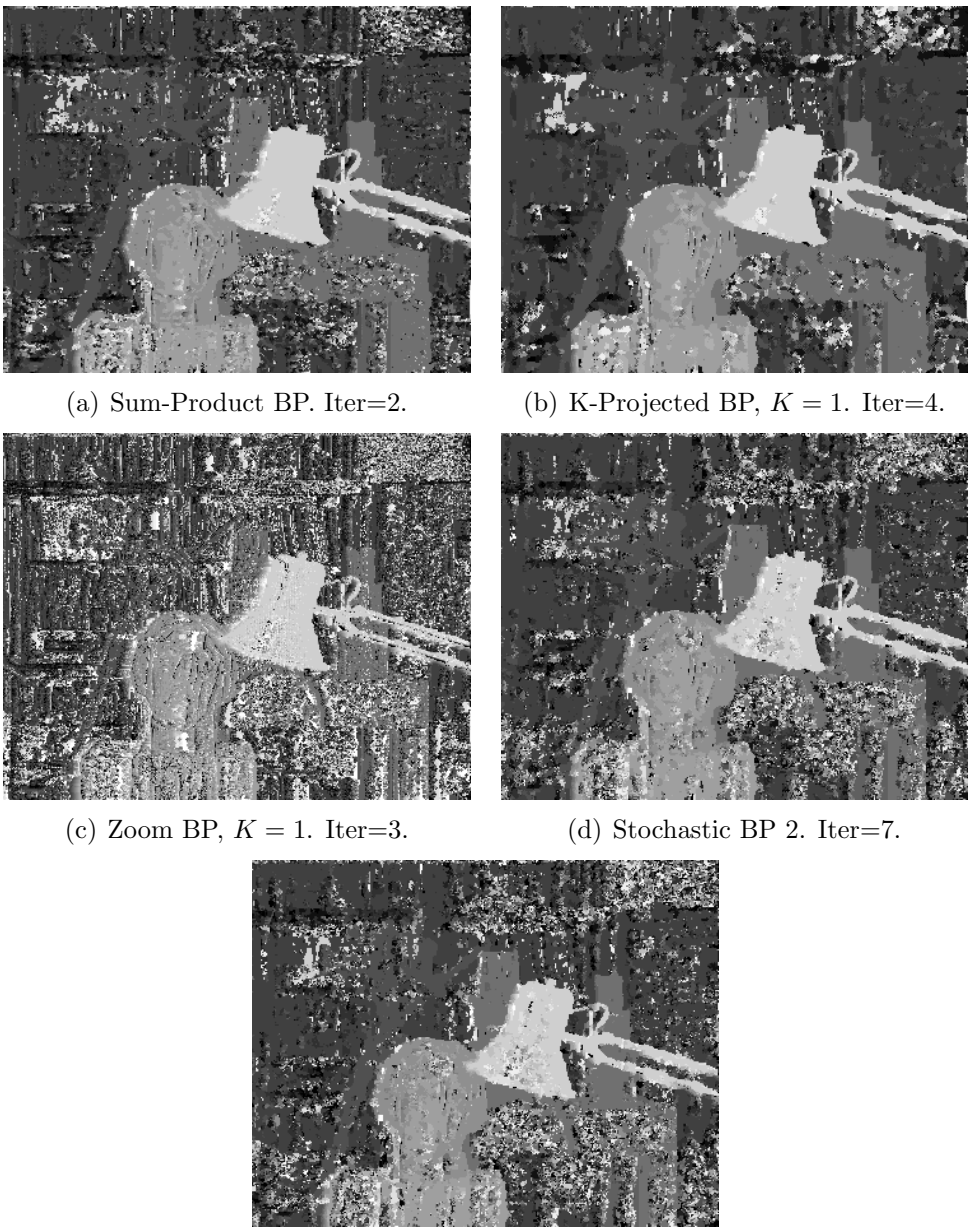
the disparity map at iteration 194 in Figure 2.11(e), which is the number of iterations of Stochastic BP 2 that had completed by 130 seconds. We see that the results of Stochastic BP 0 actually appear inferior to the results of Stochastic BP 2, especially in the regions of the fence at the back, and the mug in the front.

Finally, in Figure 2.12, we have the disparity maps that result after 1000 iterations of each of the 5 algorithms. Each of Sum-Product BP, K-Projected 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 2.13, 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 results from SBP0 are achieved after significantly more computation time.

Figure 2.14, 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 corresponding with the table, as well as the top right dark background (among other areas) in Figures 2.14(d), 2.14(e) and 2.14(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 2.9. Looking back at the Tsukuba source images, we see that these regions correspond with



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

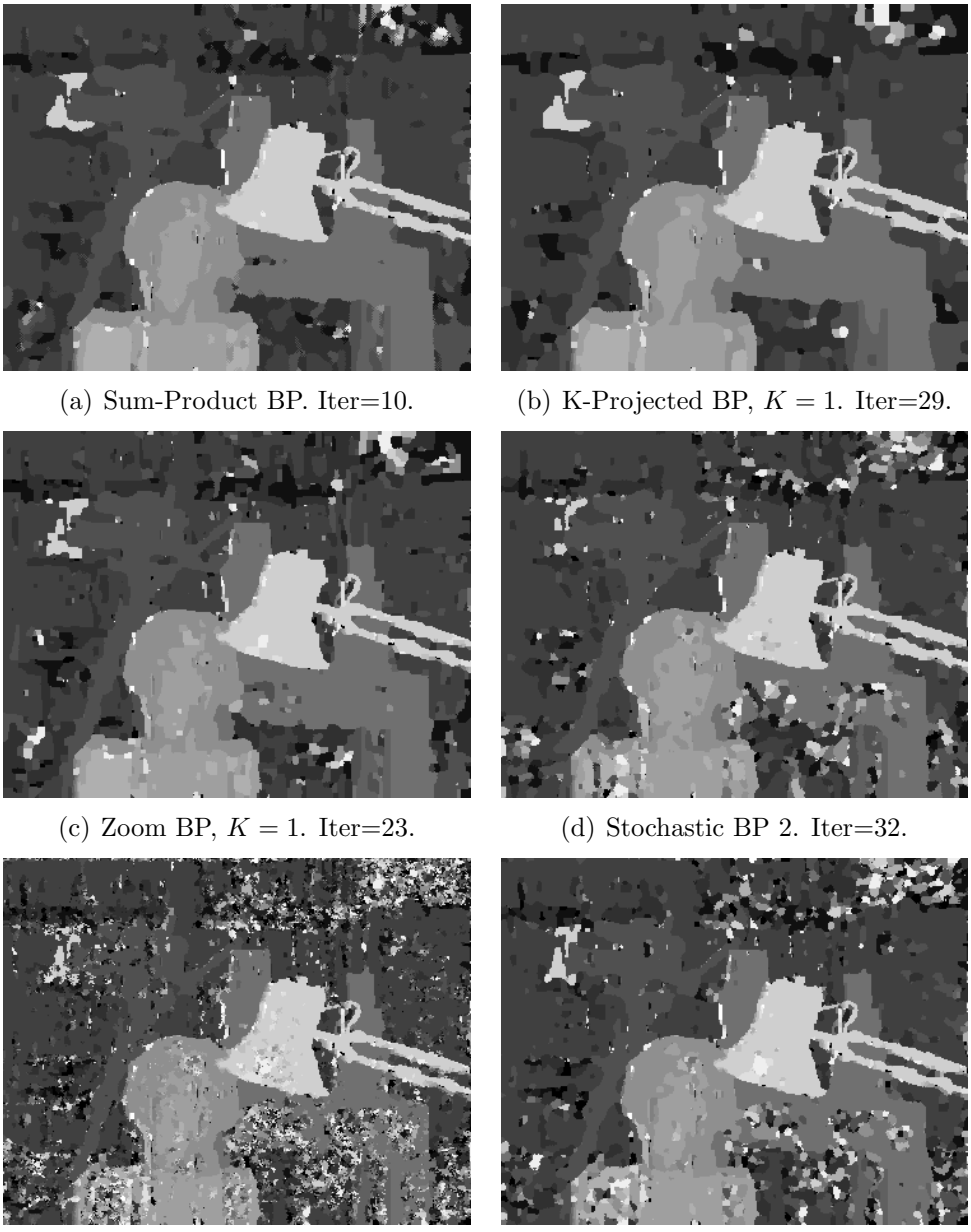
(b) K-Projected BP, $K = 1$. Iter=4.

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

(d) Stochastic BP 2. Iter=7.

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

Figure 2.13: 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.

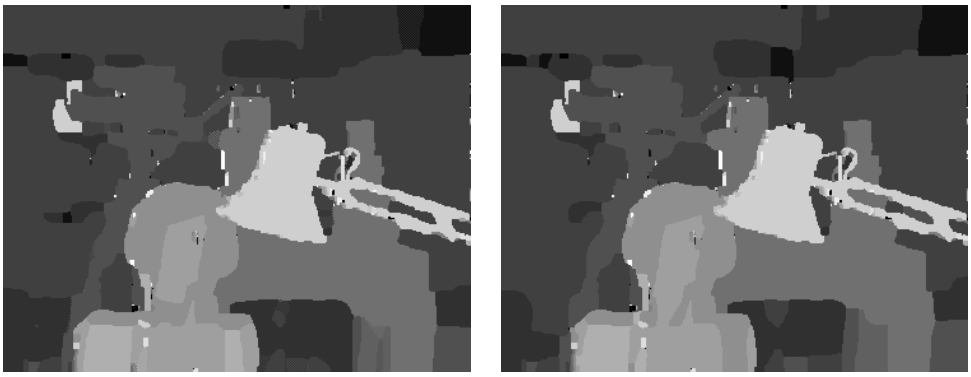
(c) Zoom BP, $K = 1$. Iter=23.

(d) Stochastic BP 2. Iter=32.

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

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

Figure 2.14: 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.

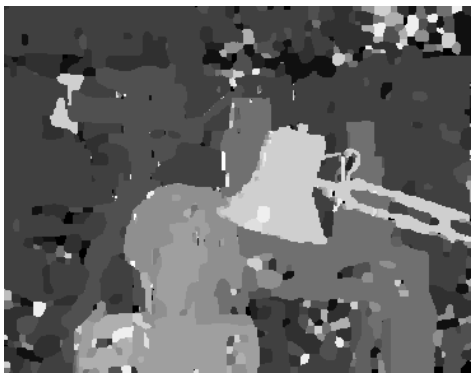


(a) Sum-Product BP.

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

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

(d) Stochastic BP 2.



(e) Stochastic BP 0/1.

Figure 2.15: Comparison of results after 1000 iterations.

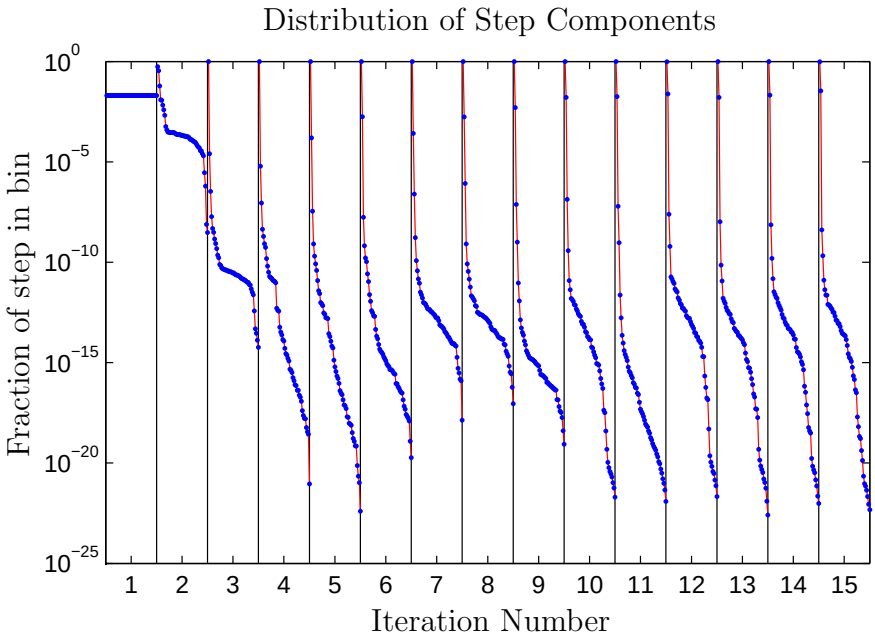


Figure 2.16: Fraction of the squared Euclidean norm of the step size $\tilde{\delta}^t(k)$ for the first 15 iterations of Sum-Product.

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 2.15, 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 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.

2.6.4 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 2.16 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

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

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 2.16, 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 $k > 1$.

2.7 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 focused on 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.

Finally, understanding how all of the belief propagation algorithms impact energy consumption would help in the design of systems that extract the greatest benefit from our algorithms. For example, if our methods are to be implemented in an integrated circuit, we would want to model energy consumption versus algorithm performance for the possible algorithm parameters, computational element energy requirements, supply voltages, and circuit speeds. If the implementation is in a distributed sensor network, this energy consumption model would also include the energy and delay costs associated with the communication of messages between network nodes.

CHAPTER 3

FACTOR GRAPHS FOR UNIVERSAL PORTFOLIOS UNDER TRANSACTION COSTS

3.1 Introduction

Sequential decisions and the sequential investment problem have been extensively studied in signal processing [34–38], computer science [39–41], finance [42–44], information theory [45–47], game theory [48, 49], and other areas. Furthermore, it has been shown [1] that factor graphs are a versatile tool for representing and extending the functionality of many existing algorithms. We consider the problem of constructing computationally feasible portfolio algorithms for investing in a stock market where we must pay transaction costs in order to adjust the allocation of wealth in our portfolio. We show that factor graphs can be used to efficiently account for these costs.[1]

One framework for studying investment strategies under penalty of such costs consists of the following market model and investment. We model the market as a sequence of price relative vectors $\mathbf{x}^n = \mathbf{x}[1], \ldots, \mathbf{x}[n]$, $\mathbf{x}[t] = (x_1[t], ..., x_m[t])^T \in \mathbb{R}_+^m$, where $\mathbb{R}_+^m$ is the positive orthant. The j^{th} entry $x_j[t]$ of the price relative vector $\mathbf{x}[t]$ represents the ratio of the opening price of the j^{th} stock on the $(t+1)^{\text{th}}$ trading period to the opening price of the same stock on the t^{th} trading period. The investment at period t is represented by a portfolio vector $\mathbf{b}[t] = (b_1[t], ..., b_m[t])^T \in \mathbb{R}_+^m$ with the constraint $\sum_{j=1}^m b_j[t] = 1$ for all t, such that we consider only long positions in each asset. We refer to this set as Δ_m. Here, each entry $b_j[t]$ corresponds to the portion of wealth invested in stock j during the t^{th} period. Note that it should be possible to include short sales and margin in our methods, in the manner of [54], but we consider only long positions in the interest of a simpler presentation.

Under the above setup, the wealth achieved by the sequence of portfolios

[1]This work has been presented in [50–53], and is reproduced with permission from IEEE.

$\mathbf{b}^n = \mathbf{b}[1], \ldots, \mathbf{b}[n]$ in the market $\mathbf{x}^n$, without transaction costs, is given by $W(\mathbf{b}^n; \mathbf{x}^n) = \prod_{t=1}^{n} \mathbf{b}^T[t]\mathbf{x}[t]$. However, to include transaction costs in our market framework, we introduce the function $C_t(\mathbf{b}_1 \to \mathbf{b}_2)$, representing the reduction in overall portfolio wealth resulting from switching from portfolio $\mathbf{b}_1$ to portfolio $\mathbf{b}_2$. We then have that the wealth achieved by the sequence of portfolios is given by

$$W_c(\mathbf{b}^n; \mathbf{x}^n) = \left(\prod_{t=1}^{n} \mathbf{b}^T[t]\mathbf{x}[t] \right) \left(\prod_{t=1}^{n-1} C_t(\mathbf{b}'[t] \to \mathbf{b}[t+1]) \right).$$

Simply put, the wealth achieved by the sequence of portfolios consists of factors $\mathbf{b}^T[t]\mathbf{x}[t]$ for the wealth change for each investment period, and additional penalty factors $C_t(\mathbf{b}'[t] \to \mathbf{b}[t+1]) \leq 1$ incurred for rebalancing between investment periods. Here, $\mathbf{b}'[t]$ is the wealth distribution at the end of period t with initial distribution $\mathbf{b}[t]$. We may also wish to include the cost of rebalancing to a new distribution at the end of the final trading period. We will write this as

$$W_c(\mathbf{b}^{n+1}; \mathbf{x}^n) = \prod_{t=1}^{n} \left((\mathbf{b}^T[t]\mathbf{x}[t]) C_t(\mathbf{b}'[t] \to \mathbf{b}[t+1]) \right).$$

We use the model of transaction costs and rebalancing referred to in [40] with fixed percentage commission c, where cD dollars are paid to the broker to buy (or sell) D dollars of stock. Finally, each portfolio vector $\mathbf{b}[t]$ must be chosen sequentially, such that the portfolio decision depends only on past information, such as the previous price relatives $\mathbf{x}[1], \ldots, \mathbf{x}[t-1]$, and not on any information from the future. In the case that there is side information, this will be modeled as a sequence of values $y^n = y[1], ..., y[n]$ from a finite alphabet $\mathcal{Y} = \{1, ..., K\}$. We assume that the investor is able to base the decision for investment period t on the value of $y[t]$.

3.1.1 Transaction costs

We use the model of transaction costs and rebalancing referred to in [40] with a fixed percentage commission, also known as *proportional transaction costs*. In particular, we make use of the following three properties of portfolio rebalancing, as given in [40]:

1. The cost of rebalancing from portfolio $\mathbf{b}_1$ to portfolio $\mathbf{b}_3$ is no more than the total cost of rebalancing from $\mathbf{b}_1$ to $\mathbf{b}_2$ and then from $\mathbf{b}_2$ to $\mathbf{b}_3$. This is trivially true under any reasonable commissions model and acceptable rebalancing strategy.

2. The cost, per dollar, of rebalancing from a distribution $\mathbf{b}_1$ to the distribution $(1-\alpha)\mathbf{b}_1+\alpha\mathbf{b}_2$, with $0 \leq \alpha \leq 1$ and $\mathbf{b}_1, \mathbf{b}_2 \in \Delta_m$, is no more than αc for some constant $0 \leq c \leq 1$ that depends only on the commission model. In particular, the factor by which the wealth of the portfolio is reduced, assuming commissions are paid for out of the wealth of the portfolio, is no worse (i.e., no less) than $(1 - \alpha c)$. This parameter c will be used as the characterizing property of the transaction costs.

3. An investment strategy I that invests an initial fraction α of its money according to investment strategy I_1 and the rest in I_2 will achieve at least α times the wealth I_1 would have achieved plus $(1 - \alpha)$ times the wealth I_2 would have achieved. More precisely, strategy I is defined by the following portfolio sequence:

$$\mathbf{b}^I[t] = \frac{\alpha W_c(I_1; \mathbf{x}^{t-1})\mathbf{b}^{I_1}[t] + (1-\alpha)W_c(I_2; \mathbf{x}^{t-1})\mathbf{b}^{I_2}[t]}{\alpha W_c(I_1; \mathbf{x}^{t-1}) + (1-\alpha)W_c(I_2; \mathbf{x}^{t-1})},$$

where $\mathbf{b}^I[t]$ is the portfolio specified by strategy I for investment period t and $W_c(I; \mathbf{x}^{t-1})$ is the wealth achieved by strategy I on the first $t-1$ trading periods, including the cost of rebalancing to portfolio $\mathbf{b}^I[t]$ in preparation for investment period t. Note that the bound on the performance of strategy I will be achieved with equality if the fraction α of the initial investment is given to an investor following I_1, the fraction $(1 - \alpha)$ is given to another investor following I_2, and these two investors independently follow their respective strategies. Strategy I can only do better by allowing these two investors to trade internally without penalty, which may help reduce the amount of commission paid. Like property 1, property 3 will hold under any reasonable commissions model and rebalancing strategy.

A special case when these properties hold is for optimal rebalancing in a market with commission taken on purchases of stock, proportional to the amount purchased. However, these properties may hold true more generally for suboptimal rebalancing procedures, when proportional commissions are charged on selling of assets, or when such commissions are charged for both buying and selling portfolio assets. As in [40], our results also apply when

there is a cost for both buying and selling stock, as well as other situations satisfying the commissions properties. As long as there is a value of c such that property 2 holds true for the combination of commission scheme and rebalancing strategy, our results will be valid.

3.1.2 Universality

The research described in this chapter is concerned with *universal* portfolio strategies, i.e., investment strategies that asymptotically achieve an exponential rate of growth of the wealth of the portfolio that is at least as high as that of the best strategy from some *competition class* of causal portfolio strategies. A significant amount of research has focused specifically on the goal of deriving such universal portfolios, such as [43], [47], [44], [40], [55], [38], [35], [56], [41], [39], and [37].

If we define $\mathcal{D}$ as the competition class, then we shall say a portfolio algorithm $\hat{\mathbf{b}}$ is universal with respect to $\mathcal{D}$ iff for every possible market sequence $\mathbf{x}^n$ we have that

$$\lim_{n\to\infty}\left(\sup_{\mathbf{d}\in\mathcal{D}}\frac{1}{n}\ln W(\mathbf{d};\mathbf{x}^n)-\frac{1}{n}\ln W(\hat{\mathbf{b}};\mathbf{x}^n)\right)\leq 0.$$

Universality in a setting with transaction costs is defined similarly, where we replace $W(\cdot;\cdot)$ with $W_c(\cdot;\cdot)$. A universal portfolio algorithm may equivalently be referred to as a portfolio with sublinear *regret*. In the given investment setup, the (cumulative) regret is defined as

$$R_n(\hat{\mathbf{b}};\mathcal{D})=\sup_{\mathbf{d}\in\mathcal{D}}\ln W(\mathbf{d};\mathbf{x}^n)-\ln W(\hat{\mathbf{b}};\mathbf{x}^n).$$

The algorithm producing $\hat{\mathbf{b}}$ has sublinear regret if $R_n(\hat{\mathbf{b}};\mathcal{D})<O(n)$. As with the definition of universality, we may define regret in a setting with transaction costs by replacing $W(\cdot;\cdot)$ with $W_c(\cdot;\cdot)$.

3.1.3 Overview

In [43], Cover presents a portfolio that is universal with respect to the class of *constant rebalanced portfolios* (CRPs). This means that the portfolio algorithm does as well as the best constant rebalanced portfolio (BCRP), where

this best CRP is determined with knowledge of the entire sequence of price relatives. (Of course, the algorithm of [43] does not have access to such information.) When we say "as well as," we mean that the amortized regret $\frac{1}{n}R_n(\hat{\mathbf{b}};\mathcal{D})$ vanishes as the number of investment periods increases, or equivalently, that the exponential growth rate of the wealth achieved by the algorithm $\frac{1}{n}\ln W(\hat{\mathbf{b}};\mathbf{x}^n)$ approaches or exceeds that of the BCRP. Cover and Ordentlich [47] then extend these results to the situation where a finite alphabet side information sequence is available to the user. In [55], Kozat and Singer extend the results of [43] to allow strategies that switch at various times. However, in [43], [47], and [55], it is assumed that no costs are incurred when reapportioning the wealth of the portfolio. Blum and Kalai [40] address this issue by constructing a sequential portfolio similar to that given by Cover in [43] that is universal with respect to CRPs in the presence of fixed percentage transaction costs. Iyengar [57] also proposed a method for incorporating transaction costs in a market with two assets that asymptotically achieves the growth rate of the best so-called interval policy, making use of similar procedures as [43]. However, there is no method given for incorporating the use of a side information sequence, for switching strategies, or other algorithms derived from [43]. In [38], Kozat and Singer do present an algorithm that is universal with respect to switching portfolios under transaction costs. In this work, we present a portfolio that is universal with respect to the same class of strategies under transaction costs considered in [38], but the algorithm is distinct from that given in [38], has a different universal performance bound, and consistently outperforms the other algorithm in our simulations. The work presented here is an attempt to combine the insights of [55], [47], and [40] in order to construct universal portfolio algorithms that account for transaction costs. Furthermore, we present factor graphs as a general tool for designing computationally feasible implementations of these universal algorithms. This research has been presented in [50], [51], [52], and [53].

We begin the discussion in Section 3.2 by presenting the algorithms of [43], [40], and [47]. In Section 3.3, we offer a new perspective on [47], and we show how this allows us to incorporate the insight of [40] to construct a new portfolio algorithm that takes into account both side information and transaction costs. We also prove that the performance of the algorithm satisfies a universal bound. In Section 3.4, we briefly discuss the difficulties

in directly implementing the portfolio from Section 3.2, and we introduce the use of factor graphs as a tool for deriving computationally more efficient implementations of, or approximations to, universal algorithms. In Section 3.5, we introduce a switching algorithm based on the portfolio from [55]. Here, we demonstrate how the techniques from Section 3.3 may be used to create a switching portfolio for use in a market with transaction costs. Furthermore, we present a proof for a universal performance bound for this switching portfolio. In Section 3.6, we demonstrate that the switching portfolio under transaction costs has structure that can be represented by factor graphs, and that this structure allows us to significantly reduce the computational complexity of the switching portfolio from Section 3.5. Finally, in Section 3.7 we present simulation results comparing our algorithms to other comparable algorithms, such as those of [47] and [40] for our side information portfolio, and the algorithms from [55] and [38] for our switching portfolio.

3.2 Preliminaries: Universal Portfolios, Transaction Costs, and Side Information

3.2.1 Cover's Universal Portfolio

In [43], Cover considers the class of *constant rebalanced portfolio* (CRP) strategies. Such a strategy chooses a particular vector $\mathbf{b}$ indicating the desired distribution for the wealth of the portfolio among the given assets, and it invests in such a way as to ensure that the wealth distribution coincides with $\mathbf{b}$ at the beginning of each trading period, including rebalancing the distribution of wealth at the beginning of each trading period. It is reasonable to hope that the best such CRP strategy should do well on a particular sequence $\mathbf{x}^n$ of price relative vectors. For example, it can be shown that if these price relative vectors are random and IID, then the growth optimal portfolio strategy in the absence of transaction costs is actually a CRP. However, in [43], Cover does not make any such statistical assumptions on the sequence of price relative vectors. Rather, it is shown that it is possible to construct an algorithm that is universal with respect to the class of constant rebalanced portfolios, as defined in Section 3.1.2.

Cover's universal portfolio from [43] is the following algorithm:

$$\hat{\mathbf{b}}[t] = \frac{\int_{\mathbf{b}\in\Delta_m} \mathbf{b} W(\mathbf{b};\mathbf{x}^{t-1})\,\mathrm{d}\mu(\mathbf{b})}{\int_{\mathbf{b}\in\Delta_m} W(\mathbf{b};\mathbf{x}^{t-1})\,\mathrm{d}\mu(\mathbf{b})}, \tag{3.1}$$

where we have that

$$\begin{aligned} W(\mathbf{b};\mathbf{x}^{t-1}) &\triangleq \textstyle\prod_{\tau=1}^{t-1} \mathbf{b}^T\mathbf{x}[\tau], \\ \Delta_m &\triangleq \left\{\mathbf{b}\in\mathbb{R}_+^m : \textstyle\sum_{j=1}^m b_j[t] = 1\right\}, \end{aligned}$$

and the distribution $\mu(\mathbf{b})$ over Δ_m is either Dirichlet $(1,\ldots,1)$ (the uniform prior on the simplex) or Dirichlet $\left(\frac{1}{2},\ldots,\frac{1}{2}\right)$. Note that this is a convex combination of all CRP strategies $\mathbf{b}\in\Delta_m$, and each CRP strategy is weighted proportionally to the wealth $W(\mathbf{b};\mathbf{x}^{t-1})$ that it would have achieved over the past market sequence.

For Dirichlet $(1,\ldots,1)$, Cover and Ordentlich [47] show that the following universal bound is achieved:

$$\sup_{\mathbf{b}\in\Delta_m} \frac{1}{n}\ln W(\mathbf{b};\mathbf{x}^n) - \frac{1}{n}\ln W(\hat{\mathbf{b}};\mathbf{x}^n) \le \frac{1}{n}(m-1)\ln(n+1).$$

For the Dirichlet $\left(\frac{1}{2},\ldots,\frac{1}{2}\right)$ prior, the following slightly improved bound is achieved:

$$\sup_{\mathbf{b}\in\Delta_m} \frac{1}{n}\ln W(\mathbf{b};\mathbf{x}^n) - \frac{1}{n}\ln W(\hat{\mathbf{b}};\mathbf{x}^n) \le \frac{1}{n}\frac{(m-1)}{2}\ln(n+1) + \frac{1}{n}\ln(2).$$

This portfolio algorithm and the given bounds, however, do not take into account transaction costs.

3.2.2 Universal Portfolio under Transaction Costs

Blum and Kalai [40] offer a solution to the problem of transaction costs with the following portfolio algorithm:

$$\hat{\mathbf{b}}[t] = \frac{\int_{\mathbf{b}\in\Delta_m} \mathbf{b} W_c(\mathbf{b};\mathbf{x}^{t-1})\,\mathrm{d}\mu(\mathbf{b})}{\int_{\mathbf{b}\in\Delta_m} W_c(\mathbf{b};\mathbf{x}^{t-1})\,\mathrm{d}\mu(\mathbf{b})}, \tag{3.2}$$

where we have that

$$W_c(\mathbf{b};\mathbf{x}^{t-1}) \triangleq \prod_{\tau=1}^{t-1}\left(\mathbf{b}^T\mathbf{x}[\tau]\times C_\tau(\mathbf{b}'\to\mathbf{b})\right),$$

and the distribution $\mu(\mathbf{b})$ over Δ_m is Dirichlet $(1,\ldots,1)$. Again, the portfolio algorithm is a convex combination of all CRP strategies, with each weighted proportionally to the wealth it would have achieved over the past, including transaction costs. It is shown [40] that the following universal bound is achieved:

$$\sup_{\mathbf{b}\in\Delta_m}\frac{1}{n}\ln W_c(\mathbf{b};\mathbf{x}^n)-\frac{1}{n}\ln W_c(\hat{\mathbf{b}};\mathbf{x}^n)\le\frac{1}{n}(m-1)\ln\left(n(1+c)+1\right).$$

3.2.3 Universal Portfolio with Side Information

Typically, an investor is able to base investment decisions on some *side information*. For our purposes, this side information is modeled as a sequence of values $y^n = y[1],...,y[n]$. Each $y[t]$ is from a finite alphabet, which we can take to be $\mathcal{Y}=\{1,...,K\}$ without loss of generality. We assume that the investor is able to base the decision for investment period t on the value of $y[t]$. In [47], the authors present an adaptation of the algorithm in (3.1) resulting in a universal portfolio algorithm that is able to make use of the side information sequence y^n. The resulting side information portfolio is the following:

$$\hat{\mathbf{b}}[t]=\frac{\int_{\mathbf{b}\in\Delta_m}\mathbf{b}W^{y[t]}(\mathbf{b};\mathbf{x}^{t-1},y^t)\,\mathrm{d}\mu(\mathbf{b})}{\int_{\mathbf{b}\in\Delta_m}W^{y[t]}(\mathbf{b};\mathbf{x}^{t-1},y^t)\,\mathrm{d}\mu(\mathbf{b})},\tag{3.3}$$

where we have that

$$W^{y[t]}(\mathbf{b};\mathbf{x}^{t-1},y^t) \triangleq \prod_{\tau\in\mathcal{T}^t_{y[t]}}\mathbf{b}^T\mathbf{x}[\tau]\tag{3.4}$$

and

$$\mathcal{T}^t_y \triangleq \{\tau\in\mathbb{N}:\tau<t\ ,\ y[\tau]=y\}.$$

The distribution $\mu(\mathbf{b})$ over Δ_m can be Dirichlet $(\alpha,\ldots,\alpha)$ for $\alpha=1$ or $\alpha=\frac{1}{2}$. It should be noted that the algorithm amounts to running independent universal portfolios of the form in (3.1) on each of the K subsequences of $\mathbf{x}^n$,

$\mathbf{x}_y^n$, consisting of the price relatives $\mathbf{x}[t]$ when $y[t] = y$. For both of the distributions $\mu(\mathbf{b})$, there is a corresponding universal bound indicating that the algorithm asymptotically performs as well as the best state-constant rebalanced portfolio $\mathbf{d} = \{\mathbf{b}_1, ..., \mathbf{b}_K\} \in \Delta_m^K$. A state-constant rebalanced portfolio is a strategy that uses the same portfolio $\mathbf{b}_y$ whenever $y[t] = y$. For Dirichlet $(1, \ldots, 1)$, we have that

$$\sup_{\mathbf{d} \in \Delta_m^K} \frac{1}{n} \ln W(\mathbf{d}; \mathbf{x}^n, y^{n+1}) - \frac{1}{n} \ln W(\hat{\mathbf{b}}; \mathbf{x}^n, y^{n+1}) \leq \frac{1}{n} K(m-1) \ln (n+1),$$

where we define

$$W(\mathbf{d}; \mathbf{x}^n, y^{n+1}) = \prod_{\tau=1}^{n} \mathbf{b}_{y[\tau]}^T \mathbf{x}[\tau].$$

For Dirichlet $\left(\frac{1}{2}, \ldots, \frac{1}{2}\right)$, we have that

$$\sup_{\mathbf{d} \in \Delta_m^K} \frac{1}{n} \ln W(\mathbf{d}; \mathbf{x}^n, y^{n+1}) - \frac{1}{n} \ln W(\hat{\mathbf{b}}; \mathbf{x}^n, y^{n+1}) \leq \frac{1}{n} \frac{K(m-1)}{2} \ln (n+1) + \frac{1}{n} K \ln (2).$$

Proving these bounds involves simply applying the results of Section 3.2.1 to each of the subsequences $\mathbf{x}_y^n$ for $y = 1, ..., K$. This is possible because the cumulative factor of wealth gained over any given subsequence is independent of all the other subsequences.

3.3 Universal Portfolios with Side Information under Transaction Costs

The natural extension to the portfolios described in Section 3.2 is to develop a portfolio algorithm that both uses a given side information sequence and takes into account transaction costs. However, consider the naive approach of attempting to take into account both transaction costs and side information. This may involve running K independent copies of the algorithm defined by (3.2), where each copy runs only on the subsequence defined by a corresponding side information value. This is in direct analogy to the development in [47] of the side information universal portfolio introduced in Section 3.2.3. We simply note that such an algorithm does not effectively

account for transaction costs. This is because the cost factors of switching between subsequences, such that $y[t] \neq y[t-1]$, are not taken into account. Furthermore, potentially most of the cost factors that would be taken into account by (3.2) as it is executed along a subsequence would be meaningless, as they will usually not truly correspond with the transitions between consecutive portfolio assignments. Hence, a more sophisticated method of incorporating both transaction costs and side information must be developed. This is one contribution of our work.

Before presenting our algorithm, we first consider that both algorithms described by (3.1) and (3.2) may be written in the following way:

$$\begin{aligned} P_t(\mathbf{d}) &= \frac{1}{Z_t} P_{t-1}(\mathbf{d}) f_{t-1}(\mathbf{d}), \\ \hat{\mathbf{b}}[t] &= \mathrm{E}_{P_t(\mathbf{d})}[g_t(\mathbf{d})] \end{aligned} \tag{3.5}$$

for some non-negative factor $f_{t-1}(\mathbf{d})$ and normalizing constant Z_t. (We use the symbol Z whenever it is necessary to normalize to a proper "probability" distribution. Its value should be obvious based on the context.) The variable $\mathbf{d}$ represents one from a set $\mathcal{D}$ of portfolio strategies, such as constant rebalanced portfolios, and the function $g_t(\cdot)$ maps the strategy $\mathbf{d}$ to a portfolio vector $\mathbf{b} \in \Delta_m$. In [43] and [40], we have that $\mathcal{D} = \Delta_m$ (i.e., $\mathbf{d} = \mathbf{b}$) and $g_t(\cdot)$ is the identity function. The distribution $P_t(\mathbf{d})$ is simply the wealth function $W(\mathbf{d}; \mathbf{x}^{t-1})$ or $W_c(\mathbf{d}; \mathbf{x}^{t-1})$, possibly multiplied by some a priori distribution $\mu(\mathbf{d})$, and then normalized. Hence, we can take $P_0(\mathbf{d})$ as the uniform distribution, and $f_0(\mathbf{d}) = \mu(\mathbf{d})$. The $E_{P_t(\mathbf{d})}[\cdot]$ indicates an expectation with respect to the distribution $P_t(\mathbf{d})$.

Now, as presented in [47], the algorithm in (3.3) is not of the form of (3.5), as it involves jumping between the various independently running algorithms for each side information value. However, we can consider the following algorithm, using side information y^n, which is in the form of (3.5) and is defined as follows:

Let the portfolio strategy space be

$$\mathcal{D} = \Delta_m^K = \{\mathbf{d} = (\mathbf{b}_1, ..., \mathbf{b}_K) : \mathbf{b}_i \in \Delta_m \text{ for } i = 1, ..., K\}. \tag{3.6}$$

We now define

$$f_0(\mathbf{d}) = \mu(\mathbf{b}_1) \times \ldots \times \mu(\mathbf{b}_K), \tag{3.7}$$

where the distribution $\mu(\cdot)$ is either the uniform Dirichlet $(1, \ldots, 1)$ prior or the Dirichlet $\left(\frac{1}{2}, \ldots, \frac{1}{2}\right)$ prior, and $P_0(\mathbf{d})$ is the uniform distribution over Δ_m^K. Furthermore, for $t \geq 1$, define

$$f_t(\mathbf{d}) = \mathbf{b}_{y[t]}^T \mathbf{x}[t]. \tag{3.8}$$

Finally, with $\mathbf{d} = (\mathbf{b}_1, ..., \mathbf{b}_K)$, we define

$$g_t(\mathbf{d}) = \mathbf{b}_{y[t]}. \tag{3.9}$$

All brought together, this definition of portfolio algorithm under (3.5) produces the following portfolio:

$$\hat{\mathbf{b}}[t] = \frac{\int_{\mathbf{d}\in\mathcal{D}} \mathbf{b}_{y[t]} W(\mathbf{d}; \mathbf{x}^{t-1}, y^t)\, \mathrm{d}\mu_{\mathcal{D}}(\mathbf{d})}{\int_{\mathbf{d}\in\mathcal{D}} W(\mathbf{d}; \mathbf{x}^{t-1}, y^t)\, \mathrm{d}\mu_{\mathcal{D}}(\mathbf{d})}, \tag{3.10}$$

where we have that

$$W(\mathbf{d}; \mathbf{x}^{t-1}, y^t) = \prod_{\tau=1}^{t-1} \mathbf{b}_{y[\tau]}^T \mathbf{x}[\tau] \tag{3.11}$$

and

$$\mu_{\mathcal{D}}(\mathbf{d}) = \prod_{y=1}^{K} \mu(\mathbf{b}_y)\ .$$

We note that again, this portfolio is a convex combination of all the portfolio strategies $\mathcal{D}$, with each weighted proportionally to the wealth it would have achieved over the past.

Now, computing this algorithm directly can be very expensive, as the expectation in (3.5) involves evaluating an integral over the potentially high dimensional space Δ_m^K. However, consider reordering the product terms in (3.11) as follows:

$$\begin{aligned} W(\mathbf{d}; \mathbf{x}^{t-1}, y^t) &= \prod_{y=1}^{K} \left(\prod_{\{\tau:\tau<t, y[\tau]=y\}} \mathbf{b}_{y[\tau]}^T \mathbf{x}[\tau] \right) \\ &= \prod_{y=1}^{K} \left(\prod_{\tau\in\mathcal{T}_y^t} \mathbf{b}_y^T \mathbf{x}[\tau] \right). \end{aligned} \tag{3.12}$$

We may then recognize the product in parentheses is the same as (3.4), with $\mathbf{b} = \mathbf{b}_y$, which gives us

$$\begin{aligned} W(\mathbf{d}; \mathbf{x}^{t-1}, y^t) &= \prod_{y=1}^{K} W^y(\mathbf{b}_y; \mathbf{x}^{t-1}, y^t) \\ &= W^{y[t]}(\mathbf{b}_{y[t]}; \mathbf{x}^{t-1}, y^t) \prod_{y \neq y[t]} W^y(\mathbf{b}_y; \mathbf{x}^{t-1}, y^t). \end{aligned} \tag{3.13}$$

Hence, (3.10) becomes

$$\hat{\mathbf{b}}[t] = \frac{\int_{\mathbf{d} \in \mathcal{D}} \mathbf{b}_{y[t]} \left(W^{y[t]}(\mathbf{b}_{y[t]}; \mathbf{x}^{t-1}, y^t) \prod_{y \neq y[t]} W^y(\mathbf{b}_y; \mathbf{x}^{t-1}, y^t) \right) \mathrm{d}\mu_{\mathcal{D}}(\mathbf{d})}{\int_{\mathbf{d} \in \mathcal{D}} \left(W^{y[t]}(\mathbf{b}_{y[t]}; \mathbf{x}^{t-1}, y^t) \prod_{y \neq y[t]} W^y(\mathbf{b}_y; \mathbf{x}^{t-1}, y^t) \right) \mathrm{d}\mu_{\mathcal{D}}(\mathbf{d})}. \tag{3.14}$$

However, it is possible to write the numerator of (3.14) as

$$\begin{aligned} &\left(\int_{\mathbf{b}_{y[t]} \in \Delta_m} \mathbf{b}_{y[t]} W^{y[t]}(\mathbf{b}_{y[t]}; \mathbf{x}^{t-1}, y^t) \, \mathrm{d}\mu(\mathbf{b}_{y[t]}) \right) \\ &\qquad \times \left(\int_{\mathbf{b}_y : y \neq y[t]} \prod_{y \neq y[t]} W^y(\mathbf{b}_y; \mathbf{x}^{t-1}, y^t) \, \mathrm{d}\mu(\mathbf{b}_y : y \neq y[t]) \right) \end{aligned}$$

and the denominator as

$$\begin{aligned} &\left(\int_{\mathbf{b}_{y[t]} \in \Delta_m} W^{y[t]}(\mathbf{b}_{y[t]}; \mathbf{x}^{t-1}, y^t) \, \mathrm{d}\mu(\mathbf{b}_{y[t]}) \right) \\ &\qquad \times \left(\int_{\mathbf{b}_y : y \neq y[t]} \prod_{y \neq y[t]} W^y(\mathbf{b}_y; \mathbf{x}^{t-1}, y^t) \, \mathrm{d}\mu(\mathbf{b}_y : y \neq y[t]) \right). \end{aligned}$$

This allows the cancellation of the integrals on the right, resulting in

$$\hat{\mathbf{b}}[t] = \frac{\int_{\mathbf{b} \in \Delta_m} \mathbf{b} W^{y[t]}(\mathbf{b}; \mathbf{x}^{t-1}, y^t) \, \mathrm{d}\mu(\mathbf{b})}{\int_{\mathbf{b} \in \Delta_m} W^{y[t]}(\mathbf{b}; \mathbf{x}^{t-1}, y^t) \, \mathrm{d}\mu(\mathbf{b})},$$

which is identical to the side information portfolio given in (3.3). This demonstrates that the side information portfolio in (3.3) is indeed of the form in

(3.5).

The purpose of this exercise is to allow us to extend the side information portfolio of [47] to the situation with transaction costs by drawing direct analogy to the insight of [40], as follows: The universal portfolio of [43] (from (3.1)) is constructed using

$$f_t(\mathbf{b}) = \mathbf{b}^T \mathbf{x}[t]. \tag{3.15}$$

This algorithm is generalized to incorporate transaction costs by modifying (3.15) to be

$$f_t(\mathbf{b}) = \mathbf{b}^T \mathbf{x}[t] \times C_t(\mathbf{b}' \rightarrow \mathbf{b}).$$

Similarly, we can modify (3.8) of the side information portfolio to be

$$f_t(\mathbf{d}) = \mathbf{b}_{y[t]}^T \mathbf{x}[t] \times C_t(\mathbf{b}_{y[t]}{}' \rightarrow \mathbf{b}_{y[t+1]}). \tag{3.16}$$

Hence, when we combine the definitions in (3.5), (3.6), (3.7), (3.9), and (3.16), we arrive at the algorithm we propose that takes into account both side information and transaction costs. The investment at each period is given by

$$\hat{\mathbf{b}}[t] = \frac{\int_{\mathbf{d}\in\mathcal{D}} \mathbf{b}_{y[t]} W_c(\mathbf{d}; \mathbf{x}^{t-1}, y^t)\, \mathrm{d}\mu_{\mathcal{D}}(\mathbf{d})}{\int_{\mathbf{d}\in\mathcal{D}} W_c(\mathbf{d}; \mathbf{x}^{t-1}, y^t)\, \mathrm{d}\mu_{\mathcal{D}}(\mathbf{d})}. \tag{3.17}$$

This result is a performance-weighted convex combination of all the portfolio strategies in the set $\mathcal{D} = \Delta_m^K$. The wealth terms $W_c(\mathbf{d}; \mathbf{x}^{t-1}, y^t)$ include the cost of rebalancing in preparation for investment period t. It should be noted that our algorithm reduces to (3.3) when there are no transaction costs, to (3.2) when $K = 1$, and to (3.1) when $K = 1$ and there are no transaction costs. In this work, we consider only the use of the uniform distribution for $\mu_{\mathcal{D}}(\cdot)$.

3.3.1 Universal Performance Bound

The following theorem applies to the portfolio with side information under transaction costs:

Theorem 1: *The wealth achieved by the portfolio algorithm* $\hat{\mathbf{b}}$ *defined*

above is such that

$$\sup_{\mathbf{d}\in\mathcal{D}} \frac{1}{n} \ln W_c(\mathbf{d}; \mathbf{x}^n, y^{n+1}) - \frac{1}{n} \ln W_c(\hat{\mathbf{b}}; \mathbf{x}^n, y^{n+1}) \leq \frac{1}{n} K(m-1) \ln\left(n(1+c)+1\right).$$

The parameter c defines the amount of transaction costs as in [40] and property 2 from Section 3.1.1. The wealth terms here include the cost of rebalancing in preparation for investment period $n+1$.

Proof. The proof of Theorem 1 proceeds as follows: Consider two side information dependent portfolio strategies $\mathbf{d}$ and $\mathbf{d}^1$ from Δ_m^K. Furthermore, suppose we choose a scalar function $\alpha(\mathbf{d})$ ($0 \leq \alpha(\mathbf{d}) \leq 1$ for all $\mathbf{d} \in \Delta_m^K$) such that $\mathbf{d} = (1-\alpha(\mathbf{d}))\mathbf{d}^1 + \alpha(\mathbf{d})\mathbf{d}^2$ for some $\mathbf{d}^2 \in \Delta_m^K$. The first property we note is that the two portfolio strategies $\mathbf{d}$ and $\mathbf{d}^1$ satisfy the following relationship:

$$\begin{aligned} W(\mathbf{d}; \mathbf{x}[t], y[t]) &= \mathbf{b}_{y[t]}^T \mathbf{x}[t] \\ &= \left((1-\alpha(\mathbf{d}))\mathbf{b}_{y[t]}^1 + \alpha(\mathbf{d})\mathbf{b}_{y[t]}^2\right)^T \mathbf{x}[t] \\ &= (1-\alpha(\mathbf{d}))\mathbf{b}_{y[t]}^{1\,T}\mathbf{x}[t] + \alpha(\mathbf{d})\mathbf{b}_{y[t]}^{2\,T}\mathbf{x}[t] \\ &\geq (1-\alpha(\mathbf{d}))W(\mathbf{d}^1; \mathbf{x}[t], y[t]). \end{aligned}$$

The $W(\mathbf{d}; \mathbf{x}[t], y[t])$ is the change in portfolio wealth for strategy $\mathbf{d}$ for (the single) trading period t without paying transaction costs.

We now examine the factors by which the wealths of $\mathbf{d}$ and $\mathbf{d}^1$ change during a single investment period, including the cost of rebalancing. For any strategy $\mathbf{d}$, we have that

$$W_c(\mathbf{d}; \mathbf{x}[t], y[t], y[t+1]) = W(\mathbf{d}; \mathbf{x}[t], y[t]) C_t(\mathbf{b}_{y[t]}{}' \rightarrow \mathbf{b}_{y[t+1]}),$$

where $W_c(\mathbf{d}; \mathbf{x}[t], y[t], y[t+1])$ now includes the cost of rebalancing the portfolio. Here, $\mathbf{b}_{y[t]}{}'$ is the portfolio distribution at the end of the investment period, where at the beginning the distribution was $\mathbf{b}_{y[t]}$, and $C_t(\mathbf{b}_{y[t]}{}' \rightarrow \mathbf{b}_{y[t+1]})$ is the cost factor for rebalancing from $\mathbf{b}_{y[t]}{}'$ to $\mathbf{b}_{y[t+1]}$.

Furthermore, by making use of the three properties of rebalancing portfo-

lios from Section 3.1.1, we have that

$$
\begin{aligned}
&W_c(\mathbf{d}; \mathbf{x}[t], y[t], y[t+1]) \\
&\quad = W(\mathbf{d}; \mathbf{x}[t], y[t]) C_t(\mathbf{b}_{y[t]}{}' \to \mathbf{b}_{y[t+1]}) \\
&\quad = \big((1-\alpha(\mathbf{d}))W(\mathbf{d}^1; \mathbf{x}[t], y[t]) + \alpha(\mathbf{d})W(\mathbf{d}^2; \mathbf{x}[t], y[t])\big) \\
&\qquad\qquad \times C_t(\mathbf{b}_{y[t]}{}' \to \mathbf{b}_{y[t+1]}) \\
&\quad \geq (1-\alpha(\mathbf{d}))W(\mathbf{d}^1; \mathbf{x}[t], y[t]) C_t(\mathbf{b}^1_{y[t]}{}' \to \mathbf{b}_{y[t+1]}) \\
&\qquad\qquad + \alpha(\mathbf{d})W(\mathbf{d}^2; \mathbf{x}[t], y[t]) C_t(\mathbf{b}^2_{y[t]}{}' \to \mathbf{b}_{y[t+1]}) \qquad (3.18) \\
&\quad \geq (1-\alpha(\mathbf{d}))W(\mathbf{d}^1; \mathbf{x}[t], y[t]) C_t(\mathbf{b}^1_{y[t]}{}' \to \mathbf{b}_{y[t+1]}) \\
&\quad \geq (1-\alpha(\mathbf{d}))W(\mathbf{d}^1; \mathbf{x}[t], y[t]) \\
&\qquad\qquad \times C_t(\mathbf{b}^1_{y[t]}{}' \to \mathbf{b}^1_{y[t+1]}) C_t(\mathbf{b}^1_{y[t+1]} \to \mathbf{b}_{y[t+1]}) \qquad (3.19) \\
&\quad \geq (1-\alpha(\mathbf{d}))W(\mathbf{d}^1; \mathbf{x}[t], y[t]) \\
&\qquad\qquad \times C_t(\mathbf{b}^1_{y[t]}{}' \to \mathbf{b}^1_{y[t+1]})(1-\alpha(\mathbf{d})c) \qquad (3.20) \\
&\quad = (1-\alpha(\mathbf{d}))W_c(\mathbf{d}^1; \mathbf{x}[t], y[t], y[t+1])(1-\alpha(\mathbf{d})c) \\
&\quad \geq (1-\alpha(\mathbf{d}))W_c(\mathbf{d}^1; \mathbf{x}[t], y[t], y[t+1])(1-\alpha(\mathbf{d}))^c \\
&\quad = W_c(\mathbf{d}^1; \mathbf{x}[t], y[t], y[t+1])(1-\alpha(\mathbf{d}))^{1+c}.
\end{aligned}
$$

Note that the inequality in (3.18) follows from property 3, (3.19) follows from property 1, and (3.20) follows from property 2 of commission costs from Section 3.1.1. Accumulated over n investment periods, we have that

$$
W_c(\mathbf{d}; \mathbf{x}^n, y^{n+1}) \;\; \geq \;\; W_c(\mathbf{d}^1; \mathbf{x}^n, y^{n+1})(1-\alpha(\mathbf{d}))^{n(1+c)}, \qquad (3.21)
$$

where these expressions for wealth include the cost of rebalancing after the final investment period.

Now, from property 3 in Section 3.1.1, we can infer that

$$
W_c(\hat{\mathbf{b}}; \mathbf{x}^n, y^{n+1}) \geq \int_{\mathbf{d}\in\Delta_m^K} W_c(\mathbf{d}; \mathbf{x}^n, y^{n+1})\, \mathrm{d}\mu(\mathbf{d}).
$$

Since the distribution $\mu(\mathbf{d})$ is taken to be uniform over Δ_m^K, we have that

$$
W_c(\hat{\mathbf{b}}; \mathbf{x}^n, y^{n+1}) \geq \frac{1}{V(\Delta_m^K)} \int_{\mathbf{d}\in\Delta_m^K} W_c(\mathbf{d}; \mathbf{x}^n, y^{n+1})\, \mathrm{d}\mathbf{d},
$$

where $V(\Delta_m^K)$ is the volume of the set Δ_m^K and the notation $\int(\cdot)\,\mathrm{d}\mathbf{d}$ indicates a

hypersurface integral over the $K(m-1)$ dimensional space Δ_m^K. By applying the inequality in (3.21), we have that

$$\begin{aligned}
&\frac{1}{V(\Delta_m^K)} \int_{\mathbf{d}\in\Delta_m^K} W_c(\mathbf{d}; \mathbf{x}^n, y^{n+1})\, \mathrm{d}\mathbf{d} \\
&\qquad \geq \frac{1}{V(\Delta_m^K)} \int_{\mathbf{d}\in\Delta_m^K} W_c(\mathbf{d}^1; \mathbf{x}^n, y^{n+1})(1-\alpha(\mathbf{d}))^{n(1+c)}\, \mathrm{d}\mathbf{d} \\
&\qquad = \frac{W_c(\mathbf{d}^1; \mathbf{x}^n, y^{n+1})}{V(\Delta_m^K)} \int_{\mathbf{d}\in\Delta_m^K} (1-\alpha(\mathbf{d}))^{n(1+c)}\, \mathrm{d}\mathbf{d}. \qquad (3.22)
\end{aligned}$$

We now more carefully consider the choice of the function $\alpha(\mathbf{d})$. One valid possibility here is to choose $\alpha(\mathbf{d}) = 1$. However, this choice leads to a trivial bound. Another valid choice is to choose $\alpha(\mathbf{d}) = \bar{\alpha}$ for the values of $\mathbf{d}$ that admit such a value of $\alpha(\cdot)$, given the constraints, and choose $\alpha(\mathbf{d}) = 1$ for the rest of Δ_m^K. This will give a more meaningful bound. However, in order the achieve the best possible bound from (3.22), we would like to choose the smallest possible function $\alpha(\mathbf{d})$. Specifically, we will choose

$$\alpha(\mathbf{d}) = \min\left\{\alpha : \mathbf{d} = (1-\alpha)\mathbf{d}^1 + \alpha\mathbf{d}^2 \text{ for some } \mathbf{d}^2 \in \Delta_m^K\right\}.$$

We note that the sublevel sets of this particular $\alpha(\mathbf{d})$ are shrunken simplices, i.e., $\{\mathbf{d} : \alpha(\mathbf{d}) \leq \bar{\alpha}\}$ is a simplex of volume $V(\bar{\alpha}) = V(\Delta_m^K)\bar{\alpha}^{K(m-1)}$. Under this choice of $\alpha(\mathbf{d})$, we turn (3.22) into

$$\begin{aligned}
W_c(\hat{\mathbf{b}}; \mathbf{x}^n, y^{n+1}) &\geq \frac{W_c(\mathbf{d}^1; \mathbf{x}^n, y^{n+1})}{V(\Delta_m^K)} \int_0^1 (1-\alpha)^{n(1+c)} \left(\frac{\mathrm{d}}{\mathrm{d}\alpha} V(\alpha)\right) \mathrm{d}\alpha \\
&= \frac{W_c(\mathbf{d}^1; \mathbf{x}^n, y^{n+1})}{V(\Delta_m^K)} V(\Delta_m^K) K(m-1) \\
&\qquad \times \int_0^1 (1-\alpha)^{n(1+c)} \alpha^{(K(m-1)-1)}\, \mathrm{d}\alpha \\
&= W_c(\mathbf{d}^1; \mathbf{x}^n, y^{n+1}) K(m-1) \\
&\qquad \times \int_0^1 (1-\alpha)^{n(1+c)} \alpha^{(K(m-1)-1)}\, \mathrm{d}\alpha. \qquad (3.23)
\end{aligned}$$

It is now possible to simplify this by use of the beta function, which is given by

$$B(x,y) \triangleq \int_0^1 (1-\alpha)^{y-1} \alpha^{x-1}\, \mathrm{d}\alpha.$$

We will be interested in the cases where $x \geq 1$ and $y \geq 1$, for integer valued

x and real valued y. The beta function can also be written in terms of the gamma function $\Gamma(\cdot)$, where we have that

$$B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}.$$

However, note that $\Gamma(x) = (x-1)! = \frac{x!}{x}$ for integers $x \geq 1$. We also have that

$$\begin{aligned}
\Gamma(x+y) &= (x+y-1)\Gamma(x+y-1) \\
&= (x+y-1)(x+y-2)\Gamma(x+y-2) \\
&= ... \\
&= \left(\prod_{j=1}^{x}(x+y-j)\right)\Gamma(x+y-x) \\
&= \Gamma(y)\left(\prod_{j=1}^{x}(y-1+j)\right).
\end{aligned}$$

Putting this together, we have that

$$\begin{aligned}
B(x,y) &= \frac{x!\Gamma(y)}{x\Gamma(y)\left(\prod_{j=1}^{x}(y-1+j)\right)} \\
&= \frac{1}{x}\frac{x!}{\left(\prod_{j=1}^{x}(y-1+j)\right)} \\
&= \frac{1}{x\left(\prod_{j=1}^{x}\frac{y-1+j}{j}\right)}.
\end{aligned} \tag{3.24}$$

However, for $j \geq 1$, note that

$$\begin{aligned}
y \geq 1 &\implies y-1 \geq 0 \\
&\implies (y-1)(j-1) \geq 0 \\
&\implies 1-y-j+jy \geq 0 \\
&\implies y-1+j \leq jy \\
&\implies \frac{y-1+j}{j} \leq y.
\end{aligned}$$

We also have that $0 < \frac{y-1+j}{j}$, and therefore we have that

$$\prod_{j=1}^{x} \frac{y-1+j}{j} \leq \prod_{j=1}^{x} y = y^x.$$

Combining this inequality with Equation (3.24), we have that

$$B(x,y) \geq \frac{y^{-x}}{x}.$$

With $x = K(m-1)$ and $y = n(1+c)+1$, we have that

$$\int_0^1 (1-\alpha)^{n(1+c)} \alpha^{(K(m-1)-1)} \, \mathrm{d}\alpha = \frac{(n(1+c)+1)^{-K(m-1)}}{K(m-1)}.$$

By substitution into (3.23), we have that

$$\begin{aligned} W_c(\hat{\mathbf{b}}; \mathbf{x}^n, y^{n+1}) &\geq W_c(\mathbf{d}^1; \mathbf{x}^n, y^{n+1}) K(m-1) \\ &\qquad \times \frac{(n(1+c)+1)^{-K(m-1)}}{K(m-1)} \\ &= W_c(\mathbf{d}^1; \mathbf{x}^n, y^{n+1}) (n(1+c)+1)^{-K(m-1)}. \end{aligned}$$

Equivalently, we have that

$$\frac{W_c(\mathbf{d}^1; \mathbf{x}^n, y^{n+1})}{W_c(\hat{\mathbf{b}}; \mathbf{x}^n, y^{n+1})} \leq (n(1+c)+1)^{K(m-1)},$$

or

$$\begin{aligned} \frac{1}{n} \ln W_c(\mathbf{d}^1; \mathbf{x}^n, y^{n+1}) - \frac{1}{n} \ln W_c(\hat{\mathbf{b}}; \mathbf{x}^n, y^{n+1}) \\ \leq \frac{1}{n} K(m-1) \ln(n(1+c)+1). \end{aligned}$$

Since this bound holds for any $\mathbf{d}^1 \in \Delta_m^K$, we have that the bound holds for the best performing $\mathbf{d}^1 \in \Delta_m^K$. Hence, we have that

$$\begin{aligned} \sup_{\mathbf{d} \in \mathcal{D}} \frac{1}{n} \ln W_c(\mathbf{d}; \mathbf{x}^n, y^{n+1}) - \frac{1}{n} \ln W_c(\hat{\mathbf{b}}; \mathbf{x}^n, y^{n+1}) \\ \leq \frac{1}{n} K(m-1) \ln (n(1+c)+1). \end{aligned}$$ ■

3.4 Computation by Factor Graphs for Portfolios with Side Information

As presented in the previous section, our portfolio that uses side information for investing in a market with transaction costs, which is summarized in (3.17), is exceedingly expensive to compute. In particular, the portfolio under transaction costs involves the evaluation of the expectation in (3.5), which requires integration over Δ_m^K, and the direct evaluation of this integral is exponential in K for both storage and computation. For this reason, we propose using factor graphs and a sum-product algorithm [1] as one method of approximating our algorithm.

Graphical representations are not new in universal portfolios and universal prediction. Examples include the transition diagrams used in [38, 39, 55] and the tree representations used in [36, 37]. However, the authors are not aware of any previous explicit use of factor graphs and sum-product algorithms for universal portfolios.

3.4.1 Factor Graph for the Universal Portfolio with Side Information

We begin this development by demonstrating the factor graph concept on the instance of the side information portfolio from [47] defined by (3.5)-(3.9). First, we note that

$$\begin{aligned} P_t(\mathbf{d}) &= \frac{1}{Z_t} \prod_{\tau=1}^{t} f_\tau(\mathbf{d}) \\ &= \frac{1}{Z_t} \left(\prod_{y=1}^{K} \mu(\mathbf{b}_y) \right) \left(\prod_{\tau=1}^{t-1} \mathbf{b}_{y[\tau]}^T \mathbf{x}[\tau] \right). \end{aligned}$$

Upon regrouping the factors in the same manner as (3.12)-(3.13), we see that

$$\begin{aligned} P_t(\mathbf{d}) &= \frac{1}{Z_t} \prod_{y=1}^{K} \left(\mu(\mathbf{b}_y) \prod_{\tau \in \mathcal{T}_y^t} \mathbf{b}_y^T \mathbf{x}[\tau] \right) \\ &= \frac{1}{Z_t} \prod_{y=1}^{K} \left(W^y(\mathbf{b}_y; \mathbf{x}^{(t-1)}, y^t) \mu(\mathbf{b}_y) \right). \end{aligned}$$

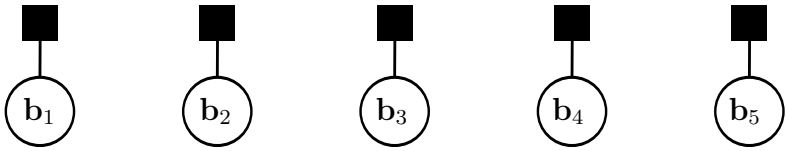


Figure 3.1: Factor graph for portfolio with side information of $K = 5$.

It is now obvious that $P_t(\mathbf{d})$ may be represented by a factor graph (forest) where the variable nodes are the portfolio vectors $\mathbf{b}_y$ for $y = 1, ..., K$. An example is show in Fig. 3.1, where in the depicted case we have that $K = 5$.

Note that due to the form of $g_t(\mathbf{d})$, we have that $\hat{\mathbf{b}}[t] = \mathrm{E}_{P_t(\mathbf{d})}[\mathbf{b}_{y[t]}]$. It is then clear by inspection of the factor graph that the algorithm reduces to that of (3.3), where this expectation only needs to be taken with respect to the following marginal distribution:

$$P_t(\mathbf{b}_{y[t]}) = \frac{W^{y[t]}(\mathbf{b}_{y[t]}; \mathbf{x}^{(t-1)}, y^t)\mu(\mathbf{b}_{y[t]})}{\int_{\mathbf{b}_{y[t]} \in \Delta_m} W^{y[t]}(\mathbf{b}_{y[t]}; \mathbf{x}^{(t-1)}, y^t)\, \mathrm{d}\mu(\mathbf{b}_{y[t]})}.$$

3.4.2 Factor Graph for the Universal Portfolio with Side Information and Transaction Costs

Similar derivations can be used to show that the algorithm we propose in Section 3.2 results in distributions $P_t(\mathbf{d})$ of the following form:

$$P_t(\mathbf{d}) \;=\; \frac{1}{Z_t} \prod_{i=1}^{K} \left(h_i(\mathbf{b}_i) \prod_{j=i+1}^{K} h_{(i,j)}(\mathbf{b}_i, \mathbf{b}_j) \right),$$

where the factors $h_i(\mathbf{b}_i)$ are further composed of factors either of the form $\mu(\mathbf{b}_i)$, $\mathbf{b}_i^T \mathbf{x}$, or $C_\tau(\mathbf{b}'_i \to \mathbf{b}_i)$, and the factors $h_{(i,j)}(\mathbf{b}_i, \mathbf{b}_j)$ are further composed of factors either of the form $C_\tau(\mathbf{b}'_i \to \mathbf{b}_j)$ or $C_\tau(\mathbf{b}'_j \to \mathbf{b}_i)$. Hence, we have that $P_t(\mathbf{d})$ may be represented by such a factor graph as we show in Fig. 3.2, where in the depicted case we have that $K = 5$.

In order to make use of the graph, we note that, as in the previous subsection, our portfolio for an investment period is $\hat{\mathbf{b}}[t] = \mathrm{E}_{P_t(\mathbf{d})}[\mathbf{b}_{y[t]}]$. Hence, we only need to compute the expectation with respect to the marginal distribution over $\mathbf{b}_{y[t]}$. Since the factor graph we have constructed has many cycles (for $K > 2$), it is not possible to compute the marginal distribution exactly

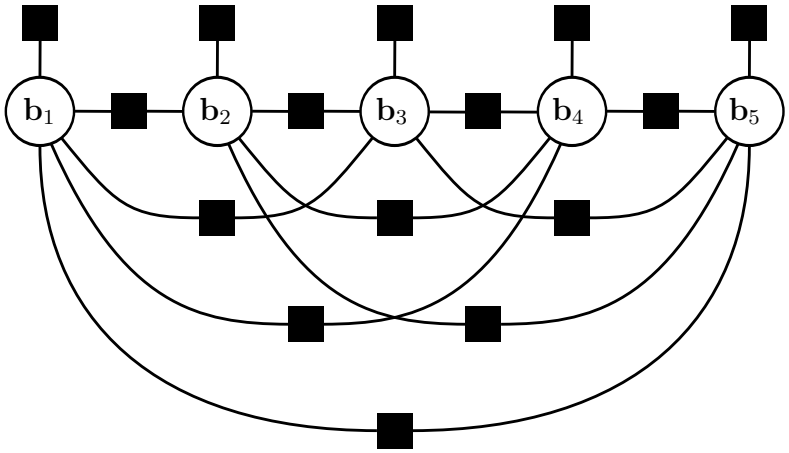


Figure 3.2: Factor graph for portfolio with side information of $K = 5$ and transaction costs.

with message passing. However, we have found through simulations that our message passing (sum-product) algorithm works well on our data sets.

First of all, we choose the distribution $\mu(\cdot)$ to be uniform. We then begin with the first investment period by initializing all of the messages over the edges of the graph to be the uniform distribution, i.e., constant functions over the simplex Δ_m. Note that with this initialization the messages correspond with exact function summaries for the graph of $P_1(\mathbf{d})$, up to a constant scale factor.

Now, let us suppose that we have the factor graph and messages from the previous distribution $P_{t-1}(\mathbf{d})$, and that these messages are close to exact function summaries on $P_{t-1}(\mathbf{d})$. For $P_t(\mathbf{d})$, we first initialize the messages in the new factor graph with the messages from $P_{t-1}(\mathbf{d})$. We should expect that this is a good initialization because the two factor graphs differ only in 1 or 2 out of the $\frac{1}{2}K(K+1)$ function nodes, depending on whether $y[t-1] = y[t]$.

In order to carry out message passing, we begin by queueing the messages going outward from the 1 or 2 function nodes that differ between $P_{t-1}(\mathbf{d})$ and $P_t(\mathbf{d})$. This is a queue of pending message updates. Now that there is at least one message on the queue, we proceed with a serial message passing schedule:

1) Begin at the front of the queue.

2) Update the pending message.

3) Suppose the message just updated goes along the edge from node A to node B. Queue up all of the messages going out of node B, except the one

going back to node A. Limit the total queue size to P message passes, or create some other stopping criterion.

4) If there are more messages on the queue, move on to the next one and go back to step 2.

In our simulations, we allow a queue size on the order of K^2 message passes. This is natural since there are K^2 edges in the graph. In particular, we have experimentally determined that $P = 3K^2$ gives good results. The final algorithm has run time and storage complexity polynomial in K per investment period.

3.5 Universal Switching Portfolios under Transaction Costs

In this section, we begin development of another portfolio that, like the portfolio presented in the preceding sections, is based on the methods from Cover's universal portfolio [43] and the transaction costs portfolio of Blum and Kalai [40]. In this case, we are extending the switching portfolio from [35] so that we can account for transaction costs in a way that is an improvement over the switching portfolio under transaction costs from [38].

To begin with, we define the set of switching strategies we would like our algorithm to compete with. First of all, a given switching strategy for a market sequence consisting of n trading periods will partition the sequence into a number $k \in \{1, ..., n\}$ of contiguous segments, such as

$$\{\mathbf{x}[\tau_1], ..., \mathbf{x}[\tau_2 - 1]\}...\{\mathbf{x}[\tau_k], ..., \mathbf{x}[n]\},$$

with $\tau_1 = 1$ and $\tau_i < \tau_{i+1}$. We call this a transition path, and note that there are $\sum_{k=0}^{n-1} \binom{n-1}{k} = 2^{n-1}$ such paths. Next, a given switching strategy must assign a portfolio for use in each segment. Hence, the switching strategy is defined by the parameters $\mathbf{s} = \{\tau_1, ..., \tau_k, \mathbf{b}_1, ..., \mathbf{b}_k\} \in \mathcal{S}_n$, with each $\mathbf{b}_i \in \Delta_m$. For simplicity, we define $\mathbf{d}_k = [\mathbf{b}_1, ..., \mathbf{b}_k] \in \Delta_m^k$, and the transition path is given by $\mathcal{T} = \{\tau_1, ..., \tau_k\}$.

3.5.1 Algorithm and Bound

In the fashion of previous universal portfolios, we will now specify how we will distribute our wealth among all the possible switching strategies in $\mathcal{S}_n$, in order to then "run them in parallel." To do this, we will first distribute wealth among the transition paths, then define how the wealth given to a particular transition path is distributed among the portfolios $\mathbf{d}_k \in \Delta_m^k$. To this end, we use the Krichevsky-Trofimov weighting $P(\mathcal{T})$ [58] to divide the wealth among the transition paths. This distribution is defined as follows:

$$P(\mathcal{T}) = \prod_{i=1}^{k} \left(P_{\tau_i}(\tau_i|\tau_{i-1}) \prod_{t=\tau_i+1}^{\tau_{i+1}-1} P_t(\tau_i|\tau_i) \right),$$

where $P_t(j|j) = \frac{t-j-.5}{t-j}$ and $P_t(t|j) = \frac{.5}{t-j}$ for $j = 1, ..., t-1$ and $t \geq 2$. For notational convenience, we have defined $\tau_0 = 0$, $\tau_{k+1} = n+1$, and $P_{\tau_1}(\tau_1|\tau_0) = P_1(1|0) = 1$. Within a particular transition path, the assigned wealth is distributed uniformly over Δ_m^k. We will use the notation $u(\mathbf{d}|\mathcal{T})$ to indicate a uniform distribution over a space of appropriate dimension. Let us denote by $\mu(\mathbf{s}) = P(\mathcal{T})u(\mathbf{d}|\mathcal{T})$ the overall mixed joint distribution over the set of switching strategies $\mathcal{S}_n$, such that $\int_{\mathbf{s}} \mathrm{d}\mu(\mathbf{s}) = 1$ and the dimension of $\mathbf{d}$ agrees with the number of partitions in $\mathcal{T}$. Then our portfolio is given in the usual form as

$$\hat{\mathbf{b}}[t] = \frac{\int_{\mathbf{s}\in\mathcal{S}_n} \mathbf{b}_{\mathbf{s}}[t] W_c(\mathbf{s}; \mathbf{x}^{t-1}) \, \mathrm{d}\mu(\mathbf{s})}{\int_{\mathbf{s}\in\mathcal{S}_n} W_c(\mathbf{s}; \mathbf{x}^{t-1}) \, \mathrm{d}\mu(\mathbf{s})}, \tag{3.25}$$

where $\mathbf{b}_{\mathbf{s}}[t]$ is the portfolio that strategy $\mathbf{s}$ would assign during trading period t, and $W_c(\cdot)$ includes the cost of rebalancing in preparation for trading period t. More specifically, we can write (3.25) as

$$\hat{\mathbf{b}}[t] = \frac{\sum_{\mathcal{T}} P(\mathcal{T}) \int_{\mathbf{d}} \mathbf{b}_{\{\mathcal{T},\mathbf{d}\}}[t] W_c(\{\mathcal{T}, \mathbf{d}\}; \mathbf{x}^{t-1}) \, \mathrm{d}u(\mathbf{d}|\mathcal{T})}{\sum_{\mathcal{T}} P(\mathcal{T}) \int_{\mathbf{d}} W_c(\{\mathcal{T}, \mathbf{d}\}; \mathbf{x}^{t-1}) \, \mathrm{d}u(\mathbf{d}|\mathcal{T})}. \tag{3.26}$$

It is possible to show that the wealth achieved by this portfolio algorithm under transaction costs satisfies the inequality in the following theorem:

Theorem 2: *We are able to construct a sequential portfolio* $\hat{\mathbf{b}}$ *such that*

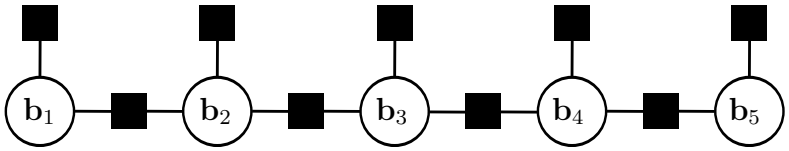


Figure 3.3: Factor graph for a transition path with $k = 5$ segments and transaction costs.

the wealth achieved by $\hat{\mathbf{b}}$ *satisfies*

$$\sup_{\mathbf{s}\in\mathcal{D}_k} \frac{1}{n} \ln W_c(\mathbf{s}; \mathbf{x}^n) - \frac{1}{n} \ln W_c(\hat{\mathbf{b}}; \mathbf{x}^n) \leq \frac{1}{n} k(m-1) \ln\left(n(1+c)+1\right) + \frac{(3k-2)}{2} \frac{\ln n}{n} + O\left(\frac{k}{n}\right).$$

Here, $\mathcal{D}_k$ *is the set of all possible switching strategies that partition the sequence of price relatives into* k *non-empty contiguous segments and assign a CRP to each segment. The parameter* c *defines the level of transaction costs as in [40].*

Hence, our algorithm is universal with respect to any particular k. We note that the algorithm presented in [38] is also universal with respect to the same class of strategies. However, the algorithm presented here is built from fundamentally different concepts and has a different performance bound.

3.5.2 Proof of the Bound

First, property 3 of transaction costs in Section 3.1.1 allows us to say that the wealth achieved by the algorithm is at least as much as the expected achieved wealth over all switching strategies, with respect to the initial distribution of wealth:

$$W_c(\hat{\mathbf{b}}; \mathbf{x}^n) \geq \sum_{\mathcal{T}} P(\mathcal{T}) \int_{\mathbf{d}_\mathcal{T}} W_c(\mathbf{d}_\mathcal{T}; \mathbf{x}^n) \, \mathrm{d}u(\mathbf{d}_\mathcal{T}|\mathcal{T}).$$

If we choose any single transition path with k partitions of the market sequence, then we have that

$$W_c(\hat{\mathbf{b}}; \mathbf{x}^n) \geq P(\mathcal{T}_k) \int_{\mathbf{d}_{\mathcal{T}_k}} W_c(\mathbf{d}_{\mathcal{T}_k}; \mathbf{x}^n) \, \mathrm{d}u(\mathbf{d}_{\mathcal{T}_k}|\mathcal{T}_k).$$

By using the steps from the proof of Theorem 1, we have that

$$\int_{\mathbf{d}_{\mathcal{T}_k}} W_c(\mathbf{d}_{\mathcal{T}_k};\mathbf{x}^n)\,\mathrm{d}u(\mathbf{d}_{\mathcal{T}_k}|\mathcal{T}_k) \geq W_c(\bar{\mathbf{d}}_{\mathcal{T}_k};\mathbf{x}^n)(n(1+c)+1)^{-k(m-1)}$$

for any $\bar{\mathbf{d}}_{\mathcal{T}_k}$ in Δ_m^k and for any transition path $\mathcal{T}_k$ with k partitions of the market sequence. Furthermore, by use of the Krichevski-Trofimov distribution over transition paths, we have that

$$-\frac{1}{n}\ln P(\mathcal{T}_k) \leq \frac{(3k-2)}{2}\frac{\ln n}{n} + O\left(\frac{k}{n}\right).$$

Together, this shows that

$$\begin{aligned}\frac{1}{n}\ln W_c(\bar{\mathbf{d}}_{\mathcal{T}_k};\mathbf{x}^n) - \frac{1}{n}\ln W_c(\hat{\mathbf{b}};\mathbf{x}^n)& \\ \leq \frac{1}{n}k(m-1)\ln\left(n(1+c)+1\right) + \frac{(3k-2)}{2}\frac{\ln n}{n} + O\left(\frac{k}{n}\right).&\end{aligned}$$

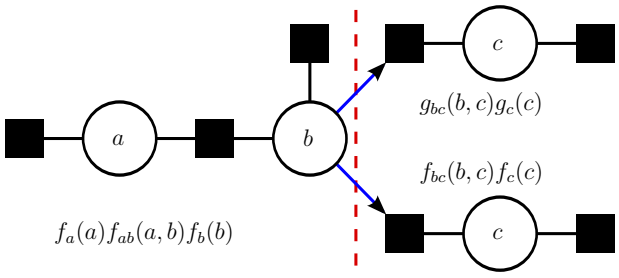
Since this bound holds for any transition path $\mathcal{T}_k$ with k partitions of the market sequence and for any $\bar{\mathbf{d}}_{\mathcal{T}_k}$ in Δ_m^k, it must hold for the best transition path of k partitions and best assignment of portfolios to each segment. Hence,

$$\begin{aligned}\sup_{\mathbf{s}\in\mathcal{D}_k}\frac{1}{n}\ln W_c(\mathbf{s};\mathbf{x}^n) - \frac{1}{n}\ln W_c(\hat{\mathbf{b}};\mathbf{x}^n)& \\ \leq \quad \frac{1}{n}k(m-1)\ln\left(n(1+c)+1\right) + \frac{(3k-2)}{2}\frac{\ln n}{n} + O\left(\frac{k}{n}\right).&\end{aligned}$$

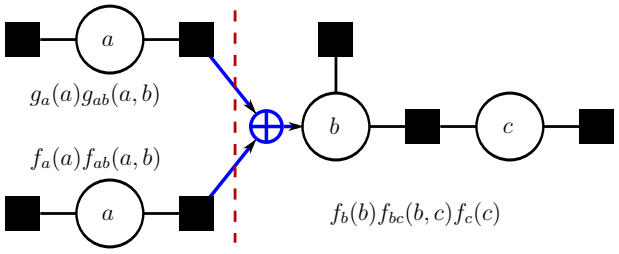
■

3.6 Implementation of the Switching Portfolio

Consider again the specification of our switching algorithm given in (3.26). Clearly, the direct computation of $\hat{\mathbf{b}}[t]$ from this is prohibitively complex, as it involves 2^{t-1} integrals over spaces with dimensionality as high as Δ_m^t. However, the structure of $W(\{\mathcal{T},\mathbf{d}\};\mathbf{x}^{t-1})$, which can be represented as a factor graph, and the choice of the Krichevsky-Trofimov weighting allow us to achieve great reduction in complexity.



(a) Graphs overlap at the left.



(b) Graphs overlap at the right.

Figure 3.4: (a) Two factor graphs share factors left of the dashed line. Arrows crossing the dashed line indicate that the same message is sent to the non-overlapping portions of the graphs. (b) Two factor graphs share factors right of the dashed line. Messages crossing the dashed line are summed and sent to the shared factors to complete the message passing.

3.6.1 Simplifying Properties

Consider first the integration of a single term from the numerator (one particular transition path) of (3.26), i.e.,

$$\int_{\mathbf{d}\in\Delta_m^k} \mathbf{b}_{\{\mathcal{T},\mathbf{d}\}}[t] W_c(\{\mathcal{T},\mathbf{d}\};\mathbf{x}^{t-1})\,\mathrm{d}\mathbf{d}.$$

(We are ignoring the distribution $u(\mathbf{d}|\mathcal{T})$ for the moment, which essentially amounts to scaling by some constant.) The key observation to point out here is that $W(\{\mathcal{T},\mathbf{d}\};\mathbf{x}^{t-1})$ factors into an expression of the form

$$\prod_{i=1}^{k} f_i(\mathbf{b}_i) \prod_{i=2}^{k} C_{\tau_i-1}(\mathbf{b}_{i-1}\to\mathbf{b}_i). \tag{3.27}$$

Furthermore, we have that $\mathbf{b}_{\{\mathcal{T},\mathbf{d}\}}[t] = \mathbf{b}_k$. Hence, we see that we can use message passing on a factor graph derived from (3.27) to compute a marginal function $f(\mathbf{b}_k)$. In Fig. 3.3, we show such a factor graph where $k = 5$. Once we have this marginal function, it is a much simpler matter to compute the

following integral:

$$\int_{\Delta_m} \mathbf{b}_k f(\mathbf{b}_k)\, \mathrm{d}\mathbf{b}_k.$$

However, it is still an overwhelming task to perform message passing on 2^{t-1} factor graphs during each trading period. Fortunately, there are some basic simplifications we can utilize in order to avoid redundant calculations, stemming from the overlap present between many of the graphs. To see how we use the overlap between graphs, consider that we have the following two functions:

$$\begin{aligned} f(a,b,c) &= f_a(a) f_{ab}(a,b) f_b(b) f_{bc}(b,c) f_c(c), \\ g(a,b,c) &= g_a(a) g_{ab}(a,b) g_b(b) g_{bc}(b,c) g_c(c). \end{aligned}$$

Suppose now that $f_a = g_a$, $f_{ab} = g_{ab}$, and $f_b = g_b$ and we would like to compute the marginal function

$$h(c) = \int_b \int_a \left(f(a,b,c) + g(a,b,c) \right)\, \mathrm{d}a\, \mathrm{d}b. \tag{3.28}$$

Rather than making use of message passing on f and g separately, we can instead factor out the overlapping part, giving us

$$\begin{aligned} h_b(b) &\triangleq f_b(b) \int_a f_a(a) f_{ab}(a,b)\, \mathrm{d}a, \\ h(c) &= f_c(c) \int_b h_b(b) f_{bc}(b,c)\, \mathrm{d}b + g_c(c) \int_b h_b(b) g_{bc}(b,c)\, \mathrm{d}b. \end{aligned}$$

This procedure is shown graphically in Fig. 3.4(a). Message passing on the common factors is performed once, from left to right. This resulting message is then sent to the parts of the graphs that differ. Finally, the full marginal can be found by summing the separate marginals.

Suppose instead that $f_b = g_b$, $f_{bc} = g_{bc}$, and $f_c = g_c$, and again we would like to compute $h(c)$ as in (3.28). In this case, we would get

$$\begin{aligned} h_f(b) &\triangleq \int_a f_a(a) f_{ab}(a,b)\, \mathrm{d}a, \\ h_g(b) &\triangleq \int_a g_a(a) g_{ab}(a,b)\, \mathrm{d}a, \\ h(c) &= f_c(c) \int_b \left[h_h(b) + h_g(b) \right] f_b(b) f_{bc}(b,c)\, \mathrm{d}b. \end{aligned}$$

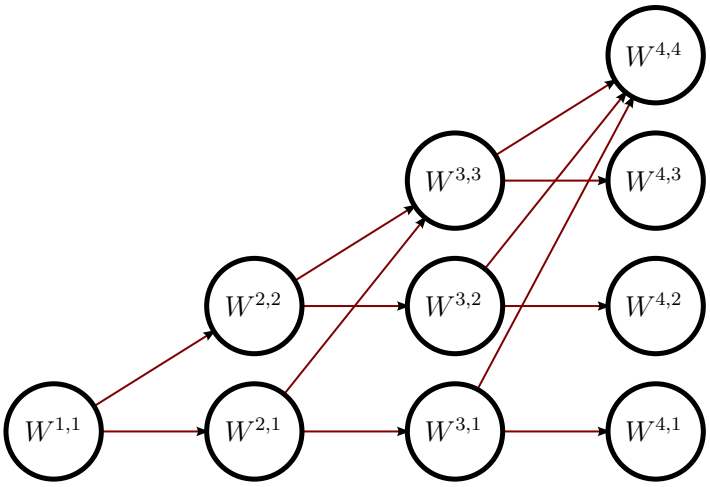


Figure 3.5: To compute $\mathbf{b}[t]$, we have 2^{t-1} factor graphs folded onto each other into a transition diagram. In this case, $2^3 = 8$ parallel factor graphs are compactly represented.

This procedure is shown graphically in Fig. 3.4(b). Message passing from the left to the right is performed separately at first. Then, upon reaching the parts of the graphs that are shared, the messages from the differing parts are combined, and message passing continues as though for a single factor graph.

By combining such simplifications from the overlap of the 2^{t-1} factor graphs needed for the computation of $\mathbf{b}[t]$, we find that the graphs from all of the transition paths can be folded onto each other and compactly represented by a transition diagram. An example of such a diagram is given in Fig. 3.5. However, this is not the final simplification. Due to the structure of transition paths and the Krichevsky-Trofimov weighting, the transition diagram for period t actually contains the transition diagram for trading period $t-1$. Hence, it is not necessary to perform all of the message passes through a new transition diagram for each period. Instead, it is possible to simply maintain the results from the previous day, then perform the few additional message passes to complete the new transition diagram.

The following is a more complete description of the algorithm:

1. Initialize:

$$W^{1,1}(\mathbf{b}) = 1 \quad \text{and} \quad t = 1.$$

2. Compute the portfolio for period t:

$$\hat{\mathbf{b}}[t] = \frac{1}{Z_t} \int_{\mathbf{b} \in \Delta_m} \mathbf{b} \sum_{j=1}^{t} W^{t,j}(\mathbf{b}) \, \mathrm{d}u(\mathbf{b}),$$

where Z_t is chosen so the elements of $\hat{\mathbf{b}}[t]$ sum to 1.

3. After period t, update for $j = 1, ..., t$:

$$W^{t,j}(\mathbf{b}) \leftarrow W^{t,j}(\mathbf{b})(\mathbf{b}^T \mathbf{x}[t]).$$

4. Update $t \leftarrow t + 1$.

5. Initialize, for $j = 1, ..., t - 1$:

$$W^{t,j}(\mathbf{b}) = W^{t-1,j}(\mathbf{b}) C_{t-1}(\mathbf{b}' \to \mathbf{b}) P_t(j|j).$$

6. Initialize:

$$W^{t,t}(\mathbf{b}) = \sum_{j=1}^{t-1} P_t(t|j) \int_{\mathbf{b}_2 \in \Delta_m} W^{t-1,j}(\mathbf{b}_2) C_{t-1}(\mathbf{b}_2' \to \mathbf{b}) \, \mathrm{d}u(\mathbf{b}_2).$$

(We now no longer need $W^{t-1,j}$, $j = 1, ..., t - 1$.)

7. Go to step 2.

The Krichevsky-Trofimov weighting comes from the factors

$$P_t(j|j) = \frac{t - j - .5}{t - j} \text{ and } P_t(t|j) = \frac{.5}{t - j}$$

for $j = 1, ..., t - 1$ and $t \geq 2$. The uniform distributions over $\mathbf{d}$ for each transition path comes from $u(\mathbf{b})$ in steps 2 and 6, i.e., the uniform distribution over Δ_m. The complexity that we achieve for calculating a portfolio $\hat{\mathbf{b}}[t]$ is only linear in t, which is a drastic improvement over evaluating (3.26) directly.

3.6.2 Proof of Correctness

In order to demonstrate the correctness of the steps 1 through 6 of the algorithm, first suppose we have that

$$W^{t,j}(\mathbf{b}) = \sum_{\mathcal{T} \in \mathcal{T}\{...,j\}} P(\mathcal{T}) \int W_c(..., \mathbf{b}_{k_\mathcal{T}-1}, \mathbf{b}; \mathbf{x}^{t-1}) \, \mathrm{d}u(\mathbf{b}_1) ... \, \mathrm{d}u(\mathbf{b}_{k_\mathcal{T}-1}), \tag{3.29}$$

where $\mathcal{T}\{..., j\}$ is the set of all transition paths with the last partition starting at time j, $P(\mathcal{T})$ is the Krichevsky-Trofimov weighting on transition path $\mathcal{T}$, and $W_c(...; \mathbf{x}^{t-1})$ is the wealth achieved through day $t-1$, including transaction costs to set up day t. Note that $\bigcup_{j=1}^{t} \mathcal{T}\{..., j\}$ constitutes the set of all transition paths. Furthermore, note that $W^{1,1}(\mathbf{b}) = 1$ satisfies this, as there is only one transition path to sum over, and there are no variables to integrate out. Hence, we have that $W^{1,1}(\mathbf{b}) = W_c(\mathbf{b}; \mathbf{x}^0)$, which is defined to be unity since no strategy could have gained or lost wealth before trading begins.[2]

Now considering day $t+1$, we have for $j = 1, ..., t$ that

$$W^{t+1,j}(\mathbf{b}) = \sum_{\mathcal{T} \in \mathcal{T}\{...,j\}} P(\mathcal{T}) \int W_c(..., \mathbf{b}_{k_\mathcal{T}-1}, \mathbf{b}; \mathbf{x}^{t}) \, \mathrm{d}u(\mathbf{b}_1) ... \, \mathrm{d}u(\mathbf{b}_{k_\mathcal{T}-1}), \tag{3.30}$$

which is simply a reiteration of (3.29) for day $t+1$. However, we can expand terms to get

$$P(\mathcal{T}) = P(\mathcal{T}[1:t]) P_{t+1}(j|j)$$

and

$$W_c(..., \mathbf{b}_{k_\mathcal{T}-1}, \mathbf{b}; \mathbf{x}^{t}) = W_c(..., \mathbf{b}_{k_\mathcal{T}-1}, \mathbf{b}; \mathbf{x}^{t-1})(\mathbf{b}^T \mathbf{x}[t]) C_t(\mathbf{b}' \to \mathbf{b}).$$

We have used the notation $\mathcal{T}[1:t]$ to indicate a truncated transition path that takes only the portion from day 1 to day t. Substituting into (3.30) and

[2] We assume there are no transaction costs for setting up the initial portfolio $\mathbf{b}[1]$.

rearranging terms, we get

$$
\begin{aligned}
&W^{t+1,j}(\mathbf{b}) \\
&\quad = P_{t+1}(j|j)(\mathbf{b}^T\mathbf{x}[t])C_t(\mathbf{b}' \to \mathbf{b}) \\
&\qquad \times \left(\sum_{\mathcal{T} \in \mathcal{T}\{...,j\}} P(\mathcal{T}[1:t]) \int W_c(..., \mathbf{b}_{k_\mathcal{T}-1}, \mathbf{b}; \mathbf{x}^{t-1}) \, \mathrm{d}u(\mathbf{b}_1) ... \, \mathrm{d}u(\mathbf{b}_{k_\mathcal{T}-1}) \right) \\
&\quad = P_{t+1}(j|j)(\mathbf{b}^T\mathbf{x}[t])C_t(\mathbf{b}' \to \mathbf{b}) W^{t,j}(\mathbf{b}).
\end{aligned}
$$

The final equality shows that the combined updates of steps 3 and 5 are correct.

Now, to verify step 6, we have that

$$
W^{t+1,t+1}(\mathbf{b}) = \sum_{\mathcal{T} \in \mathcal{T}\{...,t+1\}} P(\mathcal{T}) \int W_c(..., \mathbf{b}_{k_\mathcal{T}-1}, \mathbf{b}; \mathbf{x}^t) \, \mathrm{d}u(\mathbf{b}_1) ... \, \mathrm{d}u(\mathbf{b}_{k_\mathcal{T}-1}).
$$

We can then break up the sum into a double sum and expand some terms to get

$$
\begin{aligned}
&W^{t+1,t+1}(\mathbf{b}) \\
&\quad = \sum_{j=1}^{t} \sum_{\mathcal{T} \in \mathcal{T}\{...,j,t+1\}} P(\mathcal{T}[1:t]) P_{t+1}(t+1|j) \\
&\qquad \times \int W_c(..., \mathbf{b}_{k_\mathcal{T}-1}; \mathbf{x}^{t-1}) (\mathbf{b}_{k_\mathcal{T}-1}^T \mathbf{x}[t]) C_t(\mathbf{b}'_{k_\mathcal{T}-1} \to \mathbf{b}) \, \mathrm{d}u(\mathbf{b}_1) ... \, \mathrm{d}u(\mathbf{b}_{k_\mathcal{T}-1}).
\end{aligned}
$$

We then rearrange the locations of some of the factors and the inside summation to get

$$
\begin{aligned}
&W^{t+1,t+1}(\mathbf{b}) \\
&\quad = \sum_{j=1}^{t} P_{t+1}(t+1|j) \int_{\mathbf{b}_{k_\mathcal{T}-1} \in \Delta_m} (\mathbf{b}_{k_\mathcal{T}-1}^T \mathbf{x}[t]) C_t(\mathbf{b}'_{k_\mathcal{T}-1} \to \mathbf{b}) \\
&\qquad \times \left(\sum_{\mathcal{T} \in \mathcal{T}\{...,j,t+1\}} P(\mathcal{T}[1:t]) \right. \\
&\qquad\qquad \left. \times \int W_c(..., \mathbf{b}_{k_\mathcal{T}-1}; \mathbf{x}^{t-1}) \, \mathrm{d}u(\mathbf{b}_1) ... \, \mathrm{d}u(\mathbf{b}_{k_\mathcal{T}-2}) \right) \mathrm{d}u(\mathbf{b}_{k_\mathcal{T}-1}).
\end{aligned}
$$

At this point, we recognize the term in parentheses as $W^{t,j}(\mathbf{b}_{k_\mathcal{T}-1})$ from

(3.29), which gives us

$$W^{t+1,t+1}(\mathbf{b}) = \sum_{j=1}^{t} P_{t+1}(t+1|j) \\ \times \int_{\mathbf{b}_{k_{\mathcal{T}}-1} \in \Delta_m} (\mathbf{b}_{k_{\mathcal{T}}-1}^T \mathbf{x}[t]) C_t(\mathbf{b}'_{k_{\mathcal{T}}-1} \to \mathbf{b}) W^{t,j}(\mathbf{b}_{k_{\mathcal{T}}-1}) \, \mathrm{d}u(\mathbf{b}_{k_{\mathcal{T}}-1}).$$

This final equality is equivalent to the update in step 6 of the algorithm.

Lastly, we need to show that the computation of the portfolio in step 2 is correct. To this end, consider from (3.26) that

$$\hat{\mathbf{b}}[t] = \frac{1}{Z_t} \sum_{\mathcal{T}} P(\mathcal{T}) \int_{\mathbf{d}} \mathbf{b}_{\{\mathcal{T},\mathbf{d}\}}[t] W_c(\{\mathcal{T}, \mathbf{d}\}; \mathbf{x}^{t-1}) \, \mathrm{d}u(\mathbf{d}|\mathcal{T})$$

We can rewrite this as

$$\hat{\mathbf{b}}[t] = \frac{1}{Z_t} \sum_{\mathcal{T}} P(\mathcal{T}) \int \mathbf{b}_{k_{\mathcal{T}}} W_c(\mathbf{b}_1, ..., \mathbf{b}_{k_{\mathcal{T}}}; \mathbf{x}^{t-1}) \, \mathrm{d}u(\mathbf{b}_1) ... \, \mathrm{d}u(\mathbf{b}_{k_{\mathcal{T}}}).$$

We can then move the $\mathbf{b}_{k_{\mathcal{T}}}$ outside of all but the final integral to get

$$\hat{\mathbf{b}}[t] = \frac{1}{Z_t} \sum_{\mathcal{T}} P(\mathcal{T}) \\ \times \int_{\mathbf{b}_{k_{\mathcal{T}}}} \mathbf{b}_{k_{\mathcal{T}}} \left(\int W_c(\mathbf{b}_1, ..., \mathbf{b}_{k_{\mathcal{T}}}; \mathbf{x}^{t-1}) \, \mathrm{d}u(\mathbf{b}_1) ... \, \mathrm{d}u(\mathbf{b}_{k_{\mathcal{T}}-1}) \right) \mathrm{d}u(\mathbf{b}_{k_{\mathcal{T}}}).$$

Since $\mathbf{b}_{k_{\mathcal{T}}}$ is only a dummy variable used for the integration, we can replace it by $\mathbf{b}$ to get

$$\hat{\mathbf{b}}[t] = \frac{1}{Z_t} \sum_{\mathcal{T}} P(\mathcal{T}) \\ \times \int_{\mathbf{b}} \mathbf{b} \left(\int W_c(\mathbf{b}_1, ..., \mathbf{b}_{k_{\mathcal{T}}-1}, \mathbf{b}; \mathbf{x}^{t-1}) \, \mathrm{d}u(\mathbf{b}_1) ... \, \mathrm{d}u(\mathbf{b}_{k_{\mathcal{T}}-1}) \right) \mathrm{d}u(\mathbf{b}),$$

which removes the dependence on the transition path, and allows us to rearrange the equation to get

$$\hat{\mathbf{b}}[t] = \frac{1}{Z_t} \int_{\mathbf{b}} \mathbf{b} \sum_{\mathcal{T}} P(\mathcal{T}) \\ \times \left(\int W_c(\mathbf{b}_1, ..., \mathbf{b}_{k_{\mathcal{T}}-1}, \mathbf{b}; \mathbf{x}^{t-1}) \, \mathrm{d}u(\mathbf{b}_1) ... \, \mathrm{d}u(\mathbf{b}_{k_{\mathcal{T}}-1}) \right) \mathrm{d}u(\mathbf{b}).$$

Furthermore, we can split up the sum over transition paths into a double sum to get

$$\hat{\mathbf{b}}[t] = \frac{1}{Z_t} \int_{\mathbf{b}} \mathbf{b} \sum_{j=1}^{t} \Bigg[\sum_{\mathcal{T} \in \mathcal{T}\{...,j\}} P(\mathcal{T}) \times \left(\int W_c(\mathbf{b}_1, ..., \mathbf{b}_{k_{\mathcal{T}}-1}, \mathbf{b}; \mathbf{x}^{t-1}) \, \mathrm{d}u(\mathbf{b}_1) ... \, \mathrm{d}u(\mathbf{b}_{k_{\mathcal{T}}-1}) \right) \Bigg] \mathrm{d}u(\mathbf{b}).$$

We can then recognize the term in the square brackets as the expression in (3.29), which finally gives us step 2 of the algorithm:

$$\hat{\mathbf{b}}[t] = \frac{1}{Z_t} \int_{\mathbf{b} \in \Delta_m} \mathbf{b} \sum_{j=1}^{t} W^{t,j}(\mathbf{b}) \, \mathrm{d}u(\mathbf{b}).$$

3.7 Simulation Results

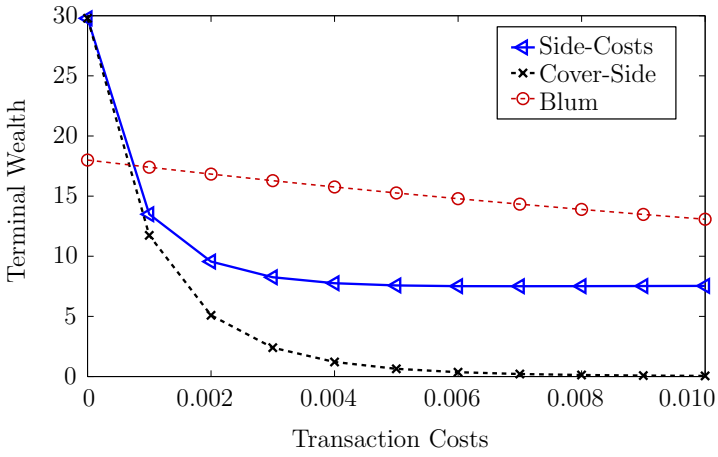
In this section, we examine results from simulations of our factor graph portfolio algorithms, and compare our algorithms with other portfolio algorithms. We use two separate data sets. The first, SET1, consists of historical stock prices collected from 34 stocks[3] in the New York Stock Exchange over a 22 year period until 1985.[4] This is the same data set used in [35], [39], [40], [43], [44], and others. The other data set, SET2, consists of data from 1996 until 2011 for 30 stocks from the S&P 500.

3.7.1 Simulation of the Side Information Portfolio

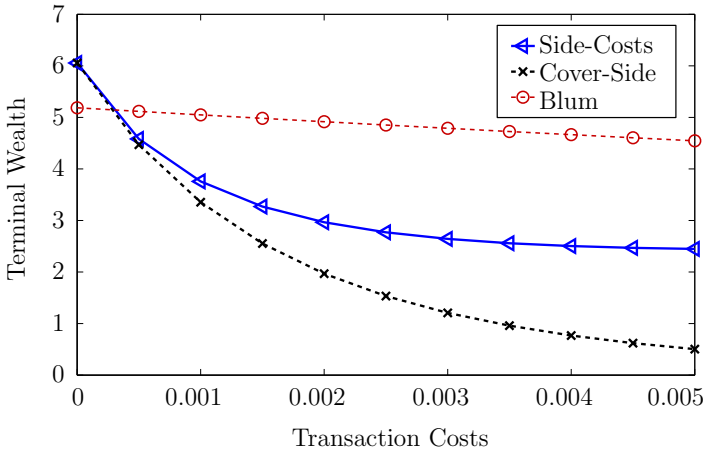
We first compare our portfolio with side information under transaction costs that uses the factor graph approximation (Side-Costs) to the algorithms of [47] (Cover-Side) and [40] (Blum). The simulations consist of portfolios containing $m = 2$ stocks and $K = 4$. The side information is formed by quantizing the price relative space $\mathbf{x}[t-1]$, i.e., $y[t] = q(\mathbf{x}[t-1])$ where we

[3] We have excluded the anomalous Kin-Ark data set from our simulations.

[4] We thank Dr. Erik Ordentlich for providing us with this historical data.



(a) SET1 Simulations.



(b) SET2 Simulations.

Figure 3.6: Terminal achieved wealth (averaged over all stock pairs) versus transaction costs c.

have that

$$q(\mathbf{x}) = \begin{cases} 1 & \\ 2 & \text{if} \\ 3 & \\ 4 & \end{cases} \begin{array}{l} x_1 \geq 1 \text{ and } x_2 \geq 1 \\ x_1 \geq 1 \text{ and } x_2 < 1 \\ x_1 < 1 \text{ and } x_2 \geq 1 \\ x_1 < 1 \text{ and } x_2 < 1 \end{array},$$

and we arbitrarily choose $y[1] = 1$. Each price relative represents one trading day.

In Fig. 3.6(a), we show results comparing the terminal wealths after the full 22 years of SET1, with 1 unit initially invested, of the three algorithms as a function of the parameter c of transaction costs. In particular, these terminal wealths are the averages of final achieved wealth for all $\binom{34}{2} = \frac{34\times 33}{2} = 561$

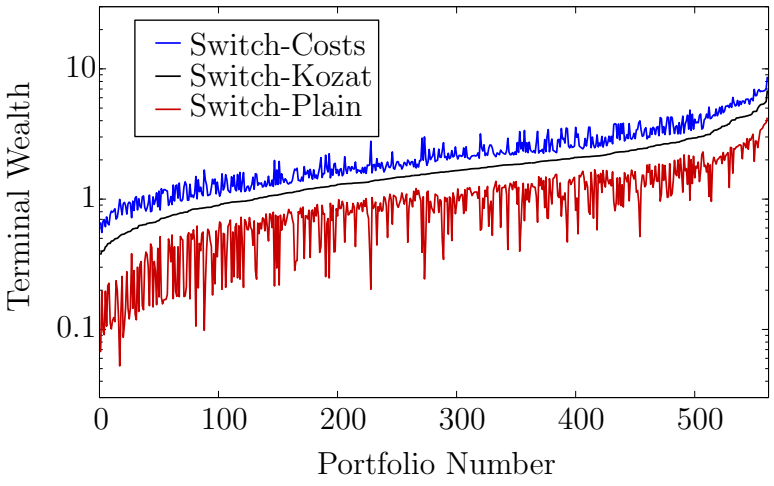
possible two-stock portfolios, and for a range of values for transaction costs. Similarly, in Fig. 3.6(b), we show the same types of results when the algorithms are used on SET2. In this case, the terminal wealths are the averages of final achieved wealth for all $\binom{30}{2} = \frac{30\times 29}{2} = 435$ possible two-stock portfolios. The graphs shows characteristics common to most of the simulation runs. For example, our algorithm ("Side-Costs" in the plot) consistently achieves equal or greater wealth than the side information algorithm of [47] ("Cover-Side") under all simulated values of the transaction costs parameter c. More specifically, Cover-Side achieves a terminal wealth that falls off nearly to zero as c increases, whereas our portfolio's wealth decreases to a certain point, and then after that point it essentially achieves the same terminal wealth. This is because the algorithm "learns" that rebalancing the portfolio, even based on the side information, is too costly, and the algorithm reverts to a buy-and-hold portfolio.

These figures also show a comparison with the transaction costs portfolio of [40] ("Blum" in the plot). We point out that, even though Blum performs consistently better than our algorithm for transaction costs above approximately $c = 0.001$ in Fig. 3.6(a), or $c = 0.0003$ in Fig. 3.6(b), it should be noted that, while the transaction costs are typically known ahead of time, it cannot be known a-priori whether the particular combination of stocks and side information will place us above or below the critical threshold of c that is observed between our algorithm or Blum having better performance. This would especially be true for institutional investors, who will typically pay some fraction of a cent per share in transaction cost. In particular, the performance of Side-Costs is generally no worse than a factor of approximately 3 below that of Blum in the cases where Blum outperforms the algorithm presented in this work. Furthermore, we note that, in the regime of low transaction costs, gains from side information are observed even though we only used the most basic possible side information source. Countless other sources of information could conceivably be quantized for use with the algorithm, such as trading volume, online news articles, or even opinions from human experts, and for a well chosen side information sequence, it may be possible for our algorithm to achieve wealth that is orders of magnitude above the portfolio of [40].

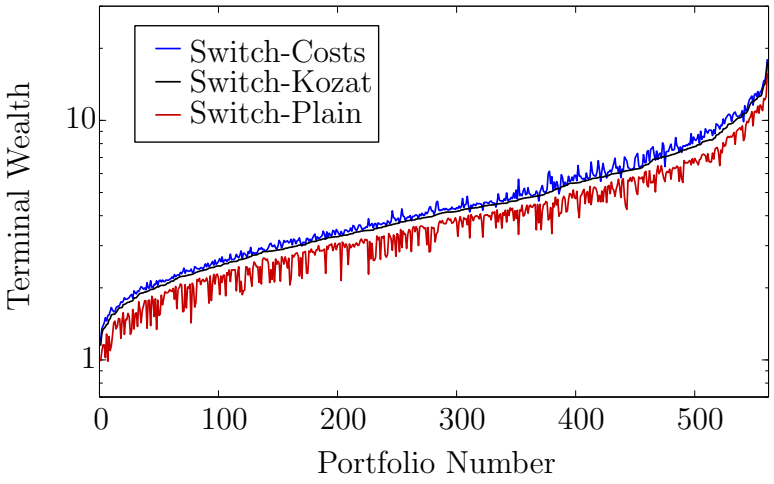
3.7.2 Simulation of the Switching Portfolio

We now compare our switching portfolio that takes into account transaction costs (Switch-Costs), with the switching portfolio from [38] that also takes transaction costs into account (Switch-Kozat), and the switching portfolio from [35] that does not account for such costs (Switch-Plain). In Fig. 3.7(a), we show the results of these simulations on SET1 with transaction costs of 10%. It consists of the wealth achieved by the end of the 22 years for each of $561 \times 3 = 1683$ separate algorithm runs. Transactions to reapportion the portfolio wealth occur only once per trading day. Each of these simulation runs is a two-stock portfolio, hence the $561 = \binom{34}{2}$, for the three algorithms being compared. The vertical axis indicates the wealth achieved by the end of the 22 year period per dollar of initial capital. The horizontal axis simply enumerates the set of two-stock portfolios, which have been sorted according to the wealth achieved by Switch-Kozat from [38]. (Note that such a sorting will result in a smoother curve for the algorithm from [38].) It can be seen that our algorithm consistently outperforms the other two algorithms in the given situation, since the values of final wealth are uniformly greater than the wealths from the other algorithms. Of course, transaction costs of 10% are unreasonably high. However, we expect that this degree of separation in the performance of the algorithms would happen at more reasonable values of transaction costs with trading occurring more frequently than once per day. For comparison, we performed this experiment with 5% transaction costs, and show these results in Fig. 3.7(b). Again we see that our algorithm fairly consistently outperforms the others, though only by a smaller margin. With the newer data set SET2, we performed essentially the same experiments, this time at 5% transaction costs in Fig. 3.8(a) and 1% in Fig. 3.8(b). The results in Fig. 3.8(a) show that the difference between the algorithms is less than in Fig. 3.7(b), though we still have that our algorithm is usually the best performer, followed by Switch-Kozat, and then Switch-Plain at the bottom. Finally, Fig. 3.8(b) shows that the algorithms perform nearly identically for smaller values of transaction costs. Hence, in the great majority of cases, we stand only to gain by incorporating our method of accounting for transaction costs.

We believe these results can be explained by considering the times when transaction costs are taken into account in each algorithm. First of all, the

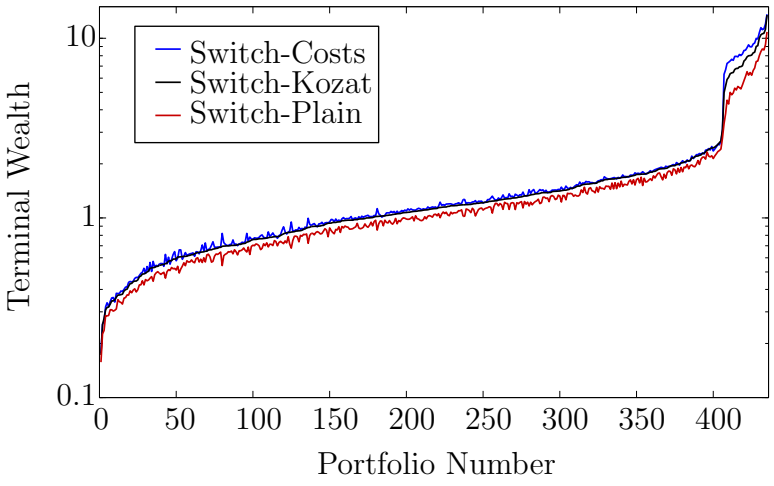


(a) 10% transaction costs.

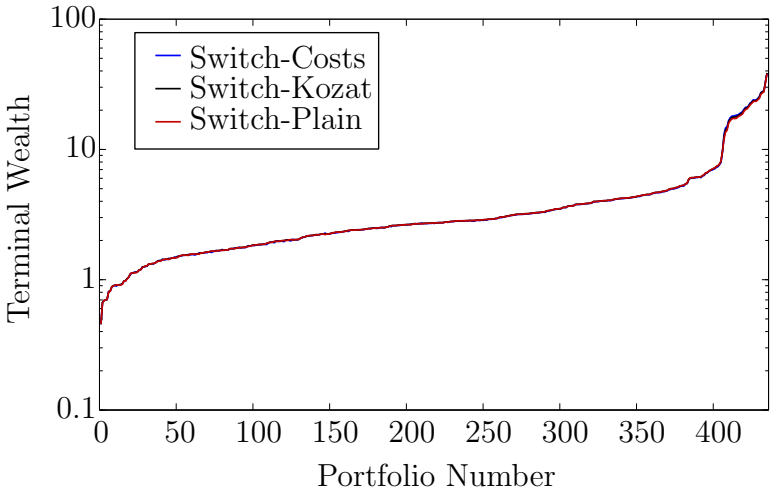


(b) 5% transaction costs.

Figure 3.7: Simulations at with SET1. Portfolio final wealth for 561 different stock pairings, comparing three algorithms. In both, the top curve is the portfolio presented here (Switch-Costs). The middle curve is the portfolio from [38] (Switch-Kozat). The bottom curve is the switching portfolio ignoring transaction costs from [35] (Switch-Plain).



(a) 5% transaction costs.



(b) 1% transaction costs.

Figure 3.8: Simulations with SET2. Portfolio final wealth for 435 different stock pairings, comparing three algorithms. In (a), the top curve is the portfolio presented here (Switch-Costs). The middle curve is the portfolio from [38] (Switch-Kozat). The bottom curve is the switching portfolio ignoring transaction costs from [35] (Switch-Plain).

portfolio from [35] performs the worst because transaction costs are never taken into account. The portfolio from [38] does somewhat better because some transaction costs are taken into account. However, only the costs of rebalancing within a partition of a transition path are considered. Our portfolio performs the best because it considers both the costs of rebalancing within a partition of a transition path as well as the cost of jumping to a new partition. Finally, the algorithms have nearly equal performance for small values of transaction costs because they actually converge to the same algorithm, Switch-Plain, as the transaction costs vanish.

3.8 Conclusion

In this work, we consider the problem of sequentially investing in a stock market where we must pay a fixed percentage commission on every transaction. By generalizing the formulations of the algorithms of Cover [43], Cover and Ordentlich [47], Blum and Kalai [40], and Kozat and Singer [55], we have shown that the key insights of these algorithms can be combined to construct more sophisticated universal portfolio algorithms for a stock market with transaction costs.

We have also introduced the use of factor graphs for the design of universal portfolio algorithms. In particular, we first constructed a universal portfolio generalizing the portfolio of [47]. This new portfolio is designed to be used in a market with side information, where we additionally have a penalty for making adjustments to our portfolio. In this case, factor graphs allow the algorithm to be approximated in a more efficient way, as compared to direct evaluation of the relevant integrals.

We then construct another universal portfolio that builds on the switching portfolio results from [55] and improves on results on switching portfolios under transaction costs from [38]. Again, we show that representing the algorithm with factor graphs reveals that various computational simplifications can be made.

We believe that there may be the potential to make our methods and algorithms even more computationally efficient. For example, the algorithms presented here still have computational complexity that is exponential in the number of stocks used for the portfolio. Kalai and Vempala [59] present

a randomized version of the basic universal portfolio from [43] that gives hope for finding techniques to apply to our algorithms to give them better computational properties when more stocks are used. However, we are not able to directly make use of their results to simplify our algorithms further. As mentioned in the conclusion of [59], their results require the log-concavity of certain wealth related functions, and hence do not extend even to the portfolio from [40], where the only addition is the consideration of transaction costs.

CHAPTER 4

COOPERATIVE ESTIMATION IN HETEROGENEOUS POPULATIONS

4.1 Introduction

The problem of distributed estimation within a network of agents has been extensively studied. This includes such topics as gossip algorithms [9, 60, 61], consensus [62–64], distributed adaptation and estimation [65–68], and others. Related to these is sequential learning or estimation, which includes least mean squares, recursive least squares, Kalman filters [69], stochastic approximation [70], etc. In this chapter, we contribute to these research areas by considering the problem of distributed estimation within a network of heterogeneous agents.[1] Specifically, we consider populations of agents, each of which is trying to learn the parameters of a model for observed data, but these parameters are only consistent (i.e., the optimal model parameters are the same for all agents) within subpopulations of the whole.

We begin with a simplified framework for studying the problem of heterogeneous populations. In particular, we consider a population of N agents, indexed $i \in \{1, ..., N\}$. At each time instant $t \in \{1, 2, ...\}$, agent i makes an observation $x_i(t) \in \{0, 1\}$ drawn according to a Bernoulli distribution with parameter p_i. The observations $x_i(t)$ are independent random variables for all i and all t. Furthermore, we suppose that there is a partitioning of the population of agents into a number of subpopulations, i.e., $G_1 \cup ... \cup G_K = \{1, ..., N\}$, such that $p_i = P_j$ if and only if $i \in G_j$. We let $G(i)$ denote the subpopulation that agent i belongs to. Lastly, the agents are connected to each other in a network given by adjacency matrix A, such that $A_{i,j} = 1$ if nodes i and j are connected, and zero otherwise. Typically, we have $A_{i,i} = 1$ for each agent i, and $A = A^T$. From this adjacency matrix, we can also determine

[1] Portions of this work have been presented in [71, 72], and are reproduced with permission from IEEE.

the neighborhood $\mathcal{N}_i$ for each agent i. Since $A_{i,i} = 1$, we have that $i \in \mathcal{N}_i$.

In [65] and [66], the authors study the problem of distributed parameter estimation using a diffusion protocol for cooperation. In [67], the authors study the problem of distributed parameter estimation for linear state-space models. However, in these works it is assumed that the underlying model parameters w_i^o to be estimated by each agent i are identical, i.e., $w_i^o = w^o$ for all i. However, it is conceivable that the population of agents actually consists of a number of subgroups, such that the model parameters to be estimated are the same within a group, but different between different groups. This is the problem we study in this chapter.

In Section 4.2, we consider the cooperative estimation of the parameters of sequences of Bernoulli random variables. First we examine the situation of homogeneous populations, where all observed Bernoulli random variables are independent and identically distributed. We then extend the cooperative estimation algorithm to the setting where there are a finite number of subpopulations with different parameters, but the same parameter within a subpopulation. In Section 4.3, we present some experimental results on our cooperative algorithm. In Section 4.4, we attempt to formulate an approximation of the squared estimation error as a function of time for the cooperative algorithm in a heterogeneous network. In Section 4.5, we provide more experimental results comparing the estimation error approximation to the observed estimation error in a number of settings. In Section 4.6, we propose how the method for heterogeneous populations can be extended to other cooperative algorithms, such as diffusion least mean squares [65]. Finally, we conclude in Section 4.7 and provide some potential research directions.

4.2 Bernoulli Populations

As mentioned, our initial goal is to have each agent i in a population estimate the parameter p_i of the IID Bernoulli random variables $x_i(t)$ that it successively observes. We would also like for these estimates to be formed in a cooperative fashion, with the hopes of reducing the number of time steps needed for a sufficiently accurate estimate. We begin by first considering how this would be done in a homogeneous population of agents, where each agent observes IID Bernoulli random variables with the same parameter P.

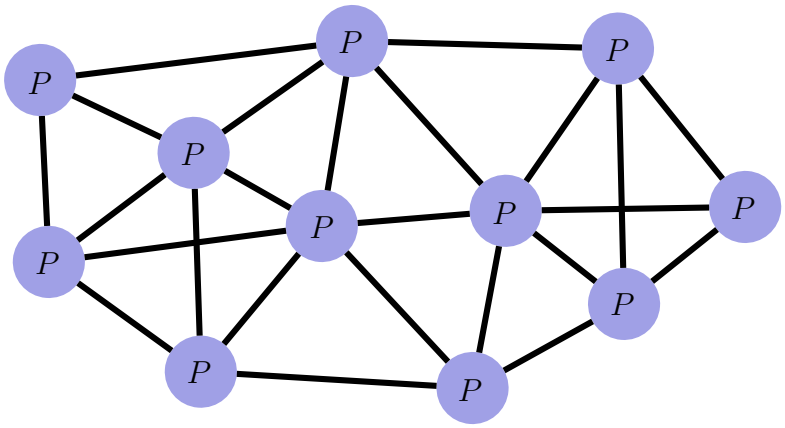


Figure 4.1: A homogeneous population of Bernoulli agents.

This situation is depicted in Fig. 4.1.

4.2.1 Sequential Algorithm from Gossip

Suppose we have only one observation per agent within the homogeneous population, i.e., $x_i(1)$ for $i \in \{1, ..., N\}$. Then the maximum likelihood estimate of P can be found as $\frac{1}{N}\sum_i x_i(1)$. The agents can form estimates of this average in a distributed manner by using, e.g., local convex combinations as discussed in [61]. This can be expressed as the following:

$$\hat{\mathbf{p}}(t+1) = \mathbf{D}(t+1)\hat{\mathbf{p}}(t),$$

where $\hat{\mathbf{p}}(t) = [p_i(t), ..., p_N(t)]^T$ is a vector consisting of the estimates for each agent, and $\hat{\mathbf{p}}(1) \triangleq \mathbf{D}(1)[x_i(1), ..., x_N(1)]^T = \mathbf{D}(1)\mathbf{x}(1)$. Furthermore, $\mathbf{D}(t) \in \mathbb{R}^{N\times N}$ is a matrix such that $\mathbf{D}(t)\mathbf{1}^{N\times 1} = \mathbf{1}^{N\times 1}$ and $0 \leq \mathbf{D}_{i,j}(t) \leq 1$, i.e., each entry in $\mathbf{D}(t)$ is in the range $[0, 1]$. Commonly, the condition that $\mathbf{D}^T(t)\mathbf{1}^{N\times 1} = \mathbf{1}^{N\times 1}$ is also given, but we require this for only some of our results. It will be made clear in the following whether we are using this condition. Finally, since the agents are only able to communicate with each other over the network edges, we have that $\mathbf{D}_{i,j}(t)$ is equal to 0 if $A_{i,j} = 0$.

Hence, we have that

$$\hat{\mathbf{p}}(t) = \left(\prod_{\tau=1}^{t} \mathbf{D}(\tau)\right)\hat{\mathbf{p}}(1) = \left(\prod_{\tau=1}^{t} \mathbf{D}(\tau)\right)\mathbf{x}(1).$$

Now, if instead each agent receives an observation of a Bernoulli random variable at the start of each iteration, we may employ the above procedure for each of the observation vectors $\mathbf{x}(t)$ and combine them as follows:

$$\hat{\mathbf{p}}(t) = \frac{1}{t}\sum_{j=1}^{t}\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)\mathbf{x}(j).$$

However, we note that

$$\begin{aligned}
\hat{\mathbf{p}}(t) &= \frac{1}{t}\sum_{j=1}^{t-1}\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)\mathbf{x}(j) + \frac{1}{t}\mathbf{D}(t)\mathbf{x}(t) \\
&= \frac{t-1}{t}\mathbf{D}(t)\frac{1}{t-1}\sum_{j=1}^{t-1}\left(\prod_{\tau=j}^{t-1}\mathbf{D}(\tau)\right)\mathbf{x}(j) + \frac{1}{t}\mathbf{D}(t)\mathbf{x}(t) \\
&= \frac{t-1}{t}\mathbf{D}(t)\hat{\mathbf{p}}(t-1) + \frac{1}{t}\mathbf{D}(t)\mathbf{x}(t) \\
&= \mathbf{D}(t)\left(\frac{t-1}{t}\hat{\mathbf{p}}(t-1) + \frac{1}{t}\mathbf{x}(t)\right). \qquad (4.1)
\end{aligned}$$

Hence, the computation of $\hat{\mathbf{p}}(t)$ may be performed in a sequential manner using a simple distributed update.

We now prove that for any $\varepsilon > 0$, $\lim_{t\to\infty}\Pr\left[|\hat{p}_i(t) - P| \geq \varepsilon\right] = 0$ for each agent i. To do this, we first note that

$$\begin{aligned}
E\left[\hat{\mathbf{p}}(t)\right] &= E\left[\frac{1}{t}\sum_{j=1}^{t}\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)\mathbf{x}(j)\right] \\
&= \frac{1}{t}\sum_{j=1}^{t}\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)E\left[\mathbf{x}(j)\right] \\
&= \frac{1}{t}\sum_{j=1}^{t}\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)P\mathbf{1}^{N\times 1} \\
&= \frac{P}{t}\sum_{j=1}^{t}\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)\mathbf{1}^{N\times 1} \\
&= \frac{P}{t}\sum_{j=1}^{t}\mathbf{1}^{N\times 1} \\
&= P\mathbf{1}^{N\times 1}.
\end{aligned}$$

Thus, we have that $E[\hat{p}_i(t)] = P$ for each agent i. This can also be shown by

induction as follows: First, suppose $E\left[\hat{\mathbf{p}}(t-1)\right] = P\mathbf{1}^{N\times 1}$. Then

$$\begin{aligned} E\left[\hat{\mathbf{p}}(t)\right] &= E\left[\mathbf{D}(t)\left(\frac{t-1}{t}\hat{\mathbf{p}}(t-1) + \frac{1}{t}\mathbf{x}(t)\right)\right] \\ &= \mathbf{D}(t)\frac{t-1}{t}E\left[\hat{\mathbf{p}}(t-1)\right] + \mathbf{D}(t)\frac{1}{t}E\left[\mathbf{x}(t)\right] \\ &= P\mathbf{1}^{N\times 1}. \end{aligned}$$

Furthermore, note that

$$\begin{aligned} E\left[\hat{\mathbf{p}}(1)\right] &= E\left[\mathbf{D}\mathbf{x}(1)\right] \\ &= \mathbf{D}E\left[\mathbf{x}(1)\right] \\ &= \mathbf{D}P\mathbf{1}^{N\times 1} \\ &= P\mathbf{1}^{N\times 1}. \end{aligned}$$

Thus, by induction, we have that $E[\hat{p}_i(t)] = P$ for each agent i.

We now consider the covariance of $\hat{\mathbf{p}}(t)$:

$$\begin{aligned} \operatorname{cov}\left[\hat{\mathbf{p}}(t)\right] &= \operatorname{cov}\left[\frac{1}{t}\sum_{j=1}^{t}\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)\mathbf{x}(j)\right] \\ &= \frac{1}{t^2}\sum_{j=1}^{t}\operatorname{cov}\left[\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)\mathbf{x}(j)\right] \\ &= \frac{1}{t^2}\sum_{j=1}^{t}\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)\operatorname{cov}\left[\mathbf{x}(j)\right]\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)^{T} \\ &= \frac{1}{t^2}\sum_{j=1}^{t}\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)(P(1-P)I)\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)^{T} \\ &= \frac{P(1-P)}{t^2}\sum_{j=1}^{t}\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)\left(\prod_{\tau=j}^{t}\mathbf{D}(\tau)\right)^{T}. \end{aligned}$$

Now consider the diagonal elements of this covariance matrix, i.e., the variances of each $\hat{p}_i(t)$. Let us define the row vector

$$\mathbf{R}_i(j,t) = i^{\text{th}} \text{ row of } \prod_{\tau=j}^{t}\mathbf{D}(\tau).$$

Then we have that

$$\text{var}\left[\hat{p}_i(t)\right] = \frac{P(1-P)}{t^2}\sum_{j=1}^{t}\mathbf{R}_i(j,t)\mathbf{R}_i(j,t)^T.$$

Since $\mathbf{D}(t)\mathbf{1}^{N\times 1} = \mathbf{1}^{N\times 1}$, we have that $\mathbf{R}_i(j,t)\mathbf{1}^{N\times 1} = 1$. Since $0 \leq \mathbf{D}_{i,j}(t) \leq 1$, we have that each entry of $\mathbf{R}_i(j,t)$ is in the range $[0,1]$. Therefore, $\mathbf{R}_i(j,t)\mathbf{R}_i(j,t)^T \leq \mathbf{R}_i(j,t)\mathbf{1}^{N\times 1} = 1$. Thus, we have that

$$\begin{aligned}
\text{var}\left[\hat{p}_i(t)\right] &\leq \frac{P(1-P)}{t^2}\sum_{j=1}^{t}1 \\
&= \frac{P(1-P)}{t^2} \times t \\
&= \frac{P(1-P)}{t} \\
&\leq \frac{1}{4t}.
\end{aligned}$$

Now, according to Chebyshev's inequality, we have that

$$\Pr\left[|\hat{p}_i(t) - P| \geq \alpha\sigma\right] \leq \frac{1}{\alpha^2},$$

where $\alpha > 0$ and $\sigma^2 = \text{var}[\hat{p}_i(t)]$. If we substitute $\alpha = \frac{\varepsilon}{\sigma}$, we get

$$\Pr\left[|\hat{p}_i(t) - P| \geq \varepsilon\right] \leq \frac{\text{var}[\hat{p}_i(t)]}{\varepsilon^2} \leq \frac{1}{4t\varepsilon^2}.$$

Hence, we have that for any $\varepsilon > 0$, $\lim_{t\to\infty}\Pr\left[|\hat{p}_i(t) - P| \geq \varepsilon\right] = 0$ for each agent i, which means that each agent's estimate $\hat{p}_i(t)$ of the Bernoulli parameter P converges to P in probability. Note that for these results we did not require the condition that $\mathbf{D}^T(t)\mathbf{1}^{N\times 1} = \mathbf{1}^{N\times 1}$.

4.2.2 Sequential Algorithm from Stochastic Approximation

We may also create an algorithm using the concepts from stochastic approximation [70]. Recall that a basic form of the Robbins-Monro stochastic approximation algorithm takes the following form:

$$\hat{w}(t) = \hat{w}(t-1) - \alpha_t Y_t(\hat{w}(t-1)),$$

where $\hat{w}(t)$ is the estimate at iteration t for the solution to $M(w) = 0$, Y_t is a stochastic function such that $M(w) = E[Y_t(w)]$, and α_t is a step size parameter satisfying $\sum_t \alpha_t^2 < \infty$ and $\sum_t \alpha_t = \infty$. We will require that $M(w) < 0$ for $w < w^o$ and $M(w) > 0$ for $w > w^o$ for some w^o. The goal in using this algorithm is to have $\hat{w}(t)$ converge to the root w^o of the deterministic function $M(w)$.

In our case, we can choose $\hat{w}(t) = \hat{p}_i(t)$ and $Y_t(\hat{w}(t)) = Y_t(\hat{p}_i(t)) = \hat{p}_i(t) - x_i(t)$. Note that $M(\hat{p}_i(t)) = \hat{p}_i(t) - p_i$, such that $M(\hat{p}_i(t))$ is an increasing function with a root at p_i. When written in a vector for a population of agents, this leaves us with the update

$$\hat{\mathbf{p}}(t) = \hat{\mathbf{p}}(t-1) - \alpha_t(\hat{\mathbf{p}}(t-1) - \mathbf{x}(t)).$$

In order to make this a cooperative algorithm among the agents in a network, we may include a diffusion step to get the following:

$$\begin{aligned}\tilde{\mathbf{p}}(t) &= \hat{\mathbf{p}}(t-1) - \alpha_t(\hat{\mathbf{p}}(t-1) - \mathbf{x}(t)) \\ \hat{\mathbf{p}}(t) &= \mathbf{D}(t)\tilde{\mathbf{p}}(t),\end{aligned}$$

which can be written as

$$\hat{\mathbf{p}}(t) = \mathbf{D}(t)\left((1-\alpha_t)\hat{\mathbf{p}}(t-1) + \alpha_t\mathbf{x}(t)\right).$$

If we choose $\alpha_t = \frac{1}{t}$, this gives us exactly the algorithm presented in Eq. (4.1) of the previous subsection.

4.2.3 Convergence Speed Analysis

In 4.2.1, we showed that the estimates of all the agents converge to P (in probability). However, there was no discussion of whether the rate of convergence is better than, e.g., noncooperative estimation or how the rate compares to a centralized maximum likelihood estimate. We will now provide results relating to these issues.

We will now consider more closely the variance of $\hat{p}_i(t)$. (Since $E[\hat{p}_i(t)] = P$, this variance is equal to the expected squared estimation error.) For simplicity, assume that we have a time invariant mixing matrix $\mathbf{D}(t) = \mathbf{D}$.

Furthermore, assume that for some positive constant C and $0 \leq \lambda < 1$, we have that $\mathbf{D}^t_{i,j} \leq \frac{1}{N} + C\lambda^t$ for all i, j, and t. The notation $\mathbf{D}^t_{i,j}$ indicates the element of matrix $\mathbf{D}^t$ at row i and column j. This would be true, for example, if $\mathbf{D}$ is a real symmetric doubly stochastic irreducible matrix, such that the unique stationary distribution is uniform.

To consider $\text{var}\,[\hat{p}_i(t)]$, we note that

$$\hat{\mathbf{p}}(t) = \frac{1}{t}\sum_{j=1}^{t} \mathbf{D}^{t-j+1}\mathbf{x}(j). \tag{4.2}$$

Therefore, we have that

$$\hat{p}_i(t) = \frac{1}{t}\sum_{j=1}^{t} \mathbf{d}_i^{(t-j+1)}\mathbf{x}(j), \tag{4.3}$$

where $\mathbf{d}_i^{(t-j+1)}$ is the i^{th} row of the matrix $\mathbf{D}^{t-j+1}$. We can then conclude that

$$\begin{aligned}
\text{var}\,[\hat{p}_i(t)] &= \text{var}\left[\frac{1}{t}\sum_{j=1}^{t} \mathbf{d}_i^{(t-j+1)}\mathbf{x}(j)\right] \\
&= \frac{1}{t^2}\text{var}\left[\sum_{j=1}^{t} \mathbf{d}_i^{(t-j+1)}\mathbf{x}(j)\right] \\
&= \frac{1}{t^2}\sum_{j=1}^{t} \text{var}\left[\mathbf{d}_i^{(t-j+1)}\mathbf{x}(j)\right] \\
&= \frac{1}{t^2}\sum_{j=1}^{t}\sum_{k=1}^{N} \text{var}\left[\left[\mathbf{d}_i^{(t-j+1)}\right]_k x_k(j)\right] \\
&= \frac{1}{t^2}\sum_{j=1}^{t}\sum_{k=1}^{N} \left[\mathbf{d}_i^{(t-j+1)}\right]_k^2 \text{var}\left[x_k(j)\right] \\
&= \frac{P(1-P)}{t^2}\sum_{j=1}^{t}\sum_{k=1}^{N} \left[\mathbf{d}_i^{(t-j+1)}\right]_k^2 \\
&\leq \frac{P(1-P)}{t^2}\sum_{j=1}^{t}\sum_{k=1}^{N} \left(\frac{1}{N} + C\lambda^{t-j+1}\right)^2 \\
&= \frac{P(1-P)}{t^2}\sum_{j=1}^{t}\sum_{k=1}^{N} \left(\frac{1}{N^2} + \frac{2C\lambda^{t-j+1}}{N} + C^2\lambda^{2(t-j+1)}\right)
\end{aligned}$$

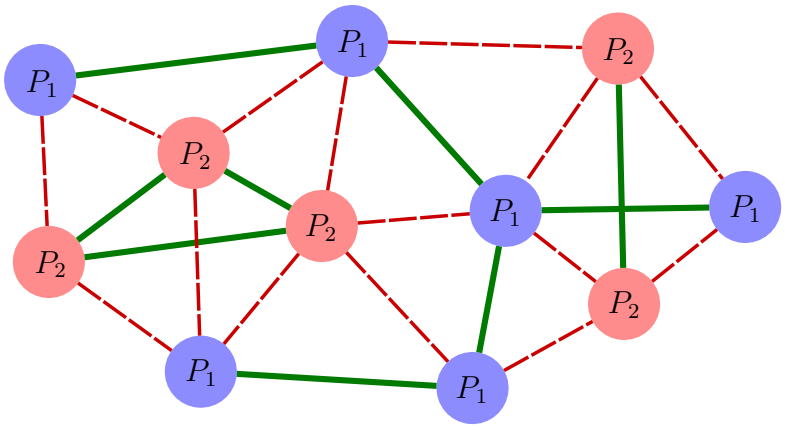


Figure 4.2: A heterogeneous population of Bernoulli agents. Cooperation over the solid green edges is helpful. Cooperation over the dashed red edges is detrimental.

$$
\begin{aligned}
&= \frac{P(1-P)}{t^2} \sum_{j=1}^{t} \left(\frac{1}{N} + 2C\lambda^{t-j+1} + NC^2\lambda^{2(t-j+1)} \right) \\
&= \frac{P(1-P)}{Nt} + \frac{P(1-P)}{t^2} \sum_{j=1}^{t} \left(2C\lambda^{t-j+1} + NC^2\lambda^{2(t-j+1)} \right) \\
&= \frac{P(1-P)}{Nt} + \frac{CP(1-P)}{t^2} \sum_{j=1}^{t} \left(2\lambda^{j} + NC\lambda^{2j} \right) \\
&\leq \frac{P(1-P)}{Nt} + \frac{CP(1-P)}{t^2} \sum_{j=1}^{\infty} \left(2\lambda^{j} + NC\lambda^{2j} \right) \\
&= \frac{P(1-P)}{Nt} + \frac{CP(1-P)}{t^2} \left(\frac{2\lambda}{(1-\lambda)} + \frac{NC\lambda^2}{(1-\lambda^2)} \right).
\end{aligned}
$$

In particular, we note that the rate is dominated by $\frac{P(1-P)}{Nt}$. This is important because $\frac{P(1-P)}{Nt}$ is the expected squared error for a centralized maximum likelihood estimate of the Bernoulli parameter, based on all of the observations from the N agents over t time instants. In other words, the distributed cooperative estimation suffers a small, asymptotically negligible regret with respect to centralized estimation.

4.2.4 Heterogeneous Bernoulli Populations

So far we have only discussed homogeneous populations, but we are interested in the situation where there are various subgroups observing different types of

sources, as shown in Fig. 4.2. Our method for adapting the algorithm to this setting involves having each agent compute an estimate of the parameter p_i based on only its own observations. This will be written as $p_i^\ell(t)$. In particular, this is taken to be

$$p_i^\ell(t) = \begin{cases} \frac{1}{2} & \text{if } t = 0 \\ \frac{1}{t}\sum_{\tau=1}^{t} x_i(\tau) & \text{if } t > 0. \end{cases}$$

We can then choose the elements of $\mathbf{D}(t)$ according to a number of different rules. For example, we may choose:

$$\mathbf{D}_{i,j}(t) = \begin{cases} 0 & \text{if } |p_i^\ell - p_j^\ell| \geq \gamma_t \text{ or } A_{i,j} = 0 \\ \frac{1}{|\tilde{\mathcal{N}}_i(t)|} & \text{if } |p_i^\ell - p_j^\ell| < \gamma_t \text{ and } A_{i,j} = 1, \end{cases} \tag{4.4}$$

where γ_t is a threshold for cooperation between agents and $|\tilde{\mathcal{N}}_i(t)|$ is the number of agents in the neighborhood $\mathcal{N}_i$ of agent i (corresponding to the elements in row i of the adjacency matrix A that equal 1) who satisfy the condition $|p_i^\ell - p_j^\ell| < \gamma_t$. Note that this generates a row stochastic matrix $\mathbf{D}(t)$, such that $\mathbf{D}(t)\mathbf{1}^{N\times 1} = \mathbf{1}^{N\times 1}$. Another rule, which generates a doubly stochastic $\mathbf{D}(t)$ is the following:

$$\mathbf{D}_{i,j}(t) = \begin{cases} \frac{1}{\max\{|\tilde{\mathcal{N}}_i(t)|, |\tilde{\mathcal{N}}_j(t)|\}} & \text{if } \begin{cases} |p_i^\ell - p_j^\ell| < \gamma_t \\ i \neq j \\ \text{and } A_{i,j} = 1 \end{cases} \\ 0 & \text{if } \begin{cases} |p_i^\ell - p_j^\ell| \geq \gamma_t \\ \text{or } A_{i,j} = 0 \end{cases} \\ 1 - \sum_{k\neq i} \mathbf{D}_{i,k}(t) & \text{if } i = j \end{cases}.$$

Therefore, $\mathbf{D}(t)$ is essentially a time varying Metropolis weight matrix, as in [73].

It is possible to show that if we choose $\gamma_t = Ct^\delta$ for some positive constant C and $-\frac{1}{2} < \delta < 0$, the subpopulations will be correctly differentiated and each agent's estimate $\hat{p}_i(t)$ will converge to the true parameter of the model of its observations. More precisely, we mean that

$$\lim_{t\to\infty} \Pr[\mathbf{D}_{i,j}(t) > 0] = \begin{cases} 1 & \text{if } G(j) = G(i) \text{ and } A_{i,j} = 1 \\ 0 & \text{else} \end{cases}$$

and

$$\lim_{t\to\infty} \Pr[|\hat{p}_i(t) - p_i| \geq \varepsilon] = 0 \quad \forall \varepsilon > 0, \forall i \in \{1, ..., N\}.$$

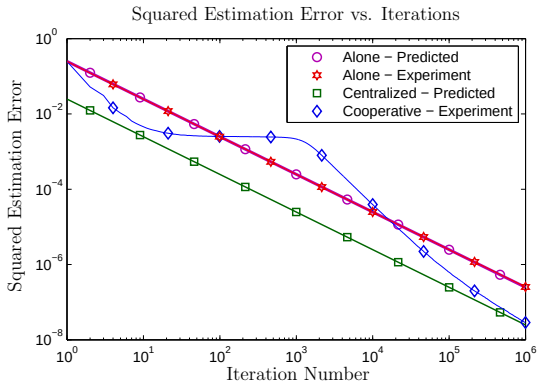
As in the case with a homogeneous population, proving these results involves arguments related to the law of large numbers.

4.3 Simulations: Heterogeneous Populations

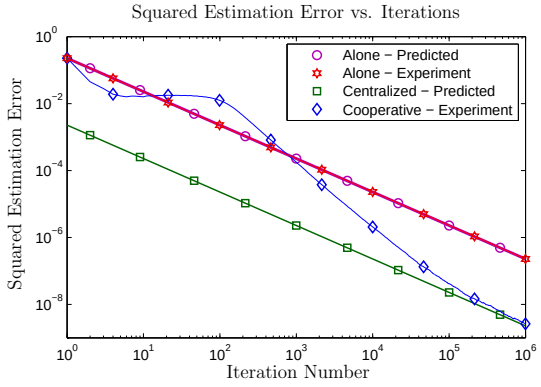
To test our algorithm for Bernoulli heterogeneous populations, we randomly place N agents for each of 2 subpopulations within a 1 unit by 1 unit square. For $N = 10$, we connect agents if they are within 0.5 unit of each other, and we have that $P_1 = 0.45$ and $P_2 = 0.55$. For $N = 100$, we connect agents if they are within 0.25 unit of each other, and we have that $P_1 = 0.35$ and $P_2 = 0.65$. For $N = 1000$, we connect agents if they are within 0.1 unit of each other, and we have that $P_1 = 0.25$ and $P_2 = 0.75$. In all cases, we let $\gamma_t = Ct^{\delta} = t^{-0.4}$. We use the Metropolis weight matrix as the diffusion weights. Finally, we have averaged the squared estimation error over 100 differently seeded networks and observation sequences.

Figure 4.3 shows the results of these experiments. In all cases, we see that cooperative estimation error decreases to a certain level of squared error, and remains at this level for a number of iterations. After this, the squared error decreases with a trend of $O(t^{-2})$ until it approaches the squared error of centralized maximum likelihood estimation within each subpopulation.

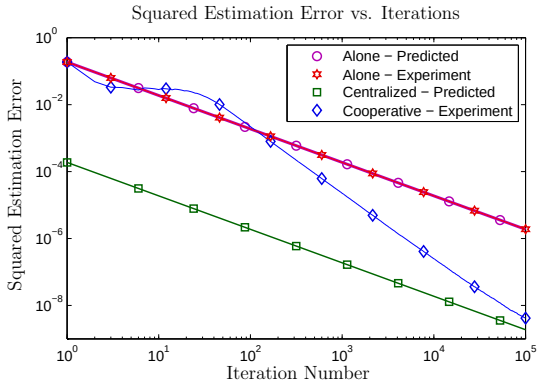
What is happening in these experiments is that the subpopulations are initially mixing estimates between each other, due to high values of γ_t. As they mix their estimates together, the estimates held by the agents converge toward the global average of the observations in the network, rather than the average of the observations separately among the subpopulations. However, since γ_t is decreasing, eventually the between-subpopulation connections become disconnected, allowing the agents of the subpopulations to collaborate correctly only with other agents in the subpopulation. This will be explored further in the following section.



(a) $N = 10$.



(b) $N = 100$.



(c) $N = 1000$.

Figure 4.3: Experiments with two Bernoulli populations. The plots include comparisons of noncooperative estimation, centralized estimation with full knowledge of subpopulation identity, and cooperative estimation using our technique.

4.4 Time to Disconnect

We will now study the convergence properties of the distributed algorithm for heterogeneous populations. To this end, we will attempt to approximate the expected squared error. Specifically, we will consider the case where we choose $\gamma_t = Ct^\delta$ such that γ_1 is large enough that all of the agents will initially collaborate with all of their neighbors. What will happen is that the estimates of all of the agents will converge to the neighborhood of the global mean parameter, and the expected squared error will remain approximately constant for some time. At some point, the subpopulations will disconnect from each other, as a result of the private estimates improving and the collaboration radius γ_t becoming more selective (smaller). We will call this the *time to disconnect* and represent it by t^*. After the time t^*, the subpopulations quickly disconnect from each other, and the expected squared error gradually converges to that of the centralized maximum likelihood estimate within connected subsets of the subpopulations. This is the behavior we observed in Figure 4.3.

To approximate the time to disconnect, we will use basic methods from large deviations theory. In particular, we would like to approximate the time when an edge between agents of different subpopulations (a "bad link") has a low probability of being used for collaboration.

We will begin by recalling a basic large deviation result for sums of independent and identically distributed (IID) random variables. Let $X, X_1, X_2, ...$ be IID random variables and that their common moment generating function $M(\theta) \triangleq E[e^{\theta X}] < \infty$ in some neighborhood B_0 of $\theta = 0$. Furthermore, suppose that the supremum,

$$I(x) \triangleq \sup_\theta \left[\theta x - \log M(\theta)\right],$$

is obtained for some θ in the interior of B_0. Now choose some $x > E[X]$. Then for each $\varepsilon > 0$, there is some N such that for all $n > N$, we have that

$$-n(I(x) + \varepsilon) \leq \log \mathrm{P}\left(\frac{1}{n}\sum_{i=1}^{n} X_i \geq x\right) \leq -nI(x).$$

Now, consider a scenario with two subpopulations, with Bernoulli parameters P_i and $P_j > P_i$. The probability of collaboration on the bad edge between

connected agents i with parameter P_i and j with P_j is $\mathrm{P}[|p_i^\ell - p_j^\ell| < \gamma_t]$, which can be approximated using large deviations. Specifically, we note that $p_i^\ell - p_j^\ell = \frac{1}{t}\sum_\tau (x_i(\tau) - x_j(\tau))$. Define the random variables $X_t = x_i(t) - x_j(t)$. Then we see that $p_i^\ell - p_j^\ell = \frac{1}{t}\sum_\tau X_\tau$ We will approximate $\mathrm{P}[|p_i^\ell - p_j^\ell| < \gamma_t]$ as $\mathrm{P}[p_i^\ell - p_j^\ell > -\gamma_t]$ for $-\gamma_t > E[p_i^\ell - p_j^\ell] = E[X] = P_i - P_j$.

Applying the large deviations result above, we have that

$$\tilde{\mathrm{P}}(t) \triangleq e^{-tI(-\gamma_t)} \approx \mathrm{P}[p_i^\ell - p_j^\ell > -\gamma_t],$$

where $I(x)$ is the large deviations rate function, given by

$$I(x) = x\theta(x) - \log\left(ae^{-\theta(x)} + b + ce^{\theta(x)}\right).$$

Here, we have that $a = (1-P_1)P_2$, $b = P_1P_2 + (1-P_1)(1-P_2)$, $c = P_1(1-P_2)$, and

$$e^{\theta(x)} = \frac{bx + \sqrt{b^2x^2 + 4ac(1-x^2)}}{2c(1-x)}.$$

Now, we are interested in the time when edges disconnect, which is when $\tilde{\mathrm{P}}(t) = e^{-tI(-\gamma_t)}$ becomes small. This transition occurs around the time that the $-tI(-\gamma_t)$ reaches -1. Hence, we choose t^* such that $t^*I(-\gamma_{t^*}) = 1$ and $\gamma_{t^*} < |P_1 - P_2|$. At this point, we will simply note that this does not take into account the number of edges that connect agents of different groups. Many edges should increase the time to disconnect, so the estimate of t^* presented here should be somewhat too early.

To approximate the convergence behavior after the disconnect time, we assume that the estimates converged to the global average of the parameters. We will also assume that the subgroups have completely disconnected from each other, the agents within subgroups are connected by some path, and that the mixing time is instantaneous. Then the expected squared error within the group associated with P_1 after t^* is approximately given by

$$E\left[\left(\frac{t^*N(P_1+d) + \sum_{\tau=t^*+1}^{t}\sum_{i=1}^{N} x_i(\tau)}{Nt} - P_1\right)^2\right],$$

where $d = \frac{P_2 - P_1}{2}$ is the estimated error right before the subgroups disconnect and the agents $1, ..., N$ belong to subgroup P_1. It can be shown that this

leads us to

$$E[(\hat{p}_i(t) - P_1)^2] \approx \frac{P_1(1-P_1)}{Nt} + \frac{d^2 t^{*2} - \frac{1}{N} t^* P_1(1-P_1)}{t^2}.$$

Again, as in the homogeneous case, we see that the convergence is dominated by $\frac{P_1(1-P_1)}{Nt}$, and therefore the convergence rate is nearly as good as centralized maximum likelihood within the subpopulation.

4.5 Simulations: Time to Disconnect

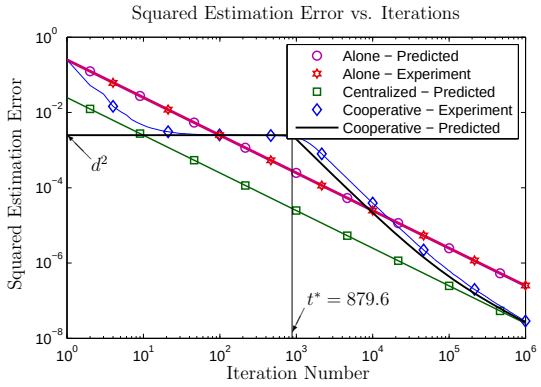
We now revisit the experiments from Section 4.3 in order to examine the quality of our approximation to the expected squared error of the heterogeneous cooperative algorithm. These results are given in Figure 4.4. In the case where $N = 10$, our approximation of the time to disconnect is $t^* = 879.6$. In the case where $N = 100$, we have the approximation that $t^* = 69.0$. Finally, for $N = 1000$, our approximation is $t^* = 20.2$. There are two things we can notice from these experiments. First, it indeed appears that the approximation method produces a value for t^* that is somewhat earlier than reality. Second, we see that the approximation of the error level before the stopping time is a little bit pessimistic. This is particularly evident in the $N = 100$ and $N = 1000$ cases.

4.6 Least Mean Squares Filter Populations

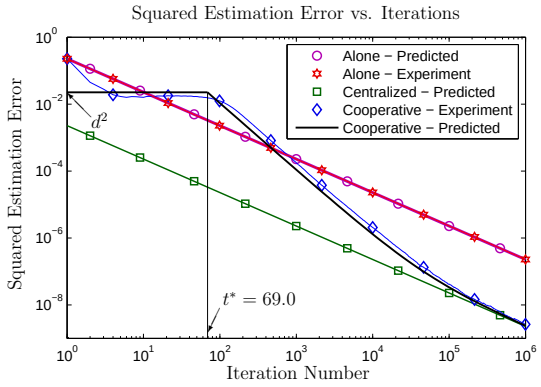
Our method of extending a distributed adaptive algorithm to heterogeneous populations can easily be used in settings beyond Bernoulli parameter estimation. For example, consider a version of the diffusion LMS algorithm [65] with a decreasing, rather than fixed, step size. In this case, we are trying to form estimates $\hat{\psi}_i(t)$ that converge to a vector w_i^o which solves $\min_{\mathbf{w}} E\|d_i - \mathbf{w}^T \mathbf{x}_i\|^2$, where observations of the random variables $d_i \in \mathbb{R}$ and $\mathbf{x}_i \in \mathbb{R}^K$ are made for each time $t \in \{1, 2, ...\}$.

Recall that the cooperative algorithm for a homogeneous population would involve an update from observations, where we have that

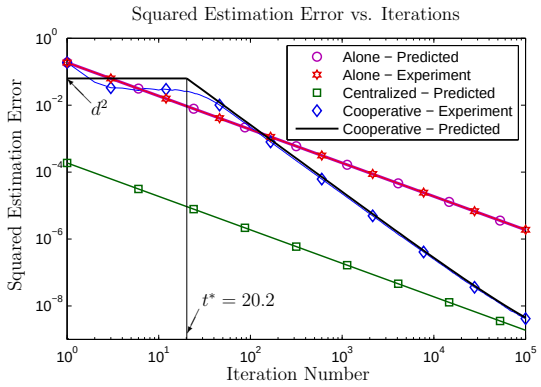
$$\hat{\phi}_i(t) = \hat{\psi}_i(t-1) + \alpha_t \left(d_i(t) - \hat{\psi}_i(t-1)^T \mathbf{x}_i(t) \right) \mathbf{x}_i(t),$$



(a) $N = 10$.



(b) $N = 100$.



(c) $N = 1000$.

Figure 4.4: Experiments with two Bernoulli populations. The black curve is the approximation to the squared error described in Section 4.4.

followed by a local mixing of estimates, such that

$$\hat{\psi}_i(t) = f_i\left(\hat{\phi}_\ell(t); \ell \in \mathcal{N}_i\right),$$

where $\hat{\phi}_i(t)$ is a temporary variable, $f_i(\cdot)$ is the local combining function, typically a convex combination of estimates within the neighborhood $\mathcal{N}_i$ of agent i, α_t is the step size, and the observations at time t are $d_i(t)$ and $\mathbf{x}_i(t)$. However, in order to use our previous concepts to extend to use in heterogeneous populations, we would keep a local (noncooperative) estimate

$$\tilde{\psi}_i(t) = \tilde{\psi}_i(t-1) + \alpha_t\left(d_i(t) - \tilde{\psi}_i(t-1)^T\mathbf{x}_i(t)\right)\mathbf{x}_i(t),$$

and then determine the local combining functions $f_{i,t}(\cdot)$ with a procedure equivalent to the determination of $\mathbf{D}_{i,j}(t)$ in Equations (4.4) and (4.2.4).

4.7 Conclusion

In this chapter, we considered the problem of cooperative distributed estimation within a network of heterogeneous agents. We began with a simplified framework consisting of a network of agents observing streams of Bernoulli random variables, and the goal is for each agent to estimate the parameter of the observed Bernoulli distributions. We provided an algorithm for a homogeneous population of such Bernoulli agents, and showed that the cooperative estimates converge in probability to the correct Bernoulli parameters. We then extended this to the situation where there are a number of subpopulations, within which the Bernoulli parameter is shared. We provide a technique for approximating the behavior of the estimation error as a function of number of iterations. We briefly note that the technique for dealing with heterogeneous populations can be used in other settings, such as in heterogeneous least mean square filter populations. Finally, we present some simulations of our algorithm comparing it to both noncooperative estimation and centralized estimation in a network of Bernoulli agents.

There are many directions that could be taken from here. First of all, we could consider the consequence of knowing the number of subpopulations or knowing the minimum distance between the underlying optimal subpopulation parameter values. For a more adaptive algorithm, rather than asymp-

totic, we could consider an algorithm with fixed, rather than decreasing, step size, in order to accommodate time varying underlying model parameters. It would also be interesting to consider the types of messages that would be sent over communication links. In our case, we sent essentially infinite precision estimates, both cooperative and noncooperative, over the links, but restricting this communication to only the cooperative estimate could be considered, or we may even consider sending only resampled symbols as is done in [28].

CHAPTER 5

ADCs, SGCs, AND SYSTEM METRICS

5.1 Introduction

Digital communication is the process of sending digital data, in the form of bits, from one location, the transmitter, to another, the receiver. Regardless of the origins of the data, the goal of the communication link is to reliably receive all of the data that was sent, or to do so with a bit error probability (BER) that is as small as possible. To achieve this task, the process of inserting redundancy into the data, in the form of forward error correction, converting the binary digital data into a waveform for transmission, and subsequent detection and estimation of the transmitted data is undertaken focusing on this system-relevant metric of performance, namely the bit error rate of the link. While the use of a system-relevant metric for link-level algorithm and architectural designs makes sense, many of the critical components in such designs including circuit and system components are often designed using waveform-centric metrics, such as signal to noise plus distortion ratio, or the total harmonic distortion [74], which consider distortion caused to a sinusoidal input, when the input is reconstructed from its acquired samples, in the case of an analog-to-digital converter.

Initial work using analog-to-digital converter (ADC)-based receivers for 10Gb/s wireline and optical transceivers leveraged the power of DSP-based back-ends making use of modest resolution ADCs [75,76] for subsequent data detection. As rates scale and resolution becomes more challenging, digital calibration has been increasingly employed [77], again focusing on minimizing converter nonlinearities due to ladder offsets or gain and phase mismatches in time-interleaved ADCs [78]. Rather than calibrating the ADC to improve such waveform-centric metrics, a flash converter structure was considered in [74] in which the sampling and reconstruction levels for the ADC

are adjusted to minimize the link BER, resulting in dramatically improved link BER performance for ISI-dominated links, in the low resolution regime typical for the 10Gb/s-100Gb/s range. The information-theoretic capacity (maximum achievable rate with vanishing error probability) of a digital communication link comprising an additive white Gaussian noise channel followed by a low-resolution quantizer was studied in [79–81], along with strategies for reducing converter resolution while maintaining link performance. In [82], mutual information was used to guide the design of achievable-rate optimal nonuniform quantizers for communication links, again demonstrating that, while SNDR and THD would degrade, achievable rate optimal designs have markedly non-uniform comparator thresholds.

Some of the challenges in the design of time-interleaved ADCs are in maintaining constant gain and sampling phase across the branches, requiring considerable calibration and processing circuitry [75]. Similar challenges arise in flash architectures, maintaining uniformly increasing offsets across comparator ladders, while maintaining uniform gain and bandwidth characteristics, again, focusing calibration on waveform-centric metrics. However, for a digital communication link, valuable resources (such as power or chip area) might be better spent minimizing the overall link bit error rate, or maximizing the information capacity of the link.

We therefore propose moving away from ADCs, which have become the most power-hungry, sensitive component in the front-end of a communication link, by abandoning the goal of analog-to-digital conversion altogether.[1] Instead, we propose to replace the tangentially relevant waveform-centric ADC metrics with the true system-level goal of the analog front-end in a communication link: to acquire statistics from the received signal waveforms that are sufficient for the problem of detecting the data that was transmitted. Rather than focusing on whether or not the waveform with all of its temporal properties are preserved by sampling, we focus on the development of a "Statistics Gathering Converter" or (SGC) whose primary role is to gather statistics for subsequent processing that will attempt to recover the transmitted data with low error probability. It is our hope that such architectures can be considerably less complex, require substantially lower power for operation, and be made less sensitive to circuit nonidealities, by focusing on simply preserving

[1]Portions of this work have been presented in [83–86], and are reproduced with permission from IEEE.

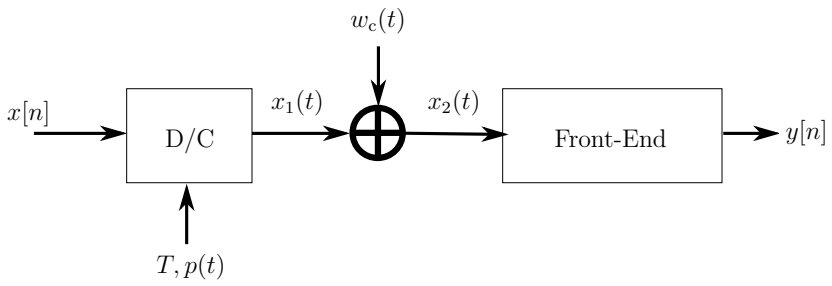


Figure 5.1: Basic model of a communication system.

the information content of the gathered statistics, rather than maintaining waveform integrity.

In Section 5.2, we give a mathematical model for SGCs. In Section 5.3, we present the mutual information rate, linear minimum mean squared error, and bit error rate system level metrics for the unquantized systems described in Section 5.2. In Section 5.4, we provide some computational methods for efficiently approximating the system metrics. In Section 5.5, we shift focus to a quantized version of the SGC, and then proceed to describe the mutual information rate and bit error rate metrics in Section 5.6. In Section 5.7, we provide some simulations examining the performance of SGCs and ADCs on system level metrics and standard ADC metrics. In Section 5.8, we discuss the design of receiver algorithms that take into account the potentially non-standard architectures of SGCs. In Section 5.9, we explore the sensitivity of and SGC architecture to the variation of certain circuit parameters. Finally, we conclude with Section 5.10 and give some potential directions for future research.

5.2 Unquantized System Model

In this work, we model the communication system as in Figure 5.1. This is a simple digital communication link with a pulse amplitude modulation (PAM) transmitter and a data converter front-end to the receiver. The sequence of transmitted symbols, $x[n] \in \mathbb{R}$, is modulated onto the channel through the

D/C converter, such that the transmitter output is

$$x_1(t) = \sum_{i=-\infty}^{\infty} x[i]p(t - iT). \tag{5.1}$$

Hence, we transmit at a rate of one symbol every T seconds, or $1/T$ symbols per second. Without loss of generality, the dispersive effects of the communication channel are lumped within the modulator pulse shape $p(t)$. Additive channel noise is modeled by the zero mean wide sense stationary Gaussian process $w_{\rm c}(t)$. The signal $x_2(t) = x_1(t) + w_{\rm c}(t)$ is received by the converter at the front-end of the receiver.

In a typical front-end converter, there will be a simple periodic sampler, such that the converter output samples consist of $y[n] = x_2(nS) + w_{\rm th}(nS)$, where the converter samples the input signal every S seconds, and incurs some noise. This noise may consist of thermal noise, aperture jitter, or comparator ambiguity, which we lump into a single noise term [87]. We assume that each sample $w_{\rm th}(nS)$ is independent of the others and zero mean IID Gaussian distributed.

In this work, we will compare the standard front-end with converters of the form shown in Figure 5.2, which we will refer to as statistics gathering converters (SGCs). We begin with the situation where the observations $y[n]$ are not quantized. In this front-end, we model M receive filters, $h_0(t), ..., h_{M-1}(t)$. Noise $w_{{\rm th},k}(t)$ is then added to the output of filter $h_k(t)$, and each of these signals is sampled every S seconds. Note that the filters $h_k(t)$ and additive noise may model any non-idealities in the behavior of the sampler. Also note that, with this front-end, we will gather M/S observations per second, as opposed to $1/S$ samples per second as with the classical ADC front-end. Again, we assume that each $w_{{\rm th},k}(nS)$ is independent for all k and n and zero mean IID Gaussian distributed. Note that this is a quite general model for data converters, with the possibility of representing many different types of converter architectures, such as standard ADCs, time-interleaved ADCs, and delay-line based converter architectures considered later in this chapter.

For the purpose of analysis, it is convenient to work with equivalent discrete time models of the communication system. Note that the entire system

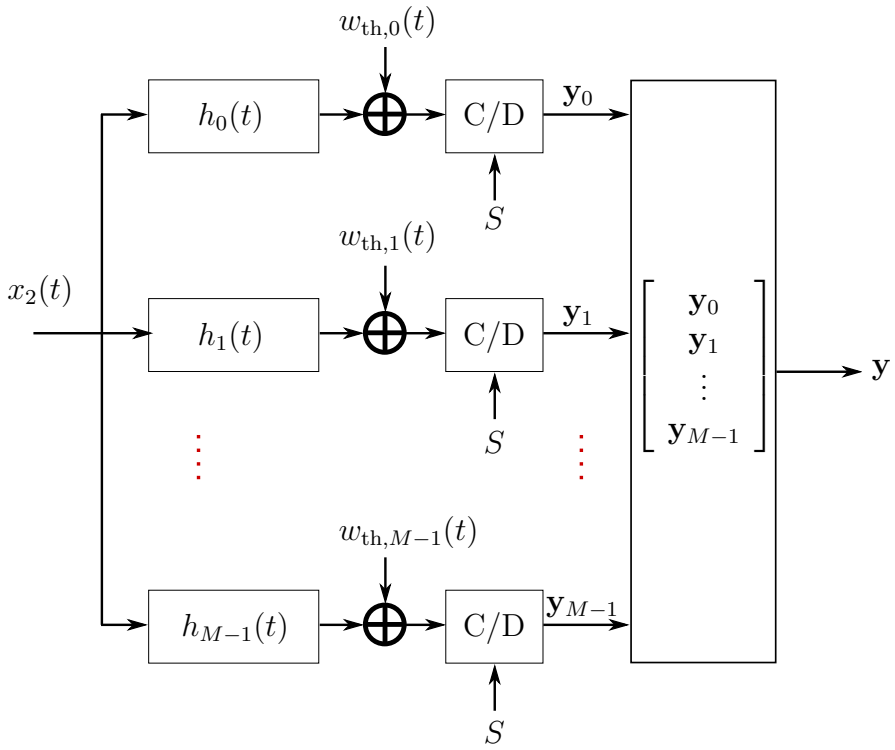


Figure 5.2: A statistics gathering front-end, consisting of a bank of filters whose outputs are sampled with a period of S.

described above is linear. As such, we will determine a matrix $\mathbf{A}$ such that

$$\begin{aligned} \mathbf{y} &= \mathbf{A}\mathbf{x} + \mathbf{w}_{\text{ch}} + \mathbf{w}_{\text{th}} \\ &= \mathbf{A}\mathbf{x} + \mathbf{v}, \end{aligned}$$

where $\mathbf{x} = [x[0], x[1], x[2], ...]^T$ is an entire sequence of transmitted data symbols, $\mathbf{w}_{\text{ch}}$ is the noise component introduced by the channel, $\mathbf{w}_{\text{th}}$ is the noise component introduced by the circuit (thermal noise, aperture jitter, or comparator ambiguity), and $\mathbf{y}$ is the entire sequence of observations. Of course, the number of elements of $\mathbf{y}$ must be approximately MT/S times the size of $\mathbf{x}$, or larger, in order to reliably estimate the full transmitted data sequence. For convenience, we have defined $\mathbf{v} = \mathbf{w}_{\text{ch}} + \mathbf{w}_{\text{th}}$.

For the front-end of Figure 5.2, let

$$\mathbf{y} = \Pi_M \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{M-1} \end{bmatrix},$$

where each $\mathbf{y}_k$ is a vector of observations taken from filter $h_k(t)$, each $\mathbf{y}_k$ has the same number of elements, and Π_M is a permutation matrix that interleaves the elements of each $\mathbf{y}_k$. Specifically, let the notation $[\mathbf{M}]_{i,j}$ indicate the element in the i^{th} row and j^{th} column of a matrix $\mathbf{M}$, where the top left element has indices $i = j = 0$. Also, let Γ be the number of elements in each $\mathbf{y}_k$, such that $\mathbf{y}$ has $M\Gamma$ elements. Then we have that

$$[\Pi_M]_{i,j} = \begin{cases} 1 & \text{if} \quad \begin{cases} \left\lfloor \frac{i}{M} \right\rfloor = \text{mod}\,(j, \Gamma) \\ \qquad \text{and} \\ \left\lfloor \frac{j}{\Gamma} \right\rfloor = \text{mod}\,(i, M) \end{cases} \\ 0 & \text{else} \end{cases},$$

where we have that the notation $\lfloor f \rfloor$ indicates the largest integer not greater than f, and $\text{mod}\,(f, g) = f - g\left\lfloor \frac{f}{g} \right\rfloor$. Hence, we have that

$$\mathbf{y} = [y_0[0], ..., y_{M-1}[0], y_0[1], ..., y_{M-1}[1], y_0[2], ..., y_{M-1}[2], \cdots]^T.$$

Note that computing the permutation $\mathbf{y} = \Pi_M \hat{\mathbf{y}}$ for column vector $\hat{\mathbf{y}}$ would be equivalent to the following MATLAB command:

$$\mathbf{y} = \texttt{reshape(reshape(}\hat{\mathbf{y}}\texttt{,M,[])',[],1);}.$$

Now, define $q_k(t) = (p * h_k)(t)$, i.e., the aggregate pulse shape from input symbol to the output of filter $h_k(t)$. We use the notation $(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) \mathrm{d}\tau$ to indicate convolution. Then each $\mathbf{y}_k$ is given by

$$\mathbf{y}_k = \begin{bmatrix} y_k[0] \\ y_k[1] \\ y_k[2] \\ \vdots \end{bmatrix} = \mathbf{A}_k \mathbf{x} + \mathbf{w}_{\text{ch},k} + \mathbf{w}_{\text{th},k},$$

where we have that $[\mathbf{A}_k]_{i,j} = q_k\left(iS - jT\right)$, i.e.,

$$\mathbf{A}_k = \begin{bmatrix} q_k(0S-0T) & q_k(0S-1T) & q_k(0S-2T) & \cdots \\ q_k(1S-0T) & q_k(1S-1T) & q_k(1S-2T) & \cdots \\ q_k(2S-0T) & q_k(2S-1T) & q_k(2S-2T) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{5.2}$$

Therefore we have that $\mathbf{A}$ from Eq. (5.2) is given by

$$\mathbf{A} = \Pi_M \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_{M-1} \end{bmatrix}.$$

We now need to specify stochastic properties of the communications system. First of all, we assume that the data symbols $x[n]$ are IID with zero mean and unit variance, i.e., $E[\mathbf{x}\mathbf{x}^H] \triangleq \mathbf{R}_{\mathbf{x}} = \mathbf{I}$. The notation $\mathbf{x}^H$ indicates complex conjugate transposition for a vector or matrix. Next, note that previous assumptions readily let us conclude that the noise components $\mathbf{w}_{\text{ch}}$ and $\mathbf{w}_{\text{th}}$ are zero mean Gaussian. We have also assumed that $E[\mathbf{w}_{\text{th}}\mathbf{w}_{\text{th}}^H] \triangleq \mathbf{R}_{\mathbf{w}_{\text{th}}} = \sigma_{\text{th}}^2\mathbf{I}$, where σ_{th}^2 is the noise variance and $\mathbf{I}$ is a properly sized identity matrix. We now wish to find an expression for $E[\mathbf{w}_{\text{ch}}\mathbf{w}_{\text{ch}}^H]$. Since we have that

$$\mathbf{w}_{\text{ch}} = \Pi_M \begin{bmatrix} \mathbf{w}_{\text{ch},0} \\ \vdots \\ \mathbf{w}_{\text{ch},M-1} \end{bmatrix},$$

we can see that

$$E[\mathbf{w}_{\text{ch}}\mathbf{w}_{\text{ch}}^H] \triangleq \mathbf{R}_{\mathbf{w}_{\text{ch}}} = \Pi_M \begin{bmatrix} \Sigma_{0,0} & \Sigma_{0,1} & \ldots & \Sigma_{0,M-1} \\ \Sigma_{1,0} & \Sigma_{1,1} & \ldots & \Sigma_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{M-1,0} & \Sigma_{M-1,1} & \ldots & \Sigma_{M-1,M-1} \end{bmatrix} \Pi_M^H.$$

For this, we have defined $\Sigma_{i,j} = E[\mathbf{w}_{\text{ch},i}\mathbf{w}_{\text{ch},j}^H]$. It is furthermore straightfor-

ward to show that

$$\Sigma_{i,j} = \begin{bmatrix} R_{i,j}(0S) & R_{i,j}(-1S) & R_{i,j}(-2S) & \dots \\ R_{i,j}(1S) & R_{i,j}(0S) & R_{i,j}(-1S) & \dots \\ R_{i,j}(2S) & R_{i,j}(1S) & R_{i,j}(0S) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

where we have that $R_{ij}(t)$ is the cross-correlation between the channel noise that is filtered and observed at the output from filter $h_i(t)$ and from filter $h_j(t)$. Note that $R_{ij}(t) = (R_{w_c} * h_i * \tilde{h}_j)(t)$, where $R_{w_c}(t)$ is the autocorrelation of the channel noise process and $\tilde{h}_j(t) = h_j^*(-t)$, i.e., the time reversed complex conjugate of the impulse response $h_j(t)$. (Note, however, that we consider only real valued systems, so that $h_j^*(t) = h_j(t)$.) Finally, we have that $\mathbf{R}_{\mathbf{v}} = \mathbf{R}_{\mathbf{w}_{\text{ch}}} + \mathbf{R}_{\mathbf{w}_{\text{th}}}$.

5.3 Metrics for Unquantized Systems

In this section, we present three metrics that we will use to compare data converter architectures. These are input to output mutual information per input symbol (MI), linear minimum mean square error (LMMSE), and bit error rate (BER). Before giving the definition of the mutual information metric, let us first write the linear input-output relationship so that the input data size is given explicitly:

$$\mathbf{y}_{(N)} = \mathbf{A}_{(N)}\mathbf{x}_{(N)} + \mathbf{v}_{(N)}.$$

Then, the mutual information metric, given in bits per input symbol, is defined as

$$C(\mathbf{A}, \mathbf{v}) = \lim_{N\to\infty} \frac{1}{2N} \left(\log_2 \frac{\left| \mathbf{A}_{(N)} \mathbf{R}_{\mathbf{x}_{(N)}} \mathbf{A}_{(N)}^H + \mathbf{R}_{\mathbf{v}_{(N)}} \right|}{\left| \mathbf{R}_{\mathbf{v}_{(N)}} \right|} \right), \tag{5.3}$$

if the limit exists, for any multivariate Gaussian distributed input sequence. When we compute this mutual information metric, we do so under the assumption of Gaussian IID distributed input symbols with zero mean and unit variance, such that $\mathbf{R}_{\mathbf{x}_{(N)}} = \mathbf{I}$. Recall that we have already specified

that the noise component of the observations is jointly Gaussian distributed. The objective of design based on mutual information would typically be to maximize the value of $C(\mathbf{A}, \mathbf{v})$.

The next metric, LMMSE, is given by the following:

$$D(\mathbf{A}, \mathbf{v}) = \lim_{N \to \infty} \frac{1}{N} \mathrm{tr} \left(\mathbf{R}_{\mathbf{x}_{(N)}} - \mathbf{R}_{\mathbf{x}_{(N)}} \mathbf{A}_{(N)}^H \mathbf{R}_{\mathbf{y}_{(N)}}^{-1} \mathbf{A}_{(N)} \mathbf{R}_{\mathbf{x}_{(N)}} \right),$$

where we have that

$$\mathbf{R}_{\mathbf{y}_{(N)}} = \mathbf{A}_{(N)} \mathbf{R}_{\mathbf{x}_{(N)}} \mathbf{A}_{(N)}^H + \mathbf{R}_{\mathbf{v}_{(N)}},$$

and the notation $\mathrm{tr}(\cdot)$ indicates the matrix trace. Note that the expression may be simplified since we have previously defined $\mathbf{R}_{\mathbf{x}_{(N)}} = \mathbf{I}$. This expression can be used for arbitrary zero mean unit variance IID input distributions. The objective of design based on LMMSE would typically be to minimize the value of $D(\mathbf{A}, \mathbf{v})$.

The final metric we consider is bit error rate (BER), which will be denoted $p_e(\mathbf{A}, \mathbf{v})$. The assumption when we use this metric is that the input symbols $x[n]$ are chosen from the finite alphabet $\{-1, +1\}$ with equal probability. Hence, we again have a zero mean unit variance input distribution. We define the BER as the average probability of detection error from the LMMSE estimate of the input symbols. We use the LMMSE estimator due to the tractability of the resultant expressions. Let $Q(x) = \Pr[X > x]$, where X is a zero mean unit variance Gaussian random variable. Let $\hat{\mathbf{x}}_{(N)} = \{\hat{x}[0], \hat{x}[1], ..., \hat{x}[N-1]\}$ be the LMMSE estimate of $\mathbf{x}_{(N)}$. Define $\mathbf{H}_{(N)} = \mathbf{A}_{(N)}^H \mathbf{R}_{\mathbf{y}_{(N)}}^{-1}$. Then we have that $E[\hat{\mathbf{x}}_{(N)} | \mathbf{x}_{(N)}] = \mathbf{H}_{(N)} \mathbf{A}_{(N)} \mathbf{x}_{(N)}$ and $\mathbf{R}_{\hat{\mathbf{x}}_{(N)} | \mathbf{x}_{(N)}} = \mathbf{H}_{(N)} \mathbf{R}_{\mathbf{v}_{(N)}} \mathbf{H}_{(N)}^H$. Finally, this gives us the (LMMSE) BER metric as

$$p_e(\mathbf{A}, \mathbf{v}) = \lim_{N \to \infty} E_{\mathbf{x}_{(N)}} \left[\frac{1}{N} \sum_{i=0}^{N-1} Q \left(\frac{x[i] E[\hat{x}[i] | \mathbf{x}_{(N)}]}{\sigma_{(N)}[i]} \right) \right],$$

where $(\sigma_{(N)}[i])^2$ are the diagonal elements of $\mathbf{R}_{\hat{\mathbf{x}}_{(N)} | \mathbf{x}_{(N)}}$. The objective of design based on bit error rate could be to minimize the value of $p_e(\mathbf{A}, \mathbf{v})$ or to achieve a given $p_e(\mathbf{A}, \mathbf{v})$ while optimizing other system characteristics such as power consumption, circuit area, etc.

All of these metrics can be closely approximated by evaluating the expression inside the limit using a suitably large block size N. However, achieving a sufficient level of accuracy in the approximation may require the use of large blocks, which may become computationally prohibitive. In the next section, we will examine computationally efficient ways of approximating the metrics. As a final comment, we note that exact a priori knowledge of the aggregate pulse shapes $q_k(t)$ at the receiver is not necessary, since we may train an adaptive equalizer to arbitrary precision using an asymptotically negligible amount of training symbols. Of course, this equalizer must exploit the particular MIMO-like structure of the SGC architecture in order to approach the performance indicated by the system metrics.

5.4 Efficient Computation of System Metrics

As mentioned, it is possible to compute approximations to the system metrics by taking large block sizes and computing the expressions directly. However, there are many cases where this could be prohibitive. For example, if a converter design is being optimized with respect to a number of parameters, the optimization procedure may require many evaluations of the metric under different values of the parameters. As such, we would like to find a way to compute the metrics quickly.

We will begin by presenting some fundamental methods. First, consider the following result on the limits of determinants of block Toeplitz matrices [88, 89]:

Let $\varphi[t]$ be a square matrix for each $-K < t < K$. Let $T_K[\varphi]$ be the block Toeplitz matrix constructed as follows:

$$T_K[\varphi] = \begin{bmatrix} \varphi[0] & \varphi[-1] & \varphi[-2] & \cdots & \varphi[1-K] \\ \varphi[+1] & \varphi[0] & \varphi[-1] & \cdots & \varphi[2-K] \\ \varphi[+2] & \varphi[+1] & \varphi[0] & \cdots & \varphi[3-K] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varphi[K-1] & \varphi[K-2] & \varphi[K-3] & \cdots & \varphi[0] \end{bmatrix}.$$

Let $D_K[\varphi]$ be the determinant of $T_K[\varphi]$, i.e., $D_K[\varphi] = |T_K[\varphi]|$. Finally, for

$z \in \mathbb{C}$ and $|z| = 1$, let

$$\varphi(z) \triangleq \sum_{t=-\infty}^{\infty} \varphi[t] z^t. \tag{5.4}$$

Then, if $\varphi(z)$ is continuous and positive definite for $|z| = 1$, we have that

$$\lim_{K \to \infty} \frac{1}{K} \log(D_K[\varphi]) = \frac{1}{2\pi} \int_0^{2\pi} \log \left| \varphi(e^{i\theta}) \right| d\theta.$$

Here, we use $i \triangleq \sqrt{-1}$. Of course, simply by applying a scale factor of $\frac{1}{\log 2}$, we have that

$$\lim_{K \to \infty} \frac{1}{K} \log_2(D_K[\varphi]) = \frac{1}{2\pi} \int_0^{2\pi} \log_2 \left| \varphi(e^{i\theta}) \right| d\theta.$$

Next, we will consider a generalization of standard linear time invariant systems. In the standard case, we may have (real or complex) scalar-valued discrete time signals $a[t]$ and $b[t]$, which may be convolved to produce $c[t] = (a * b)[t]$, where we have that

$$c[t] = (a * b)[t] = \sum_{\tau=-\infty}^{\infty} a[\tau] b[t - \tau]. \tag{5.5}$$

Let $a(\theta)$ be the discrete time Fourier transfom (DTFT) of $a[t]$, and $b(\theta)$ be the DTFT of $b[t]$. For any signal $x[t]$, the DTFT of $x[t]$ is defined as

$$x(\theta) = \sum_{t=-\infty}^{\infty} e^{-i\theta t} x[t].$$

It is known that the DTFT of $c[t]$ is simply given as $c(\theta) = a(\theta)b(\theta)$. Note that this convolution is commutative for scalar valued sequences.

Similarly, let $a_K[\omega]$ be the length $2K+1$ discrete Fourier transform (DFT) of $a_K[t]$, where $a_K[t]$ is defined as the length $2K+1$ truncation of $a[t]$, such that $a_K[t] = a[t]$ for $t = -K, ..., K$. For notational convenience, we let $a_K[t + 2K + 1] = a_K[t]$ for all t, such that the signal is periodic with a period of $2K + 1$ elements. Define $b_K[\omega]$ and $b_K[t]$ similarly. For any signal $x[t]$, we will define the length $2K + 1$ DFT of $x[t]$ as

$$x_K[\omega] = \sum_{t=-K}^{K} \exp\left(\frac{-i2\pi\omega t}{2K+1} \right) x_K[t].$$

We will take $\omega = -K, ..., K$. We will also define $c_K[t]$, the length $2K+1$ circular convolution of $a_K[t]$ and $b_K[t]$, as

$$c_K[t] = (a_K * b_K)[t] = \sum_{\tau=-K}^{K} a_K[\tau] b_K[t-\tau],$$

where $t = -K, ..., K$. Note that $t - \tau$ may extend outside of the range $\{-K, ..., K\}$, but the periodic definition of b_K ensures that the convolution wraps around properly. Similar to the DTFT, we have that $c_K[\omega] = a_K[\omega] b_K[\omega]$.

We now generalize to (real or complex) matrix-valued sequences. Let $A[t]$ be a sequence of matrices, where $A[t] \in \mathbb{C}^{L \times M}$, and similarly $B[t]$ is a sequence of matrices where $B[t] \in \mathbb{C}^{M \times N}$. Let $C[t] = (A * B)[t]$ be the convolution of these sequences, where this convolution is defined as follows:

$$C[t] = (A * B)[t] = \sum_{\tau=-\infty}^{\infty} A[\tau] B[t-\tau].$$

Note that this is a straightforward generalization of the convolution definition in Equation (5.5), except that this is now no longer commutative in general. Of course, we have that $C[t] \in \mathbb{C}^{L \times N}$. The length $2K+1$ truncated circular convolution is defined similarly as above. We let $A_K[t] = A[t]$ for $t \in \{-K, ..., K\}$ and $A_K[t] = A_K[t + 2K + 1]$ for all t. $B_K[t]$ is defined similarly. Finally we have that the circular convolution of A_K and B_K is given as

$$C_K[t] = (A_K * B_K)[t] = \sum_{\tau=-K}^{K} A_K[\tau] B_K[t-\tau],$$

where $t = -K, ..., K$.

Now, define the DTFT of a matrix sequence $X[t]$ as

$$X(\theta) = \sum_{t=-\infty}^{\infty} e^{-i\theta t} X[t].$$

Also, define the DFT of a length $2K+1$ truncation of $X[t]$, for $t = -K, ..., K$, as

$$X_K[\omega] = \sum_{t=-K}^{K} \exp\left(\frac{-i2\pi\omega t}{2K+1}\right) X_K[t]. \tag{5.6}$$

Then, just as in the scalar case, we have that $C(\theta) = A(\theta)B(\theta)$ and $C_K[\omega] = A_K[\omega]B_K[\omega]$. Again, it is clear that this operation is commutative only in special cases, where scalar sequences are an example of such a special case. Below, we will also reference the inverse DFT of a sequence $X_K[\omega]$. This inverse transform is given as follows:

$$X_K[t] = \frac{1}{2K+1} \sum_{\omega=-K}^{K} \exp\left(\frac{i2\pi\omega t}{2K+1}\right) X_K[\omega],$$

where $t = -K, ..., K$.

Now, consider block-wise defined matrices $A_K \in \mathbb{C}^{KL\times KM}$ and $B_K \in \mathbb{C}^{KM\times KN}$, where we have that

$$A_K = \begin{bmatrix} A[0] & A[-1] & A[-2] & \cdots & A[1-K] \\ A[1] & A[0] & A[-1] & \cdots & A[2-K] \\ A[2] & A[1] & A[0] & \cdots & A[3-K] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A[K-1] & A[K-2] & A[K-3] & \cdots & A[0] \end{bmatrix}$$

and

$$B_K = \begin{bmatrix} B[0] & B[-1] & B[-2] & \cdots & B[1-K] \\ B[1] & B[0] & B[-1] & \cdots & B[2-K] \\ B[2] & B[1] & B[0] & \cdots & B[3-K] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B[K-1] & B[K-2] & B[K-3] & \cdots & B[0] \end{bmatrix}.$$

We note that the matrix product $C_K = A_K B_K$ does not generally have a similar block-wise structure. However, the product $C_K = A_K B_K$ may approximately have such a block-wise structure, such as when K is large, and when $A[t]$ and $B[t]$ have a finite support around $t = 0$. In particular, the sequence that describes the approximate block-wise structure of the matrix C_K is simply the convolution $C[t] = (A * B)[t]$. This sequence $C[t]$ may be approximated by computing the circular convolution $C_K[t] = (A_K * B_K)[t]$ for some sufficiently large K, and this may be accomplished utilizing the matrix sequence DFT and inverse DFT described above. Furthermore, in

some instances we may be interested only in the quantity

$$\lim_{K\to\infty} \frac{1}{KN}\mathrm{tr}\left(A_K B_K\right),$$

where $A[t] \in \mathbb{C}^{N\times M}$ and $B[t] \in \mathbb{C}^{M\times N}$. Note that the product $C_K = A_K B_K$, in this case, has size $KN \times KN$, such that the given quantity is simply an average over the diagonal elements. If we believe that C_K is sufficiently well approximated by a block-wise structured matrix defined by the sequence $C[t] = (A * B)[t]$, where $C[t] \in \mathbb{C}^{N\times N}$, then we may write that

$$\lim_{K\to\infty} \frac{1}{KN}\mathrm{tr}\left(A_K B_K\right) \approx \frac{1}{N}\mathrm{tr}\left(C[0]\right).$$

We conjecture that this approximation is exact in certain cases, such as when the sequences $A[t]$ and $B[t]$ have finite support. Now, if we have the sequences $A[t]$ and $B[t]$, and wish to compute $\lim_{K\to\infty} \frac{1}{KN}\mathrm{tr}\left(A_K B_K\right)$, it is likely simplest to form our approximation using

$$C[0] \approx \sum_{\tau=-\bar{\tau}}^{\bar{\tau}} A[\tau]B[-\tau],$$

where $\bar{\tau}$ is chosen sufficiently large. On the other hand, we may instead want to compute the following:

$$\lim_{K\to\infty} \frac{1}{KN}\mathrm{tr}\left(A_K^1 \times A_K^2 \times \cdots \times A_K^J\right),$$

where the product of J matrices A_K^1 to A_K^J is a square matrix of size $KN \times KN$, and the component blocks of each A_K^j have compatible dimensions with respect to the matrix product. If this is the case, it is likely to be computationally advantageous to employ a frequency domain method, utilizing the DFT defined above.

A final conjecture that we will use without proof is that, if A_K^{-1} exists, A_K is approximately block Toeplitz with blocks of size $L \times L$, and the sequence of matrices $A[t]$ satisfies some condition such as finite support, then A_K^{-1} is also approximately block Toeplitz with blocks of size $L \times L$. Furthermore, the DTFT of the sequence of matrices associated with approximate block Toeplitz structure of A_K^{-1} is equal to $(A(\theta))^{-1}$.

We would now like to use the above relationships to approximate the mu-

tual information expression in Equation (5.3). To do so, note that

$$C(\mathbf{A}, \mathbf{v}) = \lim_{N\to\infty} \frac{1}{2N} \left(\log_2 \frac{\left|\mathbf{A}_{(N)}\mathbf{A}_{(N)}^H + \mathbf{R}_{\mathbf{v}_{(N)}}\right|}{\left|\mathbf{R}_{\mathbf{v}_{(N)}}\right|} \right)$$
$$= \frac{1}{2} \left(\lim_{N\to\infty} \frac{1}{N} \log_2 \left|\mathbf{A}_{(N)}\mathbf{A}_{(N)}^H + \mathbf{R}_{\mathbf{v}_{(N)}}\right| - \lim_{N\to\infty} \frac{1}{N} \log_2 \left|\mathbf{R}_{\mathbf{v}_{(N)}}\right| \right).$$

Now, suppose that $\frac{T}{S}$ is a rational number, and the numbers L_S and L_T are the smallest positive integers such that $L_S S = L_T T$. Also, suppose that the number of rows of $\mathbf{A}_{(N)}$ is some multiple of $L_S M$, and likewise for $\mathbf{R}_{\mathbf{v}_{(N)}}$. Then the matrices $\mathbf{A}_{(N),k}$ from Equation (5.2) are composed of repeating rectangular blocks of size $L_S \times L_T$ in a Toeplitz-like fashion, which implies that the matrix $\mathbf{A}_{(N)}$ is composed of repeating rectangular blocks of size $L_S M \times L_T$. This in turn implies that the matrix product $\mathbf{A}_{(N)}\mathbf{A}_{(N)}^H$ is approximately block Toeplitz with blocks of size $L_S M \times L_S M$. Note also that the covariance matrix $\mathbf{R}_{\mathbf{v}_{(N)}}$ is block Toeplitz with blocks of size $M \times M$, which furthermore implies, by grouping smaller blocks into large blocks, that $\mathbf{R}_{\mathbf{v}_{(N)}}$ is block Toeplitz with blocks of size $L_S M \times L_S M$. Specifically, let $\varphi^R[t]$ be the component blocks of $\mathbf{R}_{\mathbf{v}_{(N)}}$ and let $\varphi^A[t]$ be the (approximate) component blocks of $\mathbf{A}_{(N)}\mathbf{A}_{(N)}^H$, with all mentioned blocks being size $L_S M \times L_S M$. Then, for $0 \le i, j < L_S M$, we have that

$$\left[\varphi^R[t]\right]_{i,j} = \begin{cases} R_{\hat{\imath},\hat{\jmath}}(tS) + \sigma^2_{\text{th}} & \text{if } i - j = t = 0 \\ R_{\hat{\imath},\hat{\jmath}}(tS) & \text{else} \end{cases},$$

where $\hat{\imath} \triangleq \text{mod}\,(i, M)$, $\hat{\jmath} \triangleq \text{mod}\,(j, M)$, and $R_{\hat{\imath},\hat{\jmath}}(\cdot)$ is the cross correlation defined above.

In order to specify the matrices $\varphi^A[t]$, we first define the following discrete time matrix sequence $\rho^A[t]$ that describes the block-wise structure of $\mathbf{A}_{(N)}$:

$$\left[\rho^A[t]\right]_{i,j} = \begin{cases} [\mathbf{A}]_{(i+tL_S M),j} & \text{if } t \ge 0 \\ [\mathbf{A}]_{i,(j-tL_T)} & \text{if } t < 0 \end{cases}, \tag{5.7}$$

where $0 \le i < L_S M$, $0 \le j < L_T$, and $t \in \mathbb{Z}$. Then the sequence $\varphi^A[t]$ that describes the approximate block Toeplitz structure of $\mathbf{A}_{(N)}\mathbf{A}_{(N)}^H$ is simply

$$\varphi^A[t] = (\rho^A * \tilde{\rho}^A)[t],$$

where we define $\tilde{\rho}^A[t] = (\rho^A[-t])^H$, i.e., the time reversed Hermitian transpose of ρ^A. Furthermore, define $\rho^A(\theta)$ as the DTFT of the matrix sequence $\rho^A[t]$, and note that $\varphi^A(e^{i\theta})$ and $\varphi^R(e^{i\theta})$, defined as in Equation (5.4), are simply the reverse of the discrete time Fourier transforms of the sequences $\varphi^A[t]$ and $\varphi^R[t]$, as a function of θ. This leads us to conclude that

$$\varphi^A(e^{i\theta}) = \rho^A(-\theta)\left(\rho^A(-\theta)\right)^H,$$

which further implies that $\varphi^A(e^{i\theta})$ is non-negative definite. By similar arguments, we can conclude that $\varphi^R(e^{i\theta})$ is positive definite, where a non-zero thermal noise component $\sigma^2_{\text{th}} > 0$ ensures strictly positive definiteness. Hence, assuming that we have L_T input symbols for every block of $L_S M$ observations, we have that

$$\begin{aligned} C(\mathbf{A}, \mathbf{v}) &= \frac{1}{2L_T}\left(\lim_{N\to\infty}\frac{L_T}{N}\log_2\left|\mathbf{A}_{(N)}\mathbf{A}^H_{(N)} + \mathbf{R}_{\mathbf{v}_{(N)}}\right|\right. \\ &\qquad\qquad \left. - \lim_{N\to\infty}\frac{L_T}{N}\log_2\left|\mathbf{R}_{\mathbf{v}_{(N)}}\right|\right) \\ &\approx \frac{1}{2L_T}\left(\frac{1}{2\pi}\int_0^{2\pi}\log_2\left|\varphi^A(e^{i\theta}) + \varphi^R(e^{i\theta})\right|d\theta\right. \\ &\qquad\qquad \left. -\frac{1}{2\pi}\int_0^{2\pi}\log_2\left|\varphi^R(e^{i\theta})\right|d\theta\right). \end{aligned}$$

Note that we indicate an approximation here, since $\mathbf{A}_{(N)}\mathbf{A}^H_{(N)}$ is not truly block Toeplitz. We speculate, however, that equality indeed holds in the limit as the number of transmitted symbols N goes to infinity. We then conclude that

$$C(\mathbf{A}, \mathbf{v}) \approx \frac{1}{2L_T}\left(\frac{1}{2\pi}\int_0^{2\pi}\log_2\frac{\left|\varphi^A(e^{i\theta}) + \varphi^R(e^{i\theta})\right|}{|\varphi^R(e^{i\theta})|}d\theta\right). \tag{5.8}$$

Note that both $\left|\varphi^A(e^{i\theta}) + \varphi^R(e^{i\theta})\right|$ and $\left|\varphi^R(e^{i\theta})\right|$ are positive real values for every θ, and $\left|\varphi^A(e^{i\theta}) + \varphi^R(e^{i\theta})\right| > \left|\varphi^R(e^{i\theta})\right|$, so that the log of the ratio is positive for all θ. Furthermore, the expression within parentheses in Equation (5.8) is simply the average of the integrand over θ. This average can be approximated using the Fast Fourier Transform (FFT).

To this end, let $\varphi^A_K[\omega]$ and $\varphi^R_K[\omega]$ indicate the Discrete Fourier Transforms (DFT) of truncated portions of the sequences $\varphi^A[t]$ and $\varphi^R[t]$, as defined in Equation (5.6). Then we will approximate the mutual information by

choosing some sufficiently large K and computing

$$C(\mathbf{A}, \mathbf{v}) \approx \frac{1}{2L_T} \left(\frac{1}{2K+1} \sum_{\omega=-K}^{K} \log_2 \frac{\left| \varphi_K^A[\omega] + \varphi_K^R[\omega] \right|}{|\varphi_K^R[\omega]|} \right).$$

Next, we consider the computation of the LMMSE system metric. We will simplify under the assumption that $\mathbf{R}_{\mathbf{x}_{(N)}}$ is identity. We then have that

$$\mathbf{R}_{\mathbf{y}_{(N)}} = \mathbf{A}_{(N)} \mathbf{A}_{(N)}^H + \mathbf{R}_{\mathbf{v}_{(N)}}.$$

Therefore, we have that

$$\begin{aligned} D(\mathbf{A}, \mathbf{v}) &= 1 - \lim_{N \to \infty} \frac{1}{N} \mathrm{tr} \left(\mathbf{A}_{(N)}^H \mathbf{R}_{\mathbf{y}_{(N)}}^{-1} \mathbf{A}_{(N)} \right) \\ &= 1 - \lim_{N \to \infty} \frac{1}{N} \mathrm{tr} \left(\mathbf{R}_{\mathbf{y}_{(N)}}^{-1} \mathbf{A}_{(N)} \mathbf{A}_{(N)}^H \right) \\ &= 1 - \lim_{N \to \infty} \frac{1}{N} \mathrm{tr} \left(\left(\mathbf{A}_{(N)} \mathbf{A}_{(N)}^H + \mathbf{R}_{\mathbf{v}_{(N)}} \right)^{-1} \mathbf{A}_{(N)} \mathbf{A}_{(N)}^H \right). \end{aligned} \tag{5.9}$$

As before, we have that $\mathbf{A}_{(N)} \mathbf{A}_{(N)}^H$ is approximately block Toeplitz, composed of blocks of size $L_S M \times L_S M$, and $\mathbf{R}_{\mathbf{v}_{(N)}}$ is block Toeplitz with blocks of the same size. As such, $\mathbf{R}_{\mathbf{y}_{(N)}}^{-1}$ is approximately block Toeplitz, again with blocks of size $L_S M \times L_S M$, and hence the whole argument of the trace is approximately block Toeplitz with blocks of size $L_S M \times L_S M$. The whole expression is then simply one minus the average of the diagonal elements of the argument to the trace. Due to the approximately block Toeplitz property, we can approximate the average over the whole diagonal as simply the average over the diagonal of a single $L_S M \times L_S M$ block from the diagonal.

Thus, we would like to compute the $L_S M \times L_S M$ block that defines the diagonal of $\mathbf{R}_{\mathbf{y}_{(N)}}^{-1} \mathbf{A}_{(N)} \mathbf{A}_{(N)}^H$. This can be approximated by simply taking the $t = 0$ (matrix) element from the inverse DFT of $\left(\varphi_K^A[\omega] + \varphi_K^R[\omega] \right)^{-1} \varphi_K^A[\omega]$. This element is given as

$$\frac{1}{2K+1} \sum_{\omega=-K}^{K} \left(\varphi_K^A[\omega] + \varphi_K^R[\omega] \right)^{-1} \varphi_K^A[\omega]. \tag{5.10}$$

From this, the approximation of the LMMSE metric, using the FFT, is the

following:

$$D(\mathbf{A},\mathbf{v}) \approx 1 - \frac{1}{L_S M}\mathrm{tr}\left(\frac{1}{2K+1}\sum_{\omega=-K}^{K}\left(\varphi_K^A[\omega]+\varphi_K^R[\omega]\right)^{-1}\varphi_K^A[\omega]\right).$$

This can be simplified, such that the final approximation is the following:

$$D(\mathbf{A},\mathbf{v}) \approx 1 - \frac{1}{L_S M(2K+1)}\sum_{\omega=-K}^{K}\mathrm{tr}\left(\left(\varphi_K^A[\omega]+\varphi_K^R[\omega]\right)^{-1}\varphi_K^A[\omega]\right).$$

As with the mutual information metric, K should be chosen to be sufficiently large.

Finally, we can also use frequency domain methods for approximating the BER system metric. Recall that the BER metric is defined as

$$p_e(\mathbf{A},\mathbf{v}) = \lim_{N\to\infty} E_{\mathbf{x}_{(N)}}\left[\frac{1}{N}\sum_{i=1}^{N} Q\left(\frac{x[i]E[\hat{x}[i]|\mathbf{x}_{(N)}]}{\sigma_{(N)}[i]}\right)\right],$$

where $\hat{\mathbf{x}}_{(N)} = \{\hat{x}[0],\hat{x}[1],...,\hat{x}[N-1]\}$ is the LMMSE estimate of the sequence of input symbols $\mathbf{x}_{(N)} = \{x[0],x[1],...,x[N-1]\}$, $\mathbf{H}_{(N)} = \mathbf{A}_{(N)}^H\mathbf{R}_{\mathbf{y}_{(N)}}^{-1}$ is the LMMSE linear estimator at the receiver, $E[\hat{\mathbf{x}}_{(N)}|\mathbf{x}_{(N)}] = \mathbf{H}_{(N)}\mathbf{A}_{(N)}\mathbf{x}_{(N)}$ is the conditional expectation of the symbol estimates, given the symbols, $\mathbf{R}_{\hat{\mathbf{x}}_{(N)}|\mathbf{x}_{(N)}} = \mathbf{H}_{(N)}\mathbf{R}_{\mathbf{v}_{(N)}}\mathbf{H}_{(N)}^H$ is the covariance of $\hat{\mathbf{x}}_{(N)}$ given $\mathbf{x}_{(N)}$, and $(\sigma_{(N)}[i])^2$ are the diagonal elements of $\mathbf{R}_{\hat{\mathbf{x}}_{(N)}|\mathbf{x}_{(N)}}$. Our approximation method for the BER metric involves "filtering" a generated symbol sequence $\mathbf{x}_{(N)}$ with the matrix $\mathbf{H}_{(N)}\mathbf{A}_{(N)}$ to produce $E[\hat{x}[i]|\mathbf{x}_{(N)}]$, as well as determining the elements on the main diagonal of $\mathbf{R}_{\hat{\mathbf{x}}_{(N)}|\mathbf{x}_{(N)}}$. The fact that both $\mathbf{H}_{(N)}\mathbf{A}_{(N)}$ and $\mathbf{R}_{\hat{\mathbf{x}}_{(N)}|\mathbf{x}_{(N)}}$ are approximately block Toeplitz with blocks of size $L_T \times L_T$, and that the matrices used to construct them also have block-wise structure, will allow us to use frequency domain methods in our metric approximations.

The derivation of our computational method proceeds as follows: First, recall that the matrix sequence $\rho^A[t]$ is simply the sequence of (possibly non-square) matrices that compose the channel matrix $\mathbf{A}$. Note that we do not refer to the matrix $\mathbf{A}$ as block Toeplitz, since this designation is only for matrices composed of square blocks. We will compute $\rho_K^A[\omega]$ as the length

$2K+1$ DFT of a subsequence of $\rho^A[t]$, as in Equation (5.6), such that

$$\rho_K^A[\omega] = \sum_{t=-K}^{K} \exp\left(\frac{-i2\pi\omega t}{2K+1}\right) \rho^A[t].$$

We will also compute the following DFT sequences:

$$\begin{aligned}
\varphi_K^{\mathbf{y}}[\omega] &= \varphi_K^A[\omega] + \varphi_K^R[\omega] \\
\varphi_K^{R\hat{\mathbf{x}}}[\omega] &= \left(\rho_K^A[\omega]\right)^H \left(\varphi_K^{\mathbf{y}}[\omega]\right)^{-1} \varphi_K^R[\omega] \left(\varphi_K^{\mathbf{y}}[\omega]\right)^{-1} \rho_K^A[\omega] \\
\varphi_K^{E\hat{\mathbf{x}}}[\omega] &= \left(\rho_K^A[\omega]\right)^H \left(\varphi_K^{\mathbf{y}}[\omega]\right)^{-1} \rho_K^A[\omega].
\end{aligned}$$

Note that these operations are directly analogous to the corresponding matrix operations, where the multiplication of large matrices turns into small multiplications separately for each "frequency" parameter ω, and these operations follow from our previous discussion on sequences of matrices, their frequency domain representations, convolution, and large block-wise structured matrices.

From this, we can compute the sequence of standard deviations $\sigma[j]$, such that

$$\sigma[j] = \sqrt{\frac{1}{2K+1} \sum_{\omega=-K}^{K} \left[\varphi_K^{R\hat{\mathbf{x}}}[\omega]\right]_{j,j}}$$

for $0 \leq j < L_T$. Note that this computation parallels the construction in Equation (5.10), since, in both cases, we are interested in the diagonal elements of some large matrix. In this case, the values $(\sigma[j])^2$ for each j correspond with the diagonal elements of $\mathbf{R}_{\hat{\mathbf{x}}_{(N)}|\mathbf{x}_{(N)}}$. Again, as with the matrix $\mathbf{R}_{\mathbf{y}_{(N)}}^{-1}\mathbf{A}_{(N)}\mathbf{A}_{(N)}^H$ that results in Equation (5.10), the matrix $\mathbf{R}_{\hat{\mathbf{x}}_{(N)}|\mathbf{x}_{(N)}}$ approximately has a block Toeplitz structure, which leads to this expression for the elements on the main diagonal.

We now need an efficient method for "filtering" a generated symbol sequence $\mathbf{x}_{(N)}$ with the matrix $\mathbf{H}_{(N)}\mathbf{A}_{(N)}$ to produce $E[\hat{\mathbf{x}}_{(N)}|\mathbf{x}_{(N)}]$. This involves computing a sequence of matrices $\Lambda_K^{E\hat{\mathbf{x}}}[t]$ as the inverse DFT of $\varphi_K^{E\hat{\mathbf{x}}}[\omega]$, such that

$$\Lambda_K^{E\hat{\mathbf{x}}}[t] = \frac{1}{2K+1} \sum_{\omega=-K}^{K} \exp\left(\frac{i2\pi\omega t}{2K+1}\right) \varphi_K^{E\hat{\mathbf{x}}}[\omega],$$

where $-K \leq t \leq K$. This sequence of matrices approximates a subsequence of the matrices that compose the blocks of $\mathbf{H}_{(N)}\mathbf{A}_{(N)}$—the matrix used for

computing $E[\hat{\mathbf{x}}_{(N)}|\mathbf{x}_{(N)}]$ from a sequence of transmitted symbols. Now, let $\mathbf{x}[t]$ be a sequence of random length L_T column vectors for $-K \leq t \leq K$ such that each $[\mathbf{x}[t]]_i$ is IID Rademacher distributed, i.e., $\mathrm{P}\left([\mathbf{x}[t]]_i = -1\right) = \mathrm{P}\left([\mathbf{x}[t]]_i = +1\right) = \frac{1}{2}$. Then we have that

$$p_e(\mathbf{A}, \mathbf{v}) \approx E_{\mathbf{x}[\cdot]}\left[\frac{1}{L_T}\sum_{i=1}^{L_T} Q\left(\frac{[\mathbf{x}[0]]_i}{\sigma[i]}\left[\sum_{t=-K}^{K} \Lambda_K^{E\hat{\mathbf{x}}}[t]\mathbf{x}[0-t]\right]_i\right)\right]. \tag{5.11}$$

Note the filtering operation that occurs in the inner summation, which corresponds with the approximation of $E[\hat{\mathbf{x}}_{(N)}|\mathbf{x}_{(N)}]$. Approximating the expectation in Equation (5.11) is accomplished by convolution of the matrix sequence $\Lambda_K^{E\hat{\mathbf{x}}}[t]$ with the length L_x (possibly much larger than $2K+1$) random symbol sequence $\mathbf{x}[t]$, multiplication of each resultant value with the corresponding $\frac{[\mathbf{x}[t]]_i}{\sigma[i]}$ value, followed by application of the $Q(\cdot)$ function, and finally averaging all of the Q values. More precisely, define the convolution sequence $\hat{\mathbf{x}}[t]$ as follows:

$$\hat{\mathbf{x}}[t] = \sum_{\tau=-K}^{K} \Lambda_K^{E\hat{\mathbf{x}}}[\tau]\mathbf{x}[t-\tau].$$

Then, for some large simulation length L_x, we have that

$$p_e(\mathbf{A}, \mathbf{v}) \approx \frac{1}{L_T L_x}\sum_{i=1}^{L_T}\sum_{t=1}^{L_x} Q\left(\frac{[\mathbf{x}[t]]_i\,[\hat{\mathbf{x}}[t]]_i}{\sigma[i]}\right).$$

Finally, we will note that it may be possible to reduce the variance of this estimate by generating binary De Bruijn sequences for $\mathbf{x}[t]$, rather than simply randomly generating the sequence of transmitted symbols, and then computing $\hat{\mathbf{x}}[t]$ using a cyclic convolution.

5.5 Quantized System Model

We will now consider communication systems with quantization. This will involve a system model similar to the one presented in Section 5.2, but with some notable differences. First, we will assume that there is no noise introduced in the channel. The noise introduced before sampling will remain,

however. The reason for removing the channel component of the noise is to ensure that the noise components for each observation are independent, which will simplify our analysis. The other modification is that we will apply scalar quantization to the output of each sampler in the SGC model. Specifically, for each SGC branch we have a function $d_k(y)$ that maps a real number $y \in \mathbb{R}$ to a quantization value $d_k(y) \in \{0, ..., D_k - 1\}$. Furthermore, we will assume that there are quantization thresholds $\ell_k[j]$, with $0 \leq j \leq D_k$, $\ell_k[j] < \ell_k[j+1]$, $\ell_k[0] = -\infty$, and $\ell_k[D_k] = \infty$, such that

$$d_k(y) = j \text{ such that } \ell_k[j] < y \leq \ell_k[j+1].$$

We allow the levels $\ell_k[j]$ and the quantization set sizes D_k to be different on each SGC branch.

It will now be more convenient to work with a multiple input, multiple output mathematical model of the system. First, define the sequence of transmitted symbols $\mathbf{x}[t]$ such that

$$[\mathbf{x}[t]]_i = x[L_T t + i],$$

where $0 \leq i < L_T$. In words, we group the input symbols $x[\cdot]$ into groups of size L_T. Next, define the unquantized observation sequence $\mathbf{y}[t]$ such that

$$\begin{aligned}[\mathbf{y}[t]]_i &= y_{\mathrm{mod}(i,M)}\left[L_S t + \left\lfloor \frac{i}{M} \right\rfloor\right] \\ &= [\mathbf{y}]_{(L_S M t + i)}.\end{aligned}$$

In words, we group the observations $\mathbf{y}$ into groups of size $L_S M$. We do the same with the thermal noise component $\mathbf{w}_{\mathrm{th}}$ when we define the sequence $\mathbf{w}_{\mathrm{th}}[t]$, such that

$$[\mathbf{w}_{\mathrm{th}}[t]]_i = [\mathbf{w}_{\mathrm{th}}]_{(L_S M t + i)}.$$

Finally, we define the quantized observation sequence $\mathbf{z}[t]$ obtained by applying a vector quantization function on the observation vectors $\mathbf{y}[t]$. Specifically, we have that

$$\mathbf{z}[t] = \mathbf{d}(\mathbf{y}[t]),$$

where $[\mathbf{d}(\mathbf{y}[t])]_i = d_i([\mathbf{y}[t]]_i)$. Then, using the definition of $\rho^A[t]$ from Equa-

tion (5.7), we have that

$$\mathbf{z}[t] = \mathbf{d}\left(\mathbf{w}[t] + \sum_{k=-\infty}^{\infty} \rho^A[k]\mathbf{x}[t-k]\right).$$

We will furthermore assume that the sequence of channel matrices $\rho^A[t]$ is causal with finite support, such that $\rho^A[t] = 0$ for $t \notin \{0, ..., L_\rho - 1\}$. Then we have that

$$\mathbf{z}[t] = \mathbf{d}\left(\mathbf{w}[t] + \sum_{k=0}^{L_\rho - 1} \rho^A[k]\mathbf{x}[t-k]\right). \tag{5.12}$$

5.6 Metrics for Quantized Systems

We will now consider bit error rate (BER) and mutual information (MI) as system level metrics for systems that can be modeled as in Equation (5.12). We begin with the BER metric. In this case, we assume that each symbol vector $\mathbf{x}[t]$ will be estimated based on a window of quantized observations $\mathbf{z}[\tau]$, with $t + \tau_s \leq \tau < t + \tau_e$. We also assume that the transmitted symbols are $[\mathbf{x}[t]]_i \in \{-1, +1\}$ with equal probability. Then we have that our symbol estimates $\hat{\mathbf{x}}[t]$ are given by some decision function $\Delta(\cdot)$, i.e.,

$$\hat{\mathbf{x}}[t] = \Delta(\mathbf{z}[t + \tau_s], ..., \mathbf{z}[t + \tau_e - 1]),$$

or, equivalently,

$$[\hat{\mathbf{x}}[t]]_i = \Delta_i(\mathbf{z}[t + \tau_s], ..., \mathbf{z}[t + \tau_e - 1]),$$

for $0 \leq i < L_T$. For shorthand, we will define

$$\mathbf{Z}[t] = \{\mathbf{z}[t + \tau_s], ..., \mathbf{z}[t + \tau_e - 1]\}.$$

Then we have that $[\hat{\mathbf{x}}[t]]_i = \Delta_i(\mathbf{Z}[t])$. Now, this decision function $\Delta(\cdot)$ could take many forms. For example, [74, 90] considered linear equalizers by mapping the quantized values $[\mathbf{z}[t]]_i$ to representative real values between the corresponding quantization levels. Here, we will assume that each $\Delta_i(\cdot)$ takes the form of the BER optimal decision rule, given the window of observations.

To derive the method of computing the BER $p_e(\Delta)$ of the optimal decision

rule, first note that

$$\begin{aligned} p_e(\Delta) &= \frac{1}{L_T}\sum_{i=0}^{L_T-1} \mathrm{P}\left([\hat{\mathbf{x}}[t]]_i \neq [\mathbf{x}[t]]_i\right) \\ &= \frac{1}{L_T}\sum_{i=0}^{L_T-1} \mathrm{P}\left(\Delta_i(\mathbf{Z}[t]) \neq [\mathbf{x}[t]]_i\right) \\ &= \frac{1}{L_T}\sum_{i=0}^{L_T-1} \mathrm{P}\left(\Delta_i(\mathbf{Z}[0]) \neq [\mathbf{x}[0]]_i\right). \end{aligned} \tag{5.13}$$

Of course, we have that

$$\begin{aligned} \mathrm{P}\left(\Delta_i(\mathbf{Z}[0]) \neq [\mathbf{x}[0]]_i\right) &= \sum_{\mathbf{Z}[0]} \mathrm{P}\left(\Delta_i(\mathbf{Z}[0]) \neq [\mathbf{x}[0]]_i | \mathbf{Z}[0]\right) \mathrm{P}\left(\mathbf{Z}[0]\right) \\ &= \sum_{\mathbf{Z}[0]} \min_b \mathrm{P}\left([\mathbf{x}[0]]_i = b | \mathbf{Z}[0]\right) \mathrm{P}\left(\mathbf{Z}[0]\right), \end{aligned} \tag{5.14}$$

where the minimization is the result of using the BER optimal decision rule. Now, define $\mathbf{X}[t] = \{\mathbf{x}[t+\tau_s+1-L_\rho], ..., \mathbf{x}[t+\tau_e-1]\}$, and assume that $0 \in \{\tau_s+1-L_\rho, ..., \tau_e-1\}$. Then we have that

$$\begin{aligned} \mathrm{P}\left(\mathbf{Z}[0]\right) &= \sum_{\mathbf{x}[\cdot]} \mathrm{P}\left(\mathbf{Z}[0], \mathbf{x}[\cdot]\right) \\ &= \sum_{\mathbf{X}[0]} \mathrm{P}\left(\mathbf{Z}[0], \mathbf{X}[0]\right) \\ &= \underbrace{\sum_{\mathbf{X}[0]:[\mathbf{x}[0]]_i=-1} \mathrm{P}\left(\mathbf{Z}[0], \mathbf{X}[0]\right)}_{=\mathrm{P}\left(\mathbf{Z}[0],[\mathbf{x}[0]]_i=-1\right)} + \underbrace{\sum_{\mathbf{X}[0]:[\mathbf{x}[0]]_i=+1} \mathrm{P}\left(\mathbf{Z}[0], \mathbf{X}[0]\right)}_{=\mathrm{P}\left(\mathbf{Z}[0],[\mathbf{x}[0]]_i=+1\right)} \end{aligned} \tag{5.15}$$

Now, define the sequence $\mathbf{y}_s[t]$ as follows:

$$\mathbf{y}_s[t] = \sum_{k=0}^{L_\rho-1} \rho^A[k]\mathbf{x}[t-k].$$

Specifically, this is the noise-free component of the signal received at the data converter. If we define $\mathbf{Y}_s[t] = \{\mathbf{y}_s[t+\tau_s], ..., \mathbf{y}_s[t+\tau_e-1]\}$ and $\mathbf{Y}[t] = \{\mathbf{y}[t+\tau_s], ..., \mathbf{y}[t+\tau_e-1]\}$, then we see that $\mathrm{P}\left(\mathbf{Y}[t]|\mathbf{X}[t]\right)$ is a jointly Gaussian distribution with a mean of $\mathbf{Y}_s[t]$ (a deterministic function of $\mathbf{X}[t]$) and a covariance of $\sigma_{\mathrm{th}}^2\mathbf{I}$, where $\mathbf{I}$ is an appropriately sized identity matrix. This

allows us to conclude that

$$\begin{aligned}
&\mathrm{P}\left(\mathbf{Z}[0], \mathbf{X}[0]\right) \\
&\quad = \mathrm{P}\left(\mathbf{X}[0]\right) \mathrm{P}\left(\mathbf{Z}[0] | \mathbf{X}[0]\right) \\
&\quad = 2^{-L_T(\tau_e - \tau_s + L_\rho - 1)} \mathrm{P}\left(\mathbf{Z}[0] | \mathbf{Y}_s[0]\right), \qquad (5.16)
\end{aligned}$$

where we have that

$$\begin{aligned}
&\mathrm{P}\left(\mathbf{Z}[0] | \mathbf{Y}_s[0]\right) \\
&\quad = \prod_{\tau=\tau_s}^{(\tau_e-1)} \prod_{i=0}^{(L_S M-1)} \left[Q\left(\frac{\ell_{\mathrm{mod}(i,M)} [[\mathbf{z}[\tau]]_i] - [\mathbf{y}_s[\tau]]_i}{\sigma_{\mathrm{th}}} \right) \right. \\
&\qquad\qquad \left. - Q\left(\frac{\ell_{\mathrm{mod}(i,M)} [[\mathbf{z}[\tau]]_i + 1] - [\mathbf{y}_s[\tau]]_i}{\sigma_{\mathrm{th}}} \right) \right]. \qquad (5.17)
\end{aligned}$$

Now, going back to Equation (5.14), note that

$$\mathrm{P}\left([\mathbf{x}[0]]_i = b | \mathbf{Z}[0]\right) = \frac{\mathrm{P}\left(\mathbf{Z}[0], [\mathbf{x}[0]]_i = b\right)}{\mathrm{P}\left(\mathbf{Z}[0]\right)}.$$

However, from Equation (5.15), we have that

$$\mathrm{P}\left(\mathbf{Z}[0]\right) = \mathrm{P}\left([\mathbf{x}[0]]_i = -1, \mathbf{Z}[0]\right) + \mathrm{P}\left([\mathbf{x}[0]]_i = +1, \mathbf{Z}[0]\right), \qquad (5.18)$$

which then lets us conclude that

$$\mathrm{P}\left([\mathbf{x}[0]]_i = b | \mathbf{Z}[0]\right) = \frac{\mathrm{P}\left(\mathbf{Z}[0], [\mathbf{x}[0]]_i = b\right)}{\mathrm{P}\left(\mathbf{Z}[0], [\mathbf{x}[0]]_i = -1\right) + \mathrm{P}\left(\mathbf{Z}[0], [\mathbf{x}[0]]_i = +1\right)}. \qquad (5.19)$$

Then, the full procedure for computing the BER metric $p_e(\Delta)$ is as follows: Compute, Equation (5.13). This involves the computation of each $\mathrm{P}\left(\Delta_i(\mathbf{Z}[0]) \neq [\mathbf{x}[0]]_i\right)$. These are computed as in Equation (5.14). This furthermore requires $\mathrm{P}\left([\mathbf{x}[0]]_i = b | \mathbf{Z}[0]\right)$ and $\mathrm{P}\left(\mathbf{Z}[0]\right)$. The latter is computed as in Equation (5.18), whereas the former is computed as in Equation (5.19). Finally, both of these calculations rely on $\mathrm{P}\left(\mathbf{Z}[0], [\mathbf{x}[0]]_i = b\right)$ for $b \in \{-1, +1\}$, and these are computed as in Equations (5.15), (5.16), and (5.17).

We will now examine a mutual information metric for these quantized SGC

communication systems. Specifically, we are interested in

$$\begin{aligned} I(\mathbf{X};\mathbf{Z}) &= \lim_{n\to\infty} \frac{1}{nL_T} I(\mathbf{X}^n;\mathbf{Z}^n) \\ &= \lim_{n\to\infty} \frac{1}{nL_T} I(\mathbf{x}[0],...,\mathbf{x}[n-1];\mathbf{z}[0],...,\mathbf{z}[n-1]). \end{aligned} \tag{5.20}$$

Here, we have that

$$\mathbf{X}^n = \{\mathbf{x}[0],\mathbf{x}[1],...,\mathbf{x}[n-1]\}$$

and

$$\mathbf{Z}^n = \{\mathbf{z}[0],\mathbf{z}[1],...,\mathbf{z}[n-1]\}.$$

In order to approximate this mutual information metric, we will begin with the simulation-based method given in [91]. This simulation-based method is one for approximating the mutual information rate, e.g., bits per input symbol, in *finite-state source/channel models*. In such a model, we have that the sequences $\mathbf{X}^n$ and $\mathbf{Z}^n$, as well as a state sequence

$$\mathbf{S}^n = \{\mathbf{s}[-1],\mathbf{s}[0],\mathbf{s}[1],...,\mathbf{s}[n-1]\},$$

are related to each other according to a chain-structured probabilistic graphical model, such that

$$\mathrm{P}(\mathbf{X}^n,\mathbf{Z}^n,\mathbf{S}^n) = \mathrm{P}(\mathbf{s}[-1])\prod_{t=0}^{n-1}\mathrm{P}(\mathbf{x}[t],\mathbf{z}[t],\mathbf{s}[t]|\mathbf{s}[t-1]). \tag{5.21}$$

It is assumed that the transition probability $\mathrm{P}(\mathbf{x}[t],\mathbf{z}[t],\mathbf{s}[t]|\mathbf{s}[t-1])$ is independent of t, and that $\mathrm{P}(\mathbf{s}[t]|\mathbf{s}[-1]) > 0$ for all $\mathbf{s}[-1]$ and $\mathbf{s}[t]$ for all sufficiently large t. These conditions ensure that the random process is ergodic, and that the approximation method is valid. We will actually assume more structure in our work. Specifically, we will assume that

$$\mathrm{P}(\mathbf{x}[t],\mathbf{z}[t],\mathbf{s}[t]|\mathbf{s}[t-1]) \tag{5.22}$$

$$= \mathrm{P}(\mathbf{x}[t]|\mathbf{s}[t-1])\mathrm{P}(\mathbf{s}[t]|\mathbf{x}[t],\mathbf{s}[t-1])\mathrm{P}(\mathbf{z}[t]|\mathbf{s}[t],\mathbf{x}[t],\mathbf{s}[t-1]). \tag{5.23}$$

$$= \mathrm{P}(\mathbf{x}[t])\mathrm{P}(\mathbf{s}[t]|\mathbf{x}[t],\mathbf{s}[t-1])\mathrm{P}(\mathbf{z}[t]|\mathbf{s}[t]). \tag{5.24}$$

That (5.23) is equal to (5.22) follows from the rules of conditional probabil-

ity and is always true. The expression in (5.24) follows by making certain assumptions of conditional independence.

For the quantizing SGC modeled by Equation (5.12), we have that $\mathbf{x}[t]$ and $\mathbf{z}[t]$ are the same as defined for that model, and

$$\mathbf{s}[t] = \{\mathbf{x}[t], \mathbf{x}[t-1], ..., \mathbf{x}[t-L_\rho+1]\}.$$

The symbol distribution $\mathrm{P}(\mathbf{x}[t])$ is uniform on all values of $\mathbf{x}[t]$, the transition probability $\mathrm{P}(\mathbf{s}[t]|\mathbf{x}[t], \mathbf{s}[t-1])$ is a deterministic function of $\mathbf{x}[t]$ and $\mathbf{s}[t-1]$, and the observation condition probability $\mathrm{P}(\mathbf{z}[t]|\mathbf{s}[t])$ is apparent in Equation (5.12), and involves calculations similar to Equation (5.17).

The estimation procedure is simple. First, generate long sequences $\mathbf{X}^n$, $\mathbf{Z}^n$, and $\mathbf{S}^n$ according to the model in Equation (5.21). Next, compute $\mathrm{P}(\mathbf{X}^n)$, $\mathrm{P}(\mathbf{Z}^n)$, and $\mathrm{P}(\mathbf{X}^n, \mathbf{Z}^n)$ for the generated sequences. This can be done by using the BCJR algorithm [92], which is equivalent to the Sum-Product algorithm on the factor graph specified by the chain-structured probabilistic graphical model. In some cases, these quantities may be found exactly, such as if $\mathbf{x}[t]$ are IID random vectors (as with our quantizing SGC), or if the conditional entropy $h(\mathbf{Z}^n|\mathbf{X}^n)$ is known analytically. Finally, we conclude with

$$\hat{I}(\mathbf{X};\mathbf{Z}) = \frac{1}{nL_T}\left(\log \mathrm{P}(\mathbf{X}^n, \mathbf{Z}^n) - \log \mathrm{P}(\mathbf{X}^n) - \log \mathrm{P}(\mathbf{Z}^n)\right) \tag{5.25}$$

$$= \frac{1}{nL_T}\left(\log \mathrm{P}(\mathbf{X}^n, \mathbf{Z}^n) - \log \mathrm{P}(\mathbf{Z}^n)\right) - \frac{h(\mathbf{x}[t])}{L_T}. \tag{5.26}$$

Of course, whether Equation (5.25) or (5.26), or some other related expression, is used for the final estimate depends on which quantities can be computed directly, without simulation. In our case, where Equation (5.24) is satisfied, the algorithm for computing $\hat{I}(\mathbf{X};\mathbf{Z})$ proceeds as in Algorithm 5.1.

Now, when the mentioned conditions for validity of the algorithm are met, Algorithm 5.1 is guaranteed to converge to the true mutual information rate in the limit of n. However, note that this estimate depends on the randomly generated samples $\hat{\mathbf{x}}$, $\hat{\mathbf{s}}$, and $\hat{\mathbf{z}}$. In particular, consider drawing $\hat{\mathbf{z}} \sim \mathrm{P}(\mathbf{z}[t]|\hat{\mathbf{s}})$. If $\mathrm{P}(\mathbf{z}[t]|\hat{\mathbf{s}})$ is strongly peaked around a particular $\mathbf{z}[t]$ for every $\hat{\mathbf{s}}$, then we will need many draws from this distribution to have a representative drawing from the distribution (which includes a proportionate number of the low probability events), and likewise to have good estimates of $h_{\mathbf{Z}}$ and $h_{\mathbf{XZ}}$. This

Algorithm 5.1: Mutual Information Rate Approximation from [91].

Data: $\mathrm{P}(\mathbf{s}[-1])$, $\mathrm{P}(\mathbf{x}[t])$, $\mathrm{P}(\mathbf{s}[t]|\mathbf{x}[t], \mathbf{s}[t-1])$, $\mathrm{P}(\mathbf{z}[t]|\mathbf{s}[t])$, n
Result: $\hat{I}$ = estimated mutual information rate

1 Generate $\hat{\mathbf{s}}_{\text{prev}} \sim \mathrm{P}(\mathbf{s}[-1])$;
/* Initial beliefs */
2 Initialize $\mathrm{P}_{\mathbf{Z}}(\mathbf{s}) \leftarrow \mathrm{P}(\mathbf{s}[-1] = \mathbf{s})$ and $\mathrm{P}_{\mathbf{XZ}}(\mathbf{s}) \leftarrow \mathrm{P}(\mathbf{s}[-1] = \mathbf{s})$;
3 Initialize $h_{\mathbf{Z}} \leftarrow 0$ and $h_{\mathbf{XZ}} \leftarrow 0$;
4 **for** $t = 0, ..., n-1$ **do**
  /* Randomly generate next set of values */
5 Generate $\hat{\mathbf{x}} \sim \mathrm{P}(\mathbf{x}[t])$;
6 Generate $\hat{\mathbf{s}} \sim \mathrm{P}(\mathbf{s}[t]|\hat{\mathbf{x}}, \hat{\mathbf{s}}_{\text{prev}})$;
7 Generate $\hat{\mathbf{z}} \sim \mathrm{P}(\mathbf{z}[t]|\hat{\mathbf{s}})$;
  /* Forward Sum-Product message passing, update beliefs */
8 Compute $\hat{\mathrm{P}}_{\mathbf{Z}}(\tilde{\mathbf{s}}) = \sum_{\mathbf{x}} \sum_{\mathbf{s}} \mathrm{P}_{\mathbf{Z}}(\mathbf{s})\mathrm{P}(\mathbf{x})\mathrm{P}(\tilde{\mathbf{s}}|\mathbf{x}, \mathbf{s})$;
9 Compute $\check{\mathrm{P}}_{\mathbf{Z}}(\tilde{\mathbf{s}}) = \hat{\mathrm{P}}_{\mathbf{Z}}(\tilde{\mathbf{s}})\mathrm{P}(\hat{\mathbf{z}}|\tilde{\mathbf{s}})$;
10 Compute $\alpha_{\mathbf{Z}} = \sum_{\mathbf{s}} \check{\mathrm{P}}_{\mathbf{Z}}(\mathbf{s})$;
11 Update $\mathrm{P}_{\mathbf{Z}}(\mathbf{s}) \leftarrow \frac{1}{\alpha_{\mathbf{Z}}} \check{\mathrm{P}}_{\mathbf{Z}}(\mathbf{s})$;
  /* Update sequence entropy */
12 Update $h_{\mathbf{Z}} \leftarrow h_{\mathbf{Z}} - \log_2(\alpha_{\mathbf{Z}})$;
  /* Forward Sum-Product message passing, update beliefs */
13 Compute $\hat{\mathrm{P}}_{\mathbf{XZ}}(\tilde{\mathbf{s}}) = \sum_{\mathbf{s}} \mathrm{P}_{\mathbf{XZ}}(\mathbf{s})\mathrm{P}(\hat{\mathbf{x}})\mathrm{P}(\tilde{\mathbf{s}}|\hat{\mathbf{x}}, \mathbf{s})$;
14 Compute $\check{\mathrm{P}}_{\mathbf{XZ}}(\tilde{\mathbf{s}}) = \hat{\mathrm{P}}_{\mathbf{XZ}}(\tilde{\mathbf{s}})\mathrm{P}(\hat{\mathbf{z}}|\tilde{\mathbf{s}})$;
15 Compute $\alpha_{\mathbf{XZ}} = \sum_{\mathbf{s}} \check{\mathrm{P}}_{\mathbf{XZ}}(\mathbf{s})$;
16 Update $\mathrm{P}_{\mathbf{XZ}}(\mathbf{s}) \leftarrow \frac{1}{\alpha_{\mathbf{XZ}}} \check{\mathrm{P}}_{\mathbf{XZ}}(\mathbf{s})$;
  /* Update sequence entropy */
17 Update $h_{\mathbf{XZ}} \leftarrow h_{\mathbf{XZ}} - \log_2(\alpha_{\mathbf{XZ}})$;
18 **end**
/* Analytic computation of symbol entropy rate */
19 Initialize $h_{\mathbf{X}} \leftarrow -\sum_{\mathbf{x}} \mathrm{P}(\mathbf{x}) \log_2 \mathrm{P}(\mathbf{x})$;
20 **return** $\hat{I} \leftarrow \frac{1}{L_T} h_{\mathbf{X}} + \frac{1}{nL_T} (h_{\mathbf{Z}} - h_{\mathbf{XZ}})$;

is particularly of concern for our SGC application, because if it is not the case that $\mathrm{P}(\mathbf{z}[t]|\hat{\mathbf{s}})$ is strongly peaked, then we are wasting converter resolution and consequently needlessly consuming energy. This motivates us to explore improving on Algorithm 5.1.

Algorithm 5.2 is a modification of Algorithm 5.1 that provides improved approximation results in many circumstances. Essentially, rather than simply computing the entropy updates for $h_{\mathbf{Z}}$ (line 12) and $h_{\mathbf{XZ}}$ (line 17) based on the drawn value of $\hat{\mathbf{z}}$, we compute the expected update to each of these values with respect to the distribution from which $\hat{\mathbf{z}}$ is drawn. At this point,

Algorithm 5.2: Improved Mutual Information Rate Approximation.

Data: $\mathrm{P}(\mathbf{s}[-1])$, $\mathrm{P}(\mathbf{x}[t])$, $\mathrm{P}(\mathbf{s}[t]|\mathbf{x}[t], \mathbf{s}[t-1])$, $\mathrm{P}(\mathbf{z}[t]|\mathbf{s}[t])$, n
Result: $\hat{I}$ = estimated mutual information rate

Generate $\hat{\mathbf{s}}_{\text{prev}} \sim \mathrm{P}(\mathbf{s}[-1])$;
/* Initial beliefs */
Initialize $\mathrm{P}_{\mathbf{Z}}(\mathbf{s}) \leftarrow \mathrm{P}(\mathbf{s}[-1] = \mathbf{s})$ and $\mathrm{P}_{\mathbf{XZ}}(\mathbf{s}) \leftarrow \mathrm{P}(\mathbf{s}[-1] = \mathbf{s})$;
Initialize $h_{\mathbf{Z}} \leftarrow 0$ and $h_{\mathbf{XZ}} \leftarrow 0$;
for $t = 0, ..., n-1$ **do**
  /* Randomly generate next set of values */
  Generate $\hat{\mathbf{x}} \sim \mathrm{P}(\mathbf{x}[t])$;
  Generate $\hat{\mathbf{s}} \sim \mathrm{P}(\mathbf{s}[t]|\hat{\mathbf{x}}, \hat{\mathbf{s}}_{\text{prev}})$;
  Generate $\hat{\mathbf{z}} \sim \mathrm{P}(\mathbf{z}[t]|\hat{\mathbf{s}})$;
  /* Forward Sum-Product message passing, update beliefs */
  Compute $\hat{\mathrm{P}}_{\mathbf{Z}}(\tilde{\mathbf{s}}) = \sum_{\mathbf{x}} \sum_{\mathbf{s}} \mathrm{P}_{\mathbf{Z}}(\mathbf{s}) \mathrm{P}(\mathbf{x}) \mathrm{P}(\tilde{\mathbf{s}}|\mathbf{x}, \mathbf{s})$;
  Compute $\check{\mathrm{P}}_{\mathbf{Z}}(\tilde{\mathbf{s}}) = \hat{\mathrm{P}}_{\mathbf{Z}}(\tilde{\mathbf{s}}) \mathrm{P}(\hat{\mathbf{z}}|\tilde{\mathbf{s}})$;
  Compute $\alpha_{\mathbf{Z}} = \sum_{\mathbf{s}} \check{\mathrm{P}}_{\mathbf{Z}}(\mathbf{s})$;
  Update $\mathrm{P}_{\mathbf{Z}}(\mathbf{s}) \leftarrow \frac{1}{\alpha_{\mathbf{Z}}} \check{\mathrm{P}}_{\mathbf{Z}}(\mathbf{s})$;
  /* Update sequence entropy */
  Update $h_{\mathbf{Z}} \leftarrow h_{\mathbf{Z}} - \left(\sum_{\tilde{\mathbf{z}}} \mathrm{P}(\tilde{\mathbf{z}}|\hat{\mathbf{s}}) \log_2 \left(\sum_{\tilde{\mathbf{s}}} \hat{\mathrm{P}}_{\mathbf{Z}}(\tilde{\mathbf{s}}) \mathrm{P}(\tilde{\mathbf{z}}|\tilde{\mathbf{s}}) \right) \right)$;
  /* Forward Sum-Product message passing, update beliefs */
  Compute $\hat{\mathrm{P}}_{\mathbf{XZ}}(\tilde{\mathbf{s}}) = \sum_{\mathbf{s}} \mathrm{P}_{\mathbf{XZ}}(\mathbf{s}) \mathrm{P}(\hat{\mathbf{x}}) \mathrm{P}(\tilde{\mathbf{s}}|\hat{\mathbf{x}}, \mathbf{s})$;
  Compute $\check{\mathrm{P}}_{\mathbf{XZ}}(\tilde{\mathbf{s}}) = \hat{\mathrm{P}}_{\mathbf{XZ}}(\tilde{\mathbf{s}}) \mathrm{P}(\hat{\mathbf{z}}|\tilde{\mathbf{s}})$;
  Compute $\alpha_{\mathbf{XZ}} = \sum_{\mathbf{s}} \check{\mathrm{P}}_{\mathbf{XZ}}(\mathbf{s})$;
  Update $\mathrm{P}_{\mathbf{XZ}}(\mathbf{s}) \leftarrow \frac{1}{\alpha_{\mathbf{XZ}}} \check{\mathrm{P}}_{\mathbf{XZ}}(\mathbf{s})$;
  /* Update sequence entropy */
  Update $h_{\mathbf{XZ}} \leftarrow h_{\mathbf{XZ}} - \left(\sum_{\tilde{\mathbf{z}}} \mathrm{P}(\tilde{\mathbf{z}}|\hat{\mathbf{s}}) \log_2 \left(\sum_{\tilde{\mathbf{s}}} \hat{\mathrm{P}}_{\mathbf{XZ}}(\tilde{\mathbf{s}}) \mathrm{P}(\tilde{\mathbf{z}}|\tilde{\mathbf{s}}) \right) \right)$;
end
/* Analytic computation of symbol entropy rate */
Initialize $h_{\mathbf{X}} \leftarrow -\sum_{\mathbf{x}} \mathrm{P}(\mathbf{x}) \log_2 \mathrm{P}(\mathbf{x})$;
return $\hat{I} \leftarrow \frac{1}{L_T} h_{\mathbf{X}} + \frac{1}{n L_T} (h_{\mathbf{Z}} - h_{\mathbf{XZ}})$;

we will simply note that it is possible to extend this concept, such that the updates of $h_{\mathbf{Z}}$ and $h_{\mathbf{XZ}}$ start from a point even earlier in the generated random process sequence. We show in Algorithm 5.2 the method that uses the expectation of $\log_2(\alpha)$ with respect to only the sampling of $\hat{\mathbf{z}}$, but we could extend this expectation to also average over the possible draws of $\hat{\mathbf{s}}$, $\hat{\mathbf{x}}$, or even draws from previous iterations of the for loop. As we take more draws going backwards in the sequence into account in this expectation, the computational complexity of the entropy updates grows exponentially, but up

to a point this is compensated by the corresponding decrease in the sequence length required to achieve a certain level of accuracy.

The authors of [91] state that it is generally impractical with Algorithm 5.1 to realize more than a couple digits of accuracy in the mutual information estimate. This becomes particularly pronounced when the mutual information rate is close to the entropy rate of the source process $\mathbf{x}[t]$, or equivalently, when $\frac{1}{nL_T}\left(h_{\mathbf{Z}} - h_{\mathbf{XZ}}\right)$ is small, which is the case when $\mathrm{P}(\mathbf{z}[t]|\hat{\mathbf{s}})$ is strongly peaked. In these cases, Algorithm 5.2 provides significantly more accurate estimates of $\frac{1}{nL_T}\left(h_{\mathbf{Z}} - h_{\mathbf{XZ}}\right)$ using far shorter simulation lengths n.

5.7 Converter Performance: System Level versus Traditional Metrics

In this section, we present results of evaluating various data converter architectures under both system level metrics and traditional ADC metrics. We begin by studying time-interleaved analog-to-digital converters (TIADCs), showing that the conventional wisdom from the traditional metrics for the design of these converters is misleading for the design of communications specific TIADCs. We then make similar conclusions for a statistics gathering converter (SGC) utilizing a delay-line architecture. The goal of this section is to motivate the use of the information-based system-level metrics for the design of communication system data converters, as opposed to the traditional waveform-centric metrics.

5.7.1 Time-Interleaved Analog-to-Digital Converter

Input/Output Mutual Information

We begin with TIADCs, which can be thought of as a special case of SGCs. We show a model of a TIADC in Figure 5.3. In this section, we do not include quantization. Of course, as with non-quantizing SGCs in general, the full TIADC communication system has a linear input-output relationship. More specifically, consider a finite vector of symbols $\mathbf{x} = (x[0], ..., x[N-1])^T$ and an observation vector $\mathbf{y} = (y[0], ..., y[L-1])$. It is possible to relate these

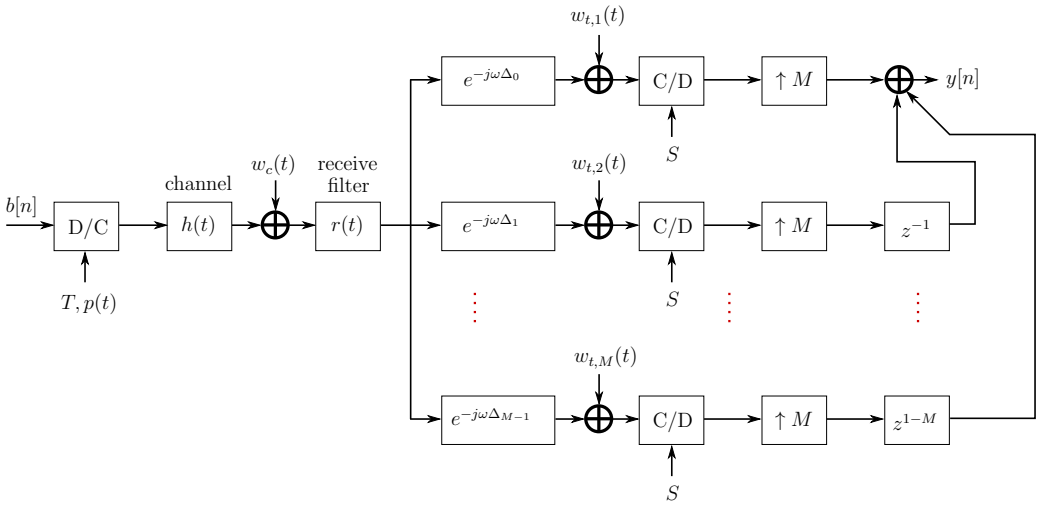


Figure 5.3: An M-fold time-interleaved receiver structure.

input and output vectors according to the relation

$$\mathbf{y} = \mathbf{A}(\mathbf{D})\mathbf{x} + \mathbf{w}_{\text{th}} + \mathbf{w}_{\text{ch}}. \tag{5.27}$$

In (5.27), $\mathbf{A}(\mathbf{D})$ is a channel matrix that depends on the relative sampling time delays of the ADC branches, $\mathbf{D} = (\Delta_0, ..., \Delta_{M-1})^T$, $\mathbf{w}_{\text{th}}$ is a vector of stationary independent Gaussian noise samples that arise from the bandlimited sampling (due to the integration time of the sample and hold circuitry) of $w_{t,j}(t)$, and $\mathbf{w}_{\text{ch}}$ is a vector of noise samples that arise from sampling the output of the receive shaping filter due to the channel noise, $w_c(t)$. The matrix $\mathbf{A}(\mathbf{D})$ is derived from the channel model as follows. First, we define the aggregate pulse shape as the three-fold convolution $q(t) = (p * h * r)(t)$. Now, we define the vector $\mathbf{q}(\mathbf{V})$ as $[\mathbf{q}(\mathbf{V})]_i = q([\mathbf{V}]_i)$, where $\mathbf{V}$ is a vector of times. Then we have that

$$\mathbf{A}(\mathbf{D}) = \begin{bmatrix} \mathbf{q}(\mathbf{D}) & \mathbf{q}(\mathbf{D}–T) & ... & \mathbf{q}(\mathbf{D}–(N–1)T) \\ \mathbf{q}(S+\mathbf{D}) & \mathbf{q}(S+\mathbf{D}–T) & ... & \mathbf{q}(S+\mathbf{D}-(N–1)T) \\ \mathbf{q}(2S+\mathbf{D}) & \mathbf{q}(2S+\mathbf{D}–T) & ... & \mathbf{q}(2S+\mathbf{D}–(N–1)T) \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \tag{5.28}$$

When we add (subtract) a scalar to (from) a vector, we mean that the scalar value should be added to (subtracted from) each element in the vector. The quantity $\frac{1}{S}$ in (5.28) corresponds with the sampling rate of a single branch of the time-interleaved ADC. To characterize the noise, we assume that each of

the noise sources is an independent Gaussian random processes. For the thermal noise component, we have that the noise covariance matrix is $\sigma_{\text{th}}^2 \mathbf{I}_{L\times L}$. In order to derive the covariance matrix of the channel noise, using the vector of delays $\mathbf{D} = (\Delta_1, ..., \Delta_M)^T$ and the vector $\mathbf{1}_{n\times m}$, the $n \times m$ matrix of ones, we define the matrix $\mathcal{T}_\mathcal{D}$ of pairwise time differences as

$$\mathcal{T}_\mathcal{D} = \mathbf{D}\mathbf{1}_{1\times M} - \mathbf{1}_{M\times 1}\mathbf{D}^T.$$

Let $\sigma_{\text{ch}}^2 R(\tau)$ be the autocorrelation function of the filtered channel noise, normalized such that $R(0) = 1$. We now construct a block Toeplitz matrix $\mathcal{T} \in \Re^{L\times L}$ of time differences as

$$\mathcal{T} = \begin{bmatrix} \mathcal{T}_\mathcal{D} & \mathcal{T}_\mathcal{D} - S\mathbf{1}_{M\times M} & \mathcal{T}_\mathcal{D} - 2S\mathbf{1}_{M\times M} & \cdots \\ \mathcal{T}_\mathcal{D} + S\mathbf{1}_{M\times M} & \mathcal{T}_\mathcal{D} & \mathcal{T}_\mathcal{D} - S\mathbf{1}_{M\times M} & \cdots \\ \mathcal{T}_\mathcal{D} + 2S\mathbf{1}_{M\times M} & \mathcal{T}_\mathcal{D} + S\mathbf{1}_{M\times M} & \mathcal{T}_\mathcal{D} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

Then the channel noise covariance matrix is given by $E\{\mathbf{w}_{\text{ch}}\mathbf{w}_{\text{ch}}^T\} = \sigma_{\text{ch}}^2 R(\mathcal{T})$, where the autocorrelation is evaluated element-wise on the matrix $\mathcal{T}$. Finally, since the channel noise and thermal noise processes are assumed independent, we have that the composite noise covariance is given by $\mathbf{R}_{\mathbf{v}_{(N)}}(\mathbf{D}) = \sigma_{\text{th}}^2 \mathbf{I}_{L\times L} + \sigma_{\text{ch}}^2 R(\mathcal{T})$, where $\mathbf{R}_{\mathbf{v}_{(N)}}(\mathbf{D}) = E\{(\mathbf{w}_{\text{ch}} + \mathbf{w}_{\text{th}})(\mathbf{w}_{\text{ch}} + \mathbf{w}_{\text{th}})^T\}$.

In order to gain some insight into the potential effects of different values of $\mathbf{D}$ on the effectiveness of the ADC samples for communication, we examine the input to output mutual information, per channel use, as a function of the delays $\mathbf{D}$. As such, we investigate the following:

$$C(\mathbf{D}) \triangleq \lim_{N\to\infty} \frac{1}{N} I(\mathbf{b}^N; \mathbf{y}^L), \text{ if the limit exists.}$$

In Equation (5.7.1), it is implicit that the observation vector $\mathbf{y}^L$ depends on the particular choice of $\mathbf{D}$, and the number of observations depends on the number of symbols. As in Section 5.3, we assume that the distribution of input symbols is fixed such that each symbol is an independent Gaussian random value with mean zero and unit variance. As in Equation (5.3), we

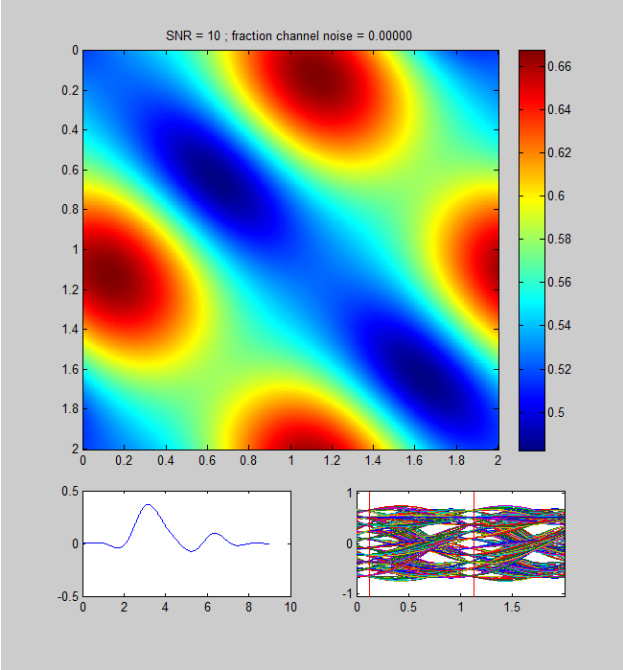


Figure 5.4: $C(\mathbf{D})$ for symbol-rate sampling at $SNR = 10$ and 0% channel noise.

have that

$$C(\mathbf{D}) = \lim_{N\to\infty} \frac{1}{2N} \left(\log_2 \frac{\left| \mathbf{A}(\mathbf{D})\mathbf{A}(\mathbf{D})^H + \mathbf{R}_{\mathbf{v}_{(N)}}(\mathbf{D}) \right|}{\left| \mathbf{R}_{\mathbf{v}_{(N)}}(\mathbf{D}) \right|} \right),$$

which can be approximated using any of the methods discussed in previous sections.

Based on our model of the time-interleaved analog-to-digital converter, there are three natural symmetry properties that we expect to observe for $C(\mathbf{D})$. We will refer to these as permutation symmetry, symbol phase symmetry, and branch phase symmetry. Permutation symmetry means that $C(\mathbf{D}) = C(\mathbf{P}_\pi \mathbf{D})$, where π is any permutation, and $\mathbf{P}_\pi$ is the corresponding permutation matrix. Symbol phase symmetry means that $C(\mathbf{D}) = C(\mathbf{D}+T)$. Finally, branch phase symmetry means that $C(\mathbf{D}) = C(\mathbf{D} + S\mathbf{e}_i)$ for any i, where $\mathbf{e}_i$ is an $M \times 1$ vector with 1 in the i^{th} position and 0 in every other position.

Figure 5.4 shows a basic situation for a two-channel time-interleaved ADC in which there is only circuit noise and the signal to noise ratio is 10dB.

As of 1999, the overall circuit noise-induced SNR due to input referred-thermal noise, aperture jitter due to uncertainty in the sampling time, and comparator ambiguity due to regeneration time constants from the integrated circuit fabric, was around 20dB for converters in the 10Gs/s regime and would be around 10dB for converters near 40Gs/s [87]. The horizontal axis of the upper subplot is Δ_0 and the vertical axis of the upper subplot is Δ_1. The setup uses values of $T = 1$ and $S = 2$. Hence, this is a symbol-rate sampling converter. We note that the points of maximum mutual information (per input symbol) lie on the diagonals where $|\Delta_0 - \Delta_1| = 1$, i.e., the optimal sampling scheme is equispaced sampling, i.e. one sample every $T = 1$ second. We can also observe the symmetries mentioned. Permutation symmetry is evident in the reflection symmetry across the $\Delta_0 = \Delta_1$ diagonal. Symbol phase symmetry is also evident, i.e., $C(\mathbf{D}) = C(\mathbf{D} + \mathbf{1})$. Branch phase symmetry is evident in the appearance of periodic boundary conditions in the figures. Furthermore, these symmetries, in combination, result in reflection symmetry across the $\Delta_0 = \Delta_1 \pm 1$ diagonals. The lower left subplot shows the function $q(t)$ for this channel, as defined above. The lower right subplot shows the eye diagram for the channel, as derived from $q(t)$, as well as the optimal sampling points.

Figure 5.5 shows a similar situation to Figure 5.4, where the difference is that we now have 99% of the noise power coming from noise sources in the channel. Again, the optimal sampling scheme, with respect to the choice of ADC branch delays, is equispaced sampling. The same symmetry properties observed in Figure 5.4 are also visible. What we note here is that while equispaced samples seem to be the optimal choice within our setup for symbol-rate sampling—irrespective of the choice of channel, SNR, or how much of the noise power is due to channel sources versus circuit noise—we lose very little by imprecisely choosing the ADC branch delays. In fact, by allowing the branch delays to be off by up to plus or minus 10% of a symbol period, we only decrease our achievable data rate by approximately 2% or 3% in the worst case.

Figures 5.6 and 5.7 show a setup with oversampling by a factor of 2. The channel is the same as that used to produce Figures 5.4 and 5.5; we set $SNR = 10$dB, and $T = 1$. However, corresponding with the oversampling setup, we have that $S = 1$. Figure 5.6 visualizes $C(\mathbf{D})$ where all of the noise power comes from circuit sources. We observe the symmetries in these figures

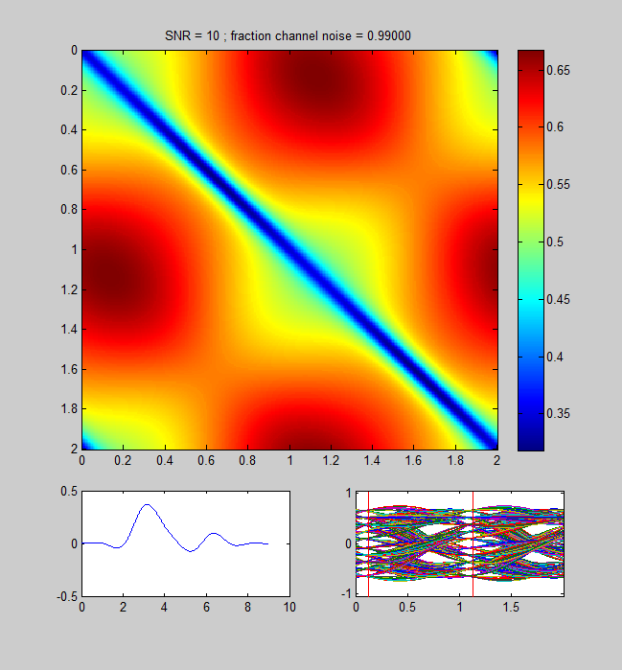


Figure 5.5: $C(\mathbf{D})$ for symbol-rate sampling at $SNR = 10$ and 99% channel noise.

as well. However, in this case the sample phase symmetry is redundant, since the branch phase symmetry accounts for it when $S = T$. Of particular interest is that the value of $\mathbf{D}$ that maximizes $C(\mathbf{D})$ in this case is on the diagonal where $\Delta_1 = \Delta_2$, which is contrary to the conventional wisdom that the samples should be equispaced.

Now, Figure 5.7 illustrates a channel noise dominated case, where 90% of the noise power comes from the channel. Here, we see that the optimal $\mathbf{D}$ is neither on the equispaced sampling diagonal nor on the simultaneous samples diagonal, as in Figure 5.6, where $\Delta_1 = \Delta_2$. Note that when there is no circuit noise present, due to the root-raised cosine filter, an oversampling converter that takes two samples per symbol period contains sufficient information to completely reconstruct the analog output of the channel (including the channel noise). In this limit, the mutual information becomes uniform as a function of the delays, for all pairs (Δ_1, Δ_2) such that $\Delta_1 \neq \Delta_2$.

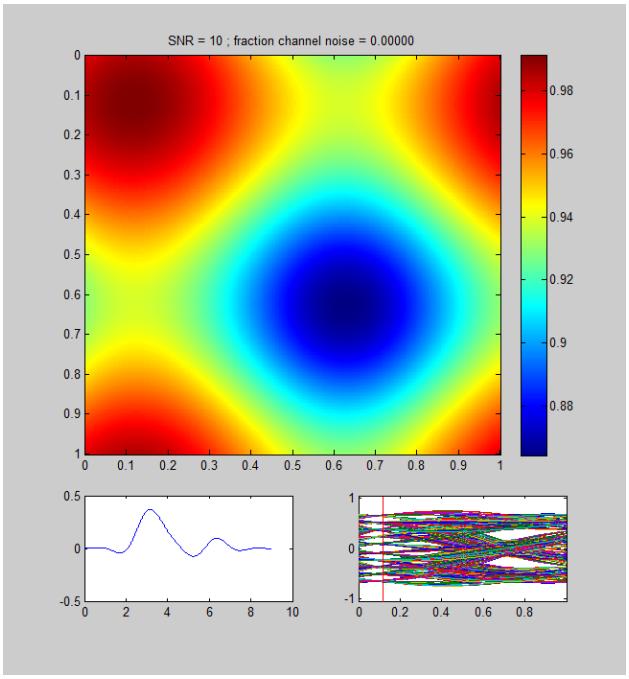


Figure 5.6: $C(\mathbf{D})$ for oversampling by a factor of 2 at $SNR = 10$ and 0% channel noise.

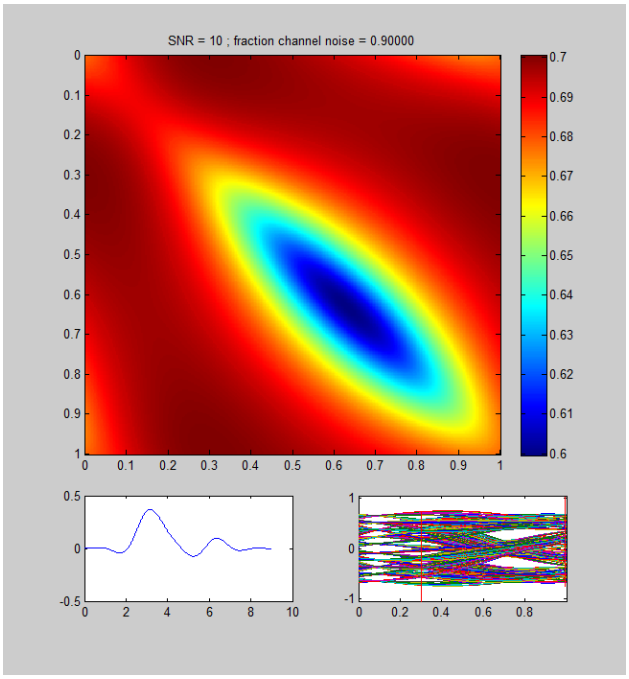


Figure 5.7: $C(\mathbf{D})$ for oversampling by a factor of 2 at $SNR = 10$ and 90% channel noise.

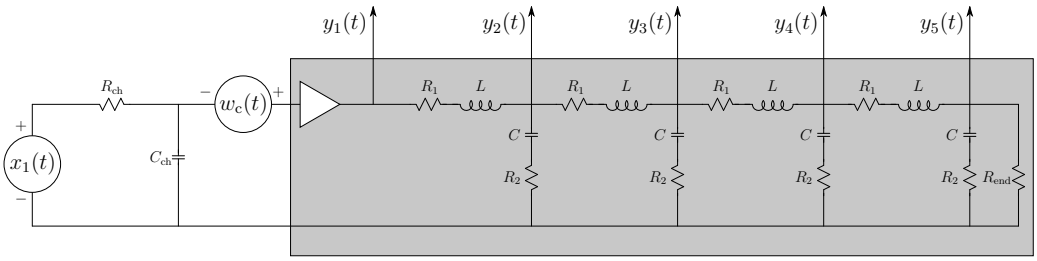


Figure 5.8: A transmitter and channel connected to a statistics gathering front-end, consisting of a delay line with five taps that may connect to samplers.

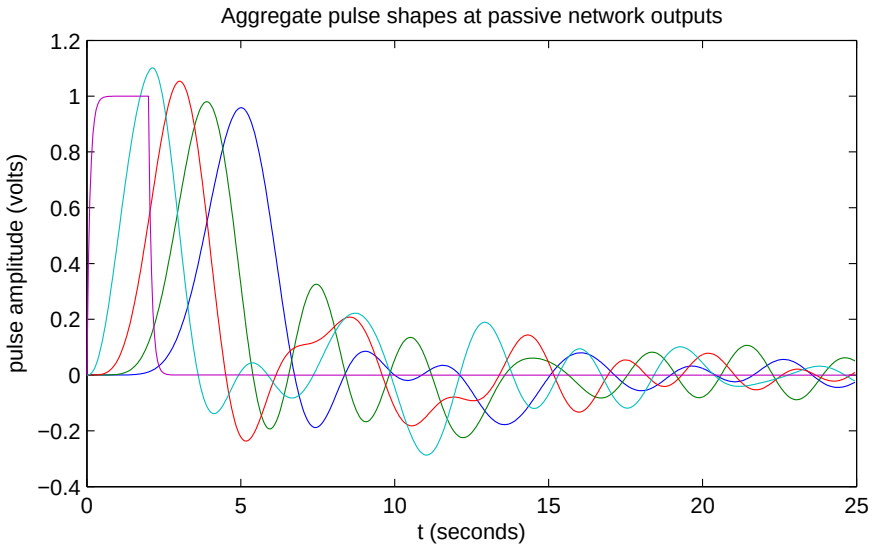


Figure 5.9: Aggregate pulse shapes $q_j(t)$ with $T = 2$s, as observed at outputs $y_1(t)$ (magenta, leftmost) to $y_5(t)$ (blue, rightmost).

5.7.2 Delay-Line Statistics Gathering Converter

Circuit Models

The particular communication system we will consider here utilizes a delay-line SGC front-end architecture, and is shown in Figure 5.8. Here, we have a transmitter that generates the voltage signal $x_1(t)$. This is connected to a channel consisting of an RC low pass filter, where we have chosen $R_{ch}C_{ch} = 0.1$s for our simulations. The noise signal $w_c(t)$ is added to the output of the channel to produce $x_2(t)$, which is the input to the front-end. The front-end has an input buffer to isolate the dynamics of the passive delay-line from that of the channel. This buffer feeds a chain of resistors, in-

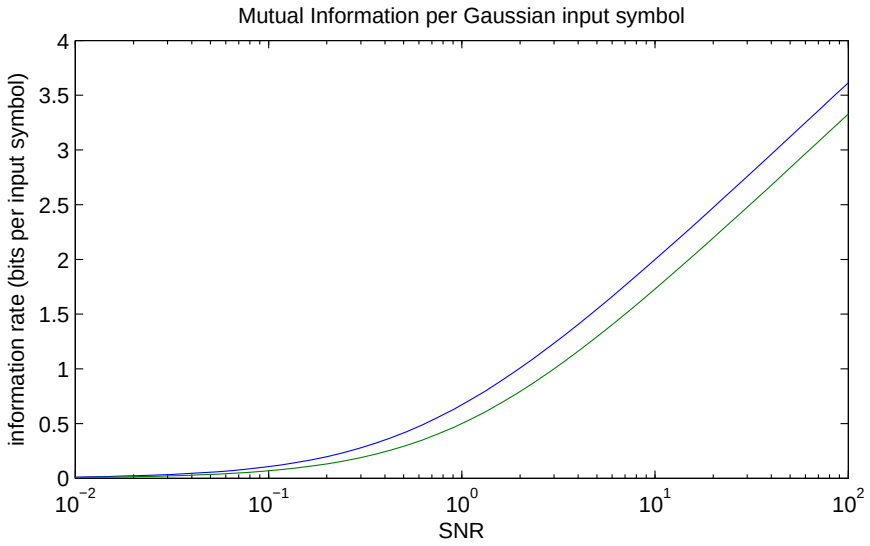


Figure 5.10: Mutual information versus SNR for a system taking outputs $y_1(t)$, $y_3(t)$, and $y_5(t)$ (upper curve, blue), compared with baudrate sampling classical ADC (lower curve, green). This uses $T = 2$s and $S = 6$s. The delay-line statistics gatherer outperforms the ADC.

ductors, and capacitors. We use $R_1 = R_2 = 0.01\Omega$, $L = 0.7$H, and $C = 1.0$F, chosen such that the delay per section is approximately 1s. An additional terminating resistor is at the end of the chain with $R_{\text{end}} = 1.0\Omega$. The observations used for the recovery of the transmitted data come from sampling a subset of the outputs from the delay-line, indicated by $y_j(t)$ in Figure 5.8, $j = 1, ..., 5$. Each of the chosen outputs is sampled simultaneously once every S seconds.

We apply the assumptions discussed in Section 5.3 about the input distribution, depending on which metric we are using to evaluate a communication architecture. We model the noise as having the following autocorrelation function:

$$R_{w_c}(t) = \sigma_{\text{ch}}^2 \exp\left(\frac{-|t|}{R_{\text{ch}} C_{\text{ch}}}\right),$$

which corresponds with white noise that has been filtered by the channel.

In our simulations, we compare statistics gathering front-ends to traditional ADC front-ends. ADC front-ends are simply modeled as sampling the signal $y_1(t)$ in Figure 5.8. (Note that the delay-line has no effect on the signal $y_1(t)$.) To make fair comparisons between the different front-ends, we fix the analog channel noise autocorrelation and the discrete time thermal noise

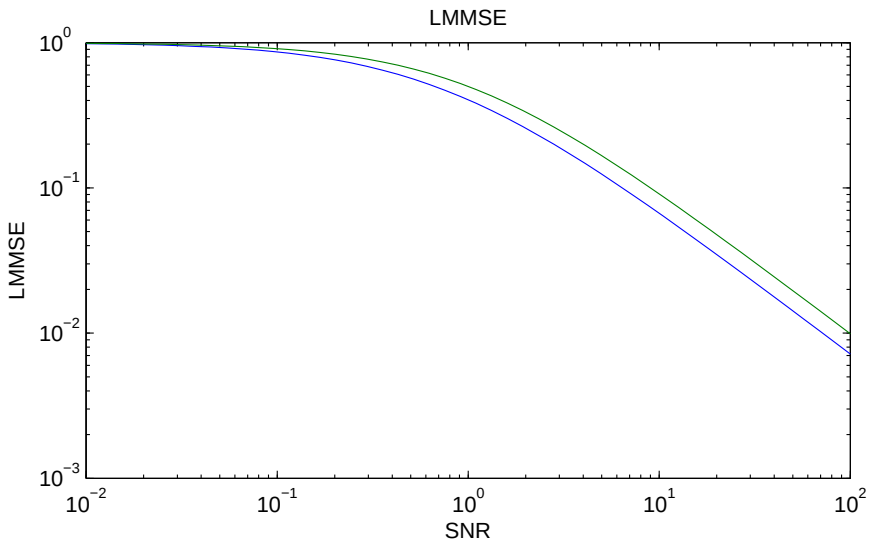


Figure 5.11: LMMSE versus SNR for a system taking outputs $y_1(t)$, $y_3(t)$, and $y_5(t)$ (lower curve, blue), compared with baudrate sampling classical ADC (upper curve, green). This uses $T = 2$s and $S = 6$s. The delay-line statistics gatherer outperforms the ADC.

variance, and we define the SNR as

$$\text{SNR} = \frac{E[x[i]^2]}{R_{w_c}(0) + \sigma_{\text{th}}^2} = \frac{1}{\sigma_{\text{ch}}^2 + \sigma_{\text{th}}^2}. \tag{5.29}$$

This is necessary because, in the discrete time model, the statistics of the component of noise coming from the channel depends on the model of the front-end, even for the same model for everything before the front-end. Comparisons should be made between different front-ends applied to the same channel model. For now, we also only draw comparisons where the number of samples generated per second is the same. For example, we may consider a situation with a symbol period of $T = 4$s and a twice baud ADC sampling once every $S = 2$s. This could be compared with a delay-line front-end that takes three simultaneous samples from signals $y_1(t)$, $y_3(t)$, and $y_5(t)$ once every $S = 6$s. Thus, the average sampling rate in both cases is one sample every two seconds.

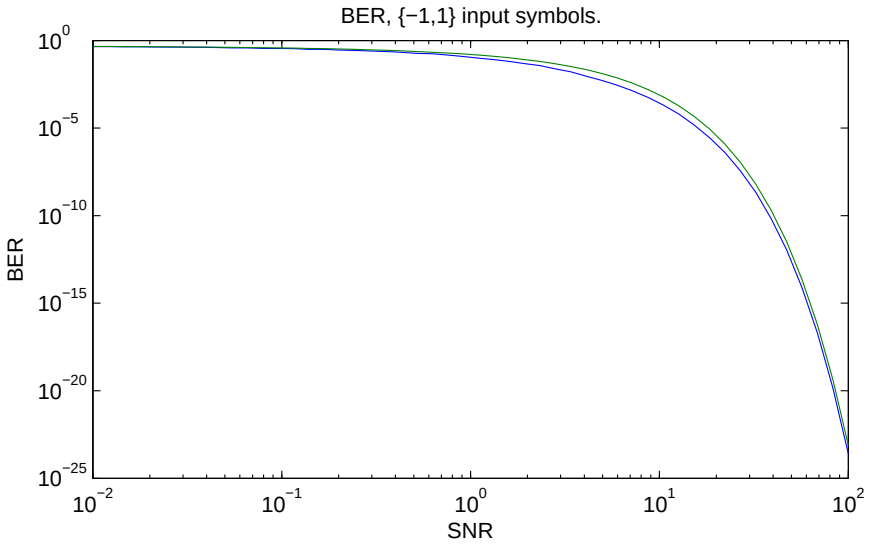


Figure 5.12: Bit error rate versus SNR for a system taking outputs $y_1(t)$, $y_3(t)$, and $y_5(t)$ (lower curve, blue), compared with baudrate sampling classical ADC (upper curve, green). This uses $T = 2$s and $S = 6$s. The delay-line statistics gatherer outperforms the ADC.

Simulation Results

As compared with converters designed for high reconstruction accuracy, converter architectures that allow the introduction of signal distortions can still preserve relevant statistics about the transmitted signal for the purpose of data recovery, as measured by the proper system level metrics, and can even improve the performance in this task, while simultaneously removing the need for precise fabrication and calibration. It is the goal of this section to demonstrate this by comparing a system using the described delay line SGC with a comparable design utilizing a classical ADC.

In this set of simulations, we let the symbol period be $T = 2$s. Symbols are modulated onto the channel as specified by Equation (5.1). Aggregate pulse shapes from each of the five delay line outputs are shown in Figure 5.9. For the SGC, we take only outputs $y_1(t)$, $y_3(t)$, and $y_5(t)$.

We first look at the performance of the SGC versus that of a classical ADC as a function of SNR. We have the converter sample these signals when $t = 6n + \delta$ for integer values of n and $\delta = 1$s. We compare this with an ADC that takes samples when $t = 2n + \delta$. We choose $\sigma^2_{\text{ch}} = \sigma^2_{\text{th}}$ and vary the SNR as defined in Equation (5.29) from 0.01 up to 100. What we observe is that the SGC outperforms the ADC front-end over the range of SNR values for

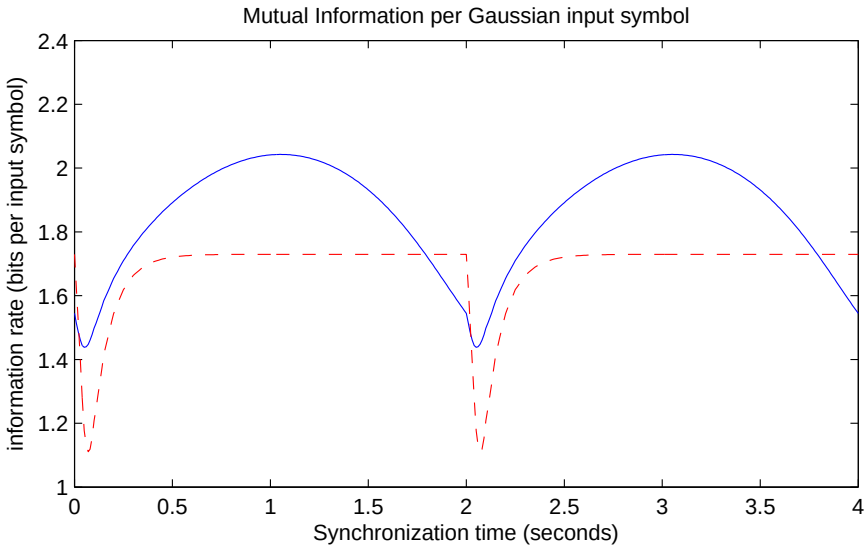


Figure 5.13: Mutual information versus synchronization point for a system taking outputs $y_1(t)$, $y_3(t)$, and $y_5(t)$ (solid curve, blue), compared with baudrate sampling classical ADC (dashed curve, red). This uses $T = 2$s and $S = 6$s. SNR $= 10$.

each of the three metrics. These results are shown in Figures 5.10, 5.11, and 5.12.

We next compare SGC versus ADC performance for a fixed SNR $= 10$, with $\sigma^2_{\text{ch}} = \sigma^2_{\text{th}} = 0.05$, and examine the performance of the front-ends as the sampling parameter δ is varied in the range $(0, T)$. These results are shown in Figures 5.13, 5.14, and 5.15. In these figures, we plot the system metric versus sampling parameter δ over two sample periods to highlight the fact that these metrics are periodic with respect to the sampling period. What we observe is that, in this particular setup, there is a wide range of δ for which the SGC outperforms the optimal value of δ for an ADC. This can also tell us something about what happens if there is timing uncertainty in the converters. For example, suppose that there is a jitter on the sampling times that is random and slowly time varying. In this case, the performance with respect to a metric of the front-end with jitter will be $E_\delta[\mathcal{M}(\mathbf{A}, \mathbf{v}|\delta)]$, where $\mathcal{M}(\mathbf{A}, \mathbf{v}|\delta)$ is any of the three discussed metrics, conditional on a particular value of the timing offset δ. Depending on the distribution on δ, this example SGC architecture can perform favorably to an ADC in all three metrics.

Finally, we briefly examine the performance of the SGC architecture ac-

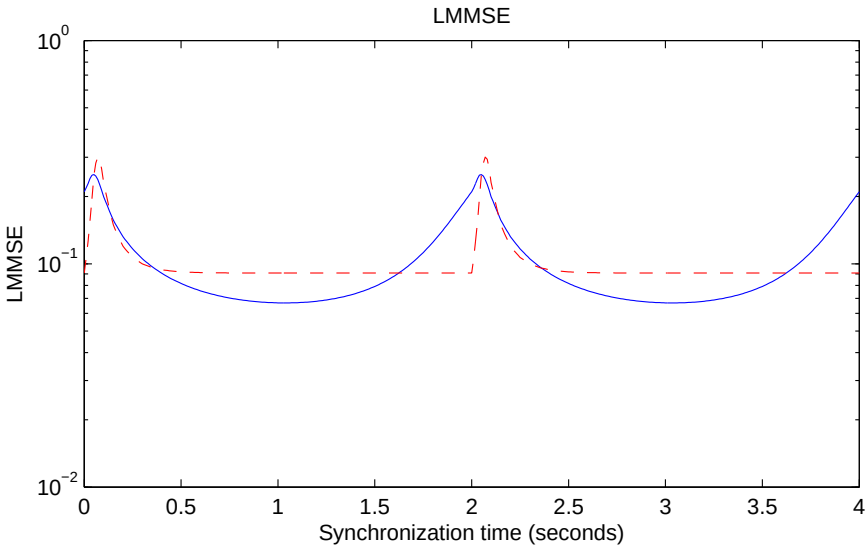


Figure 5.14: Linear minimum mean square estimation error versus synchronization point for a system taking outputs $y_1(t)$, $y_3(t)$, and $y_5(t)$ (solid curve, blue), compared with baudrate sampling classical ADC (dashed curve, red). This uses $T = 2$s and $S = 6$s. SNR $= 10$.

Fraction of Nyquist	SFDR (dB)	$-$THD (dB)
0.1	25.7324	23.3989
0.2	22.0213	20.1997
0.5	14.6129	12.6588
0.8	19.1103	16.8946
0.9	18.0881	15.1638

Table 5.1: SFDR and THD for an SGC using outputs $y_1(t)$, $y_3(t)$, and $y_5(t)$ with $T = 2$s and $S = 6$s.

cording to the waveform-centric metrics SFDR and THD. These results are given in Table 5.1. As compared with most ADCs referenced in [93], the SGC appears to have terrible performance. However, as indicated by the performance on system metrics, this SGC can even outperform an ideal ADC for communications. This should be expected since the waveform-centric metrics implicitly assume that the sequence given by the converter consists of equispaced (in time) samples. While the delay-line SGC somewhat approximates this, other SGCs may give samples with no such sequential-in-time interpretation while retaining the desired level of communication performance.

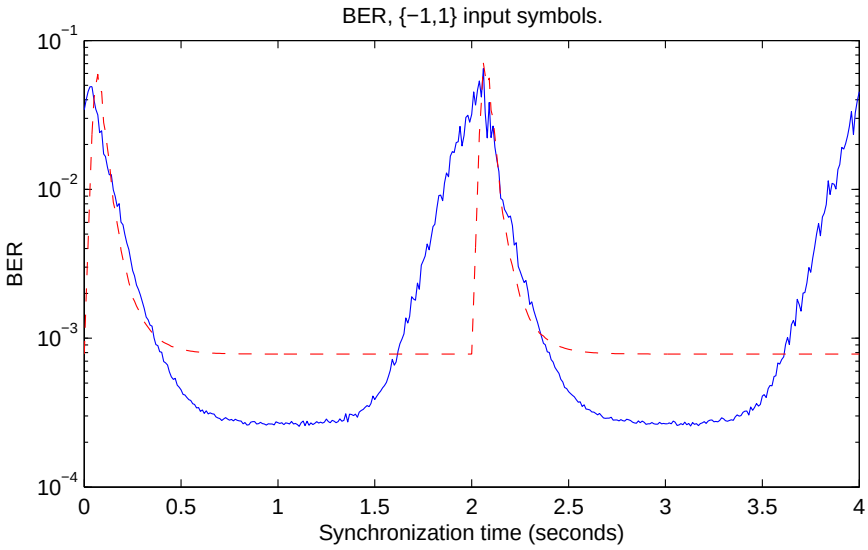


Figure 5.15: Bit error rate versus synchronization point for a system taking outputs $y_1(t)$, $y_3(t)$, and $y_5(t)$ (solid curve, blue), compared with baudrate sampling classical ADC (dashed curve, red). This uses $T = 2$s and $S = 6$s. SNR $= 10$.

5.8 Receiver Design with Statistics Gathering Converters

We first examine the relevance of the LMMSE and BER system level metrics for the design of practical systems. This is important because the metrics assume linear detectors of unbounded complexity, whereas applications require finite, and often low, complexity. We compare the performance indicated by the system metrics with that achieved by carefully designed low complexity equalizers. Specifically, in this set of analyses the SGC equalizer consists of three filters, each with 11 taps. Let $\{y_1[3i]\}$, $\{y_3[3i-1]\}$ and $\{y_5[3i-2]\}$ for integers i be the sequences of outputs from the respective delay line taps 1, 3, and 5. Then the full interleaved observation sequence is $\{..., y[0], y[1], ...\} = \{..., y_1[0], y_5[1], y_3[2], y_1[3], y_5[4], ...\}$. To estimate symbol $x[i]$, we use filter $h_{\text{mod}(i,3)}$ to get the estimate $\hat{x}[i] = h_{\text{mod}(i,3)}^{\top}\mathbf{y}[i]$, where $\mathbf{y}[i] = (y[i-5], ..., y[i+5])^{\top}$. In this work, we use a simple LMS update with a step size parameter μ to independently adapt each of the estimation filters. The need for the three separate estimation filters arises because the characteristics of the observation vector $\mathbf{y}[i]$ are similar to those of $\mathbf{y}[i+3j]$ for integer values j, but potentially quite different from $\mathbf{y}[i+3j+1]$ and

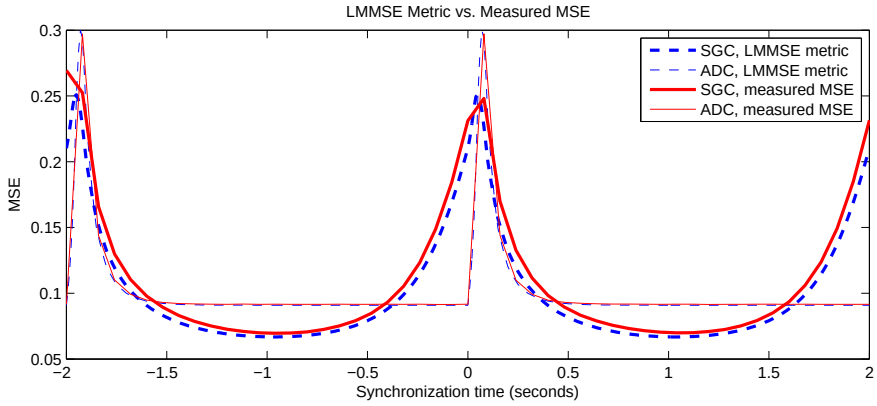


Figure 5.16: LMMSE Metric and measured MSE vs. sync time.

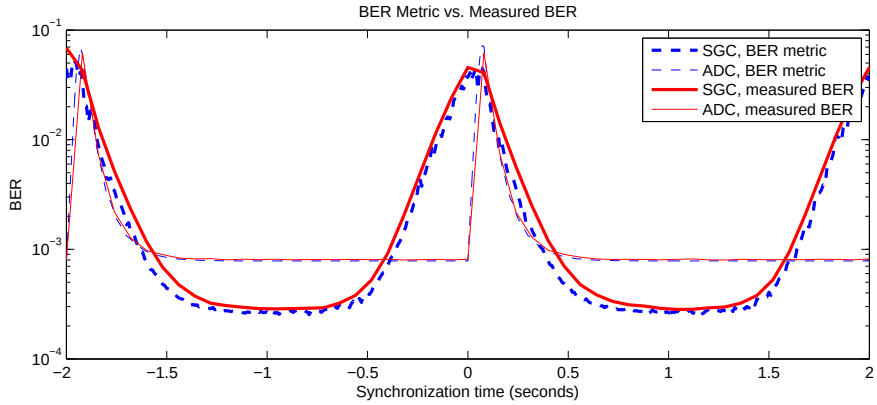


Figure 5.17: BER Metric and measured BER vs. sync time.

$\mathbf{y}[i+3j+2]$. This can be seen in the channel matrix $\mathbf{A}$. For a baud-sampled system using an ADC, $\mathbf{A}$ will have a Toeplitz structure, but in a system using our SGC (with the outputs arranged in a vector as $\{y[i]\}$ is here), it will instead have a *blockwise* Toeplitz structure. The width of the blocks composing $\mathbf{A}$ determines the number of separate estimation filters to maintain and adapt.

Figures 5.16 and 5.17 show how our system level metrics compare with the actual performance of a communication system utilizing either an SGC or ADC with postprocessing of the observations by appropriate low complexity equalizers. On the horizontal axis, we have the synchronization time of the samplers (both ADC and SGC), with the center of the eye being at odd integer times $2i+1$. The thick lines in both figures are for the SGC, thin for the ADC. The dashed lines are for the value of the system level metric achieved for the particular synchronization time, whereas the solid lines are

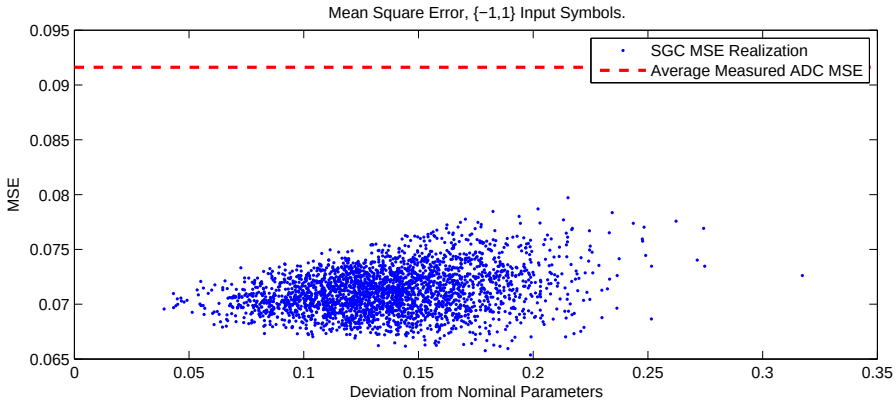


Figure 5.18: MSE of realizations of SGCs under process variation. Standard Deviation = 0.05.

for the measured MSE and BER of a communication system utilizing the respective data converter with the appropriate equalization method. For measuring the MSE and BER of the SGC systems, we trained the length 11 specialized LMS equalizer with parameter $\mu = 0.01$ on 30000 symbols and measured the MSE and BER on 2e7 subsequent symbols. For ADC systems, we do the same, but with a standard LMS equalizer. The figures show that the measured performance achieved by systems with constrained complexity is quite similar to the theoretical performance indicated by the system level metrics. In particular, the system level metrics correctly indicate that a system utilizing an SGC has the potential to outperform one with an ADC. Furthermore, the system level metrics and the corresponding measured performance have the same qualitative characteristics.

5.9 Robustness of System Metric Performance under Parameter Variation

In this set of simulations, we show that the communications performance of the SGC front-end, when paired with the described low complexity adaptive equalizer, is still able to beat the performance of the ideal ADC under significant levels of deviation from the nominal circuit parameters. In particular, recall that we have four each of capacitors and inductors in the SGC with nominal values $L = 0.7$H and $C = 1.0$F. We model process variation in the circuit by letting the true circuit parameter be Gaussian distributed

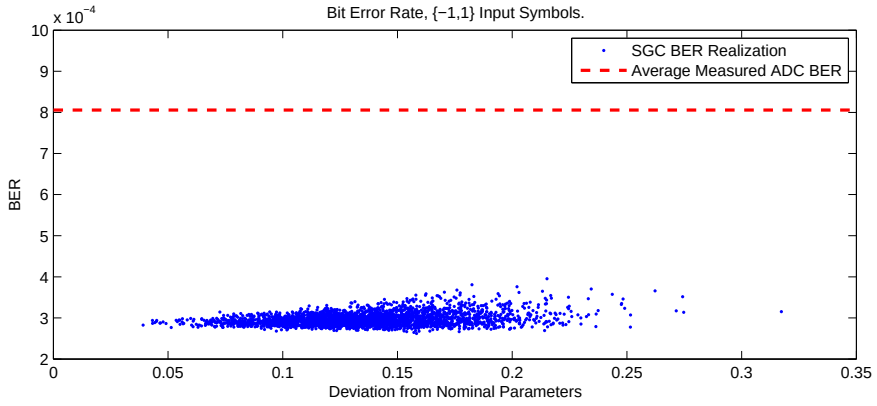


Figure 5.19: BER of realizations of SGCs under process variation. Standard Deviation = 0.05.

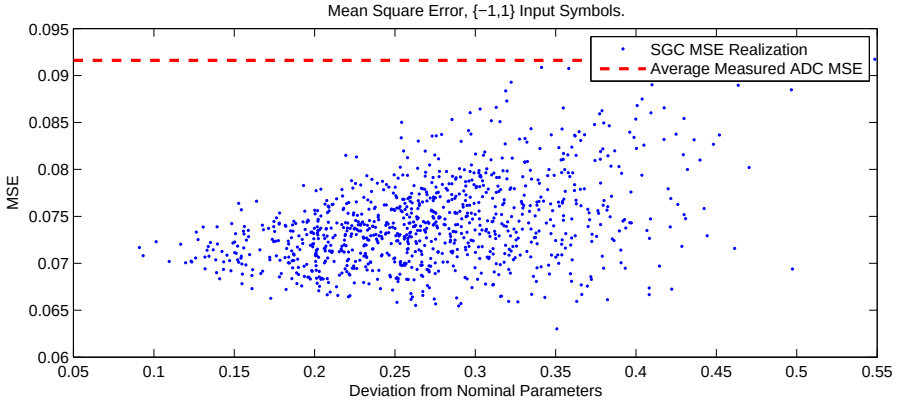


Figure 5.20: MSE of realizations of SGCs under process variation. Standard Deviation = 0.10.

with the nominal value as mean, and standard deviation of 0.05 or 0.10 (H or F). Figures 5.18 and 5.19 show the results of 2800 random instances of the SGC system for standard deviation of 0.05, where we train the length 11 specialized LMS equalizer with parameter $\mu = 0.01$ on 30000 symbols and measure the MSE and BER on 2e7 subsequent symbols. Each point in the plots represents the measured MSE / BER versus the norm of the deviation from the nominal parameters, i.e. $|P - \bar{P}|_2$, where P is the vector of 8 circuit parameters and $\bar{P}$ is the vector of 8 nominal values. The dashed lines indicate the performance level of the system when using a classical ADC. It is clear that the performance is robust to the parameter variation, and remains better than the ADC in all cases. Similar results hold when the standard deviation of the parameters is doubled to 0.1, as shown in Figures 5.20 and 5.21, with a small number of instances (5 out of 1000 points for both MSE

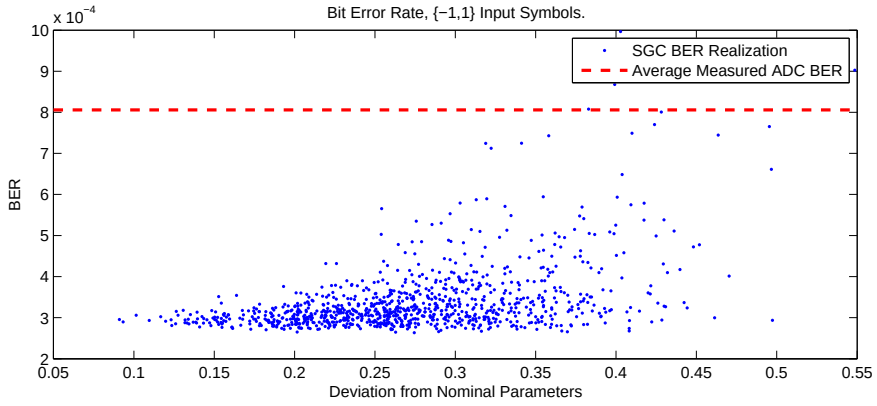


Figure 5.21: BER of realizations of SGCs under process variation. Standard Deviation = 0.10.

and BER) with high deviation from the nominal parameters having worse performance than the ADC.

5.10 Conclusion

In this chapter, we have taken a holistic view on the design of communications systems, and we have brought into question the usual methods of designing and evaluating the performance of data converters. In this work, we propose that the typical performance metrics used to design ADCs, such as THD and SFDR, should be thrown out. In their place, we propose considering metrics that are directly aligned with the system level goal of reliable information transfer. We also propose shifting to the idea of creating Statistics Gathering Converters (SGCs) instead of Analog to Digital Converters (ADCs), where we may consider certain design choices that would not be considered viable possibilities from the typical ADC perspective. We have given a general mathematical framework for these SGCs, as well as a number of metrics to be used for validating their designs: Mutual Information Rate (MI), Linear Minimum Mean Squared Error (LMMSE), and Bit Error Rate (BER) for unquantized systems, and MI and BER for quantized systems. We have furthermore given computational methods for evaluating these metrics on proposed designs. Finally, we have presented evidence from simulations that shows the potential for SGCs to outperform the classical ADCs in a number of scenarios, and discussed the design choices for receiver algorithms that are

necessary in order to fully realize these potential gains.

There is certainly more work that could be done. For instance, there are many conceivable SGC architectures that could be envisioned that are very different from classical ADCs. It would be interesting to examine a number of architectures, optimize them with respect to the system metrics, and compare them with the communication performance or energy consumption of a standard ADC receiver design. Work in this direction has already been done by optimizing the quantization levels of an ADC with respect to the BER metric [74, 90], but there are many other parameters that could be tuned in these converter designs.

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.

[34] A. C. Singer and M. Feder, "Universal linear prediction by model order weighting," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2685–2699, October 1999.

[35] S. S. Kozat and A. C. Singer, "Switching strategies for sequential decision problems with multiplicative loss with application to portfolios," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2192–2208, June 2009.

[36] S. S. Kozat, A. C. Singer, and G. C. Zeitler, "Universal piecewise linear prediction via context trees," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3730–3745, July 2007.

[37] S. S. Kozat, A. C. Singer, and A. J. Bean, "Universal portfolios via context trees," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 2093 – 2096.

[38] S. S. Kozat and A. C. Singer, "Universal switching portfolios under transaction costs," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 5404 – 5407.

[39] Y. Singer, "Switching portfolios," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998, pp. 1498–1519.

[40] A. Blum and A. Kalai, "Universal portfolios with and without transaction costs," *Machine Learning*, vol. 35, no. 3, pp. 193–205, June 1999.

[41] A. Agarwal, E. Hazan, S. Kale, and R. E. Schapire, "Algorithms for portfolio management based on the Newton method," in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 9–16.

[42] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, March 1952.

[43] T. M. Cover, "Universal portfolios," *Mathematical Finance*, vol. 1, no. 1, pp. 1–29, January 1991.

[44] D. P. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth, "Online portfolio selection using multiplicative updates," *Mathematical Finance*, vol. 8, no. 4, pp. 325–347, October 1998.

[45] M. Feder, N. Merhav, and M. Gutman, "Universal prediction of individual sequences," *IEEE Transactions on Information Theory*, vol. 38, no. 4, pp. 1258–1270, July 1992.

[46] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and Computation*, vol. 108, no. 2, pp. 212–261, February 1994.

[47] T. M. Cover and E. Ordentlich, "Universal portfolios with side information," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 348–363, March 1996.

[48] R. Bell and T. M. Cover, "Game theoretic optimal portfolios," *Management Science*, vol. 34, no. 6, pp. 724–733, June 1988.

[49] D. P. Foster and R. Vohra, "Regret in the on-line decision problem," *Games and Economic Behavior*, vol. 29, no. 1-2, pp. 7–35, October 1999.

[50] A. J. Bean and A. C. Singer, "Factor graphs for universal portfolios," in *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*, 2009, pp. 1375 – 1379.

[51] A. J. Bean and A. C. Singer, "Universal switching and side information portfolios under transaction costs using factor graphs," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010, pp. 1986 – 1989.

[52] A. J. Bean and A. C. Singer, "Factor graph switching portfolios under transaction costs," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2011, pp. 5748 – 5751.

[53] A. J. Bean and A. C. Singer, "Universal switching and side information portfolios under transaction costs using factor graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 4, pp. 351–365, August 2012.

[54] T. M. Cover and E. Ordentlich, "Universal portfolios with short sales and margin," in *Proceedings of the 1998 IEEE International Symposium on Information Theory*, August 1998, p. 174.

[55] S. S. Kozat and A. C. Singer, "Universal constant rebalanced portfolios with switching," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2007, pp. 1129 – 1132.

[56] G. Stoltz and G. Lugosi, "Internal regret in online portfolio selection," *Machine Learning*, vol. 59, no. 1-2, pp. 125–159, May 2005.

[57] G. Iyengar, "Universal investment in markets with transaction costs," *Mathematcal Finance*, vol. 15, no. 2, pp. 359–371, April 2005.

[58] R. E. Krichevsky and V. K. Trofimov, "The performance of universal encoding," *IEEE Transactions on Information Theory*, vol. 27, no. 2, pp. 199–207, March 1981.

[59] A. Kalai and S. Vempala, "Efficient algorithms for universal portfolios," *Journal of Machine Learning Research*, vol. 3, pp. 423–440, November 2002.

[60] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, June 2006.

[61] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, vol. 5, December 2003, pp. 4997–5002.

[62] M. H. Degroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, March 1974.

[63] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, November 1984.

[64] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. AC-31, no. 9, pp. 803–812, September 1986.

[65] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3166, July 2008.

[66] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[67] S. Ram, V. V. Veeravalli, and A. Nedić, "Distributed and recursive parameter estimation in parametrized linear state-space models," *IEEE Transactions on Automatic Control*, vol. 55, no. 2, pp. 488–492, February 2010.

[68] S. Kirti and A. Scaglione, "Scalable distributed Kalman filtering through consensus," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 2008, pp. 2725–2728.

[69] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ: Wiley-Interscience, 2003.

[70] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[71] A. Bean and A. Singer, "Cooperative estimation in heterogeneous populations," in *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, November 2011, pp. 696–699.

[72] A. Bean, P. Kairouz, and A. Singer, "Convergence rates for cooperation in heterogeneous populations," in *2012 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, November 2012, pp. 531–534.

[73] L. Xiao, S. Boyd, and S. Lall, "Distributed average consensus with time-varying Metropolis weights," June 2006, unpublished.

[74] M. Lu, N. Shanbhag, and A. Singer, "BER-optimal analog-to-digital converters for communication links," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2010, pp. 1029–1032.

[75] A. Nazemi et al., "A 10.3GS/s 6bit (5.1 ENOB at nyquist) time-interleaved/pipelined ADC using open-loop amplifiers and digital calibration in 90nm CMOS," in *2008 Symposium on VLSI Circuits, Technical Digest of Papers*, 2008, pp. 18–19.

[76] H. M. Bae, J. Ashbrook, J. Park, N. Shanbhag, A. Singer, and S. Chopra, "An MLSE receiver for electronic-dispersion compensation of OC-192 links," *Journal of Solid-State Circuits*, vol. 41, no. 11, pp. 2541–2554, November 2006.

[77] P. Nikaeen and B. Murmann, "Digital compensation of dynamic acquisition errors at the front-end of high-performance A/D converters," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 3, pp. 499–508, 2009.

[78] C. Grace, P. J. Hurst, and S. H. Lewis, "A 12b 80MS/s pipelined ADC with bootstrapped digital calibration," in *IEEE International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers*, February 2004, pp. 460–461.

[79] J. Singh, O. Dabeer, and U. Madhow, "Communication limits with low precision analog-to-digital conversion at the receiver," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2007.

[80] J. Singh, O. Dabeer, and U. Madhow, "Capacity of the discrete-time AWGN channel under output quantization," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2008.

[81] O. Dabeer, J. Singh, and U. Madhow, "On the limits of communication performance with one-bit analog-to-digital conversion," in *Proceedings of the IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2006.

[82] G. Zeitler, A. Singer, and G. Kramer, "Low-precision A/D conversion for maximum information rate in channels with memory," *IEEE Transactions on Communications*, vol. 60, no. 9, pp. 2511–2521, September 2012.

[83] A. Singer, A. J. Bean, and J. W. Choi, "Mutual information and time-interleaved analog-to-digital conversion," in *Information Theory and Applications Workshop*, 2010, pp. 1–5.

[84] A. J. Bean and A. Singer, "A deflection criterion for time-interleaved analog-to-digital converters," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, May 2011, pp. 265–268.

[85] A. J. Bean and A. C. Singer, "The SGC: A simple architecture for gathering statistics in communication links," in *IEEE Workshop on Signal Processing Systems*, 2013.

[86] A. J. Bean and A. Singer, "Statistics gathering converters: System level metrics, simulated performance, and process variation robustness," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2014, pp. 8365–8369.

[87] R. Walden, "Analog-to-digital converter survey and analysis," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 539–550, April 1999.

[88] H. Widom, "Asymptotic behavior of block Toeplitz matrices and determinants," *Advances in Mathematics*, vol. 13, no. 3, pp. 284–322, July 1974.

[89] H. Widom, "Asymptotic behavior of block Toeplitz matrices and determinants. II," *Advances in Mathematics*, vol. 21, no. 1, pp. 1–29, July 1976.

[90] R. Narasimha, M. Lu, N. Shanbhag, and A. Singer, "BER-optimal analog-to-digital converters for communication links," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3683–3691, July 2012.

[91] D. M. Arnold, H.-A. Loeliger, P. O. Vontobel, A. Kavčić, and W. Zeng, "Simulation-based computation of information rates for channels with memory," *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3498–3508, August 2006.

[92] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, March 1974.

[93] B. Murmann, "ADC performance survey 1997-2013," [Online]. Available: http://www.stanford.edu/ murmann/adcsurvey.html.