UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

ADAPTIVE DISTRIBUTED SOURCE CODING BASED ON BAYESIAN INFERENCE

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

FENG CHEN
Norman, Oklahoma
2015

ADAPTIVE DISTRIBUTED SOURCE CODING BASED ON BAYESIAN INFERENCE

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Dr. Samuel Cheng, Chair

Dr. Pramode Verma

Dr. Nikola Petrov

Dr. Choon Yik Tang

Dr. Gregory MacDonald

Acknowledgments

I would like to take this opportunity to express my deepest gratitude to my adviser and mentor Dr. Samuel Cheng for his hard work and his support during my academic career. During the last three years, Dr. Cheng offered me an amazing research environment, and in this environment I progressed from the status of an amateur researcher to the status of a real researcher. Without his guidance and support, I would not be able to complete this dissertation. I would also like to express my gratitude to Dr. Verma, Dr. Petrov, Dr. Tang, and Dr. MacDonald for serving as my committee members and for guiding me during my dissertation work.

I would like to thank all of the members of Dr. Cheng's group and my other classmates at OU-Tulsa TCOM: Shuang Wang, Lijuan Cui, Amin, Nafise, Lei Yang, Dong Han, etc.

I also want to thank all of my former mathematics and physics teachers. These teachers prompted and encouraged my love for math and science from the time that I was 7 years old.

I would like to offer my sincere gratitude to my family members. My parents raised me and gave me my preliminary education. I would also like to thank my wife, Huiling Wu, who encouraged me to complete this dissertation, and who gave birth to my dear son Michael.

Table of Contents

Chapter **Page**

List of Tables

Table **Page**

List of Figures

Figure **Page**

Abstract

Distributed Source Coding (DSC) is an important topic for both in information theory and communication. DSC utilizes the correlations among the sources to compress data, and it has the advantages of being simple and easy to carry out. In DSC, Slepian-Wolf (S-W) and Wyner-Ziv (W-Z) are two important problems, which can be classified as lossless compression and loss compression, respectively. Although the lower bounds of the S-W and W-Z problems have been known to researchers for many decades, the code design to achieve the lower bounds is still an open problem.

This dissertation focuses on three DSC problems: the adaptive Slepian-Wolf decoding for two binary sources (ASWDTBS) problem, the compression of correlated temperature data of sensor network (CCTDSN) problem and the streamlined genome sequence compression using distributed source coding (SGSCUDSC) problem. For the CCTDSN and SGSCUDSC problems, sources will be converted into the binary expression as the sources in ASWDTBS problem for encoding. The Bayesian inference will be applied to all of these three problems. To efficiently solve these Bayesian inferences, message passing algorithm will be applied. For a discrete variable that takes a small number of values, the belief propagation (BP) algorithm is able to implement the message passing algorithm efficiently. However, the complexity of the BP algorithm increases exponentially with the number of values of the variable. Therefore, the BP algorithm can only deal with discrete variable that takes a small number of values and limited continuous variables. For the more complex variables, deterministic approximation methods are used. These methods, such as the variational Bayes (VB) method and expectation propagation (EP) method, can efficiently incorporated into the message

passing algorithm.

A virtual binary asymmetric channel (BAC) channel was introduced to model the correlation between the source data and the side information (SI) in ASWDTBS problem, in which two parameters are required to be learned. The two parameters correspond to the crossover probabilities that are $0 \rightarrow 1$ and $1 \rightarrow 0$. Based on this model, a factor graph was established that includes LDPC code, source data, SI and both of the crossover probabilities. Since the crossover probabilities are continuous variables, the deterministic approximate inference methods will be incorporated into the message passing algorithm. The proposed algorithm was applied to the synthetic data, and the results showed that the VB-based algorithm achieved much better performance than the performances of the EP-based algorithm and the standard BP algorithm. The poor performance of the EP-based algorithm was also analyzed.

For the CCTDSN problem, the temperature data were collected by crossbow sensors. Four sensors were established in different locations of the laboratory and their readings were sent to the common destination. The data from one sensor were used as the SI, and the data from the other 3 sensors were compressed. The decoding algorithm considers both spatial and temporal correlations, which are in the form of Kalman filter in the factor graph. To deal with the mixtures of the discrete messages and the continuous messages (Gaussians) in the Kalman filter region of the factor graph, the EP algorithm was implemented so that all of the messages were approximated by the Gaussian distribution. The testing results on the wireless network have indicated that the proposed algorithm outperforms the prior algorithm.

The SGSCUDSC consists of developing a streamlined genome sequence

compression algorithm to support alternative miniaturized sequencing devices, which have limited communication, storage, and computation power. Existing techniques that require a heavy-client (encoder side) cannot be applied. To tackle this challenge, the DSC theory was carefully examined, and a customized reference-based genome compression protocol was developed to meet the low-complexity need at the client side. Based on the variation between the source and the SI, this protocol will adaptively select either syndrome coding or hash coding to compress variable lengths of code subsequences. The experimental results of the proposed method showed promising performance when compared with the state of the art algorithm (GRS).

CHAPTER 1

INTRODUCTION

In a wireless sensor network, all of the sensors may send their own data to a common destination. However, due to the limited capacity of the communication channel and the limited memory, the data must first be compressed so that it can be sent to the common destination. One major difference between wireless sensor networks and other mobile networks is that the observations among sensors are highly correlated. This means that there is a significant amount of redundancy among the observations. Distributed Source Coding (DSC) theory is concerned with the separate compressions of these correlated sources at each sensor and the joint decompression at the destination.

In 1973, Slepian and Wolf published their classical DSC paper [1]. In this paper, Slepian and Wolf proved a very surprising result, that is, generally speaking, it is possible to have no performance loss in separate encoding compared to the case in which joint encoding is allowed. Wyner and Ziv later considered a lossy version of asymmetric DSC in [2, 3]. Because of the contributions of Slepian-Wolf and Wyner-Ziv, we usually refer to lossless DSC as an S-W problem and lossy source coding with side information as a W-Z problem.

Since most of the problems in this dissertation can be classified as S-W coding or W-Z problems, these two codings will be briefly introduced first. Then, the contributions of this dissertation will be summarized. Finally, the organization of this dissertation will be outlined.

Before continuing, some notations should be classified:

- the uppercase letters $X, Y, \ldots$ denote the data source or random variables;
- the lowercase letters $x, y, \ldots$ denote one sample from $X, Y, \ldots$;
- $\mathcal{X}, \mathcal{Y}, \ldots$ denote the alphabets;
- x^n denotes one n-length sequence $\{x_1, \ldots, x_n\}$;
- $\hat{x}$ denotes the decoded word (estimation) of x.

1.1 S-W Coding

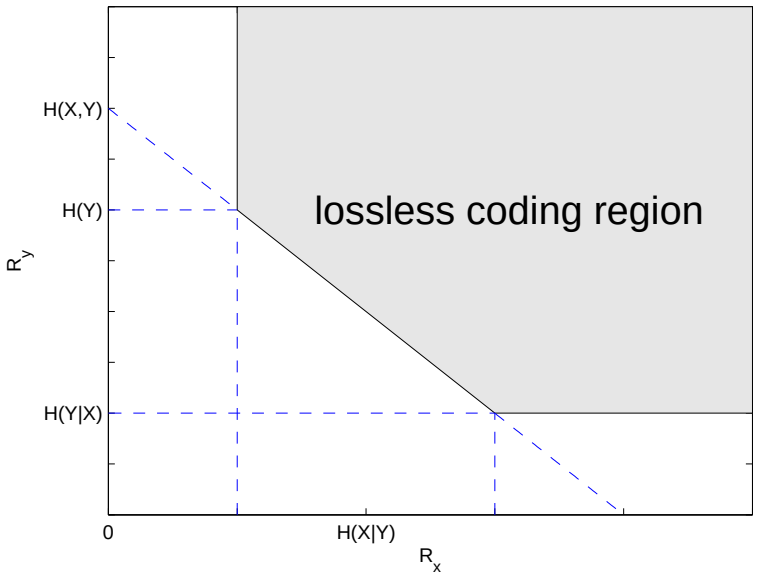


Figure 1.1: Lossless Coding Region

Definition 1.1 *Given two correlated joint sources* (X, Y)*, where* $X \in \mathcal{X}$ *and* $Y \in \mathcal{Y}$*, we will encode the n-length i.i.d sequences of* (X, Y) *with rate*

(R_X, R_Y) *with the maps*

$$\begin{aligned} f_1 &: \mathcal{X}^n \to \{1, \cdots, 2^{nR_x}\} \\ f_2 &: \mathcal{Y}^n \to \{1, \cdots, 2^{nR_y}\} \end{aligned} \tag{1.1}$$

and decode the codeword with the map

$$g : \{1, \cdots, 2^{nR_x}\} \times \{1, \cdots, 2^{nR_y}\} \to \mathcal{X}^n \times \mathcal{Y}^n. \tag{1.2}$$

If the decoding errors of these sequences are arbitrarily small, i.e.,

$$\begin{aligned} \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} D(x_i, \hat{x}_i) = 0, \\ \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} D(y_i, \hat{y}_i) = 0, \end{aligned} \tag{1.3}$$

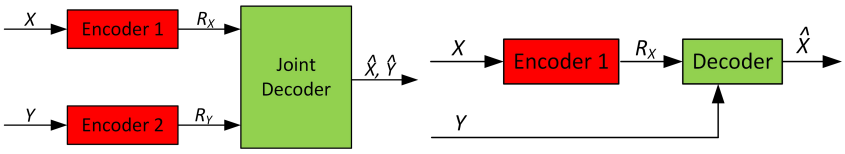
where $D(\cdot, \cdot)$ is the distortion function, then we designate these lossless codings as S-W codings.

It turns out that the achievable rate region (see Figure 1.1) of the S-W coding is given by [4]

$$\begin{aligned} R_X &\geq H(X|Y) \\ R_Y &\geq H(Y|X) \\ R_X + R_Y &\geq H(X, Y), \end{aligned} \tag{1.4}$$

where $H(\cdot)$ is the entropy operation.

Depending on whether the source Y will be compressed or not, the S-W coding can be classified as two schemes: non-asymmetric S-W coding or asymmetric S-W coding(see Figure 1.2). For asymmetric S-W coding, the



(a) (b)

Figure 1.2: Two types of S-W coding: (a) Non-asymmetric S-W coding; (b) asymmetric S-W coding.

source Y is uncompressed, which indicates $R_Y = H(Y)$. Hence, to satisfy (1.4), R_X should be $R_X \geq H(X,Y) - R_Y = H(X,Y) - H(Y) = H(X|Y)$.

Remark 1.1 *The proof of the S-W Theorem [1] is non-constructive, in that Slepian and Wolf did not indicate how to implement DSC efficiently. The research on practical algorithms for DSC was stagnant until the publication of the work by Pradhan and Ramchandran in 1999 [5]. They rediscovered an early work by Wyner in which he suggested the use of channel codes for asymmetric S-W coding [6].*

Remark 1.2 *With few exceptions, such as [7], the majority of S-W code designs are based on this channel coding formulation. Practical syndrome-based schemes for S-W coding using channel codes have been studied in [5, 8–26].*

1.2 W-Z Coding

The lossy decoding version of Figure 1.2(b) is called the "W-Z coding" problem. For a given distortion d, the W-Z coding problem tries to find the coding method that achieves the distortion requirement with a minimum rate, i.e.,

for an n block encoding scheme

$$\begin{aligned} &\{f^*, g^*\} = \arg\min_{f,g}\{R(f,g)\}, \\ &s.t. \\ &f : \mathcal{X}^n \to \{1, \ldots, 2^{nR}\} \qquad , \\ &g : \mathcal{Y}^n \times \{1, \ldots, 2^{nR} - 1\} \to \mathcal{X}^n \\ &\limsup_{n\to\infty} \mathbb{E}[D(x^n, g(f(x^n)))] \leq d \end{aligned} \tag{1.5}$$

where f and g are the encoding and decoding mappings, respectively, $\mathbb{E}[\cdot\,]$ denotes the expectation operation, and $D(\cdot\,,\cdot\,)$ denotes the distortion function. The minimum rate R is usually written as $R_Y^*(d)$

In [2,3], the authors obtained that

$$R_Y^*(d) = \inf_{p\in\mathcal{M}(d)}[I(X;Z) - I(Y;Z)], \tag{1.6}$$

$Z \in \mathcal{Z}$, where $\mathcal{Z}$ is arbitrary finite set, and $\mathcal{M}(d)$ is the set of $p(x,y,z)$ which satisfies

$$\begin{aligned} &\sum_{z\in\mathcal{Z}} p(x,y,z) = p(x,y), \\ &Y, Z \text{ are conditionally independent given } X. \end{aligned} \tag{1.7}$$

The W-Z problem is usually implemented by quantizing the source data followed by the S-W coding.

1.3 Contributions

This dissertation will focus on how to apply the Bayesian inference framework to solve parts of the S-W and W-Z problems. The Bayesian model

can be conveniently expressed by a factor graph, in which the message passing algorithm can be implemented to solve the model. Generally, Belief Propagation (BP) is an efficient message passing algorithm. However, for complex situations, such as those involving continuous random variables, it is generally infeasible to apply the BP algorithm. To solve this problem, the deterministic approximate methods will be incorporated into the message passing algorithm so that the factor graph can be efficiently decoded.

The contributions of this dissertation can be summarized as follows:

- Research the properties of different deterministic approximate algorithms, such as VB and EP for DSC decoding.

- Proposed an adaptive joint S-W decoding algorithm for binary sources. The correlation of the binary sources can be viewed as a virtual binary asymmetrical channel (BAC). Many of prior works can be view as special cases of this model.

- Build a temperature network and propose a new factor graph for this network. The proposed algorithm achieves a much better performance on the network data.

- First apply the DSC to the genome data compression and decompression. The proposed algorithms have been applied to two genome sequence datasets - the Arabidopsis Information Resource (TAIR) [27] and the Institute for Genomic Research (TIGR) [28]. The proposed algorithm achieves a better performance than that of the GRS algorithm.

Among all of above contributions, the DSC for temperature network and the DSC for genome data have already been published in [29, 30].

1.4 Organization

This dissertation is organized as follows: The basic concepts of Bayesian inference, a factor graph, message passing algorithms and deterministic approximation inference will be reviewed in Chapter 2. The BAC model will be studied in Chapter 3. Chapter 4 is about DSC for wireless temperature networks. In Chapter 5, the DSC for the genome data algorithm will be proposed and tested on two datasets. Chapter 6 provides a conclusion for this dissertation.

CHAPTER 2

BAYESIAN INFERENCE

A Bayesian technique considers all possible modes when inferring an outcome. In this chapter, the general idea of Bayesian inference will be explained. Then, the prior selection criteria for the Bayesian inference will be introduced. Finally, the factor graph and the inference methods in the factor graph will be introduced.

2.1 Introduction

First, let's classify some notations.

- $p(\cdot)$ is the probability distribution function.
- The upper case letters X, Y, and Z denote random variables or the variable nodes of the factor graph, and the lower case letters x, y, and z denote their corresponding values.
- θ is the parameter of the distribution model.
- $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ denotes the set of all observation data.

Suppose one wants to estimate the parameter of the model $P(x|\theta)$. From the point of view of classical probability, there exists a ground truth value of θ. Given an observation set $\mathbf{x}$, one common approach to estimate θ is to maximize the likelihood (MLE) function $p(\mathbf{x}|\theta)$, i.e.

$$\hat{\theta}_{MLE} = \arg\max_{\theta} p(\mathbf{x}|\theta). \tag{2.1}$$

Differing from classical probability, the parameter θ, under the Bayesian framework, is treated as a random variable. According to Bayes' theorem, the posterior distribution of θ can be written as

$$p(\theta|\mathbf{x}) = \frac{p(\theta)p(\mathbf{x}|\theta)}{p(\mathbf{x})}, \tag{2.2}$$

where $p(\theta)$ is the prior distribution, and $p(\mathbf{x})$ is the constant and called as evidence, which can be calculated as

$$p(\mathbf{x}) = \int p(\theta)p(\mathbf{x}|\theta)d\theta. \tag{2.3}$$

Therefore, one has the expression

$$\text{posterior} \propto \text{prior} \times \text{likelihood}. \tag{2.4}$$

The prior reflects the existing knowledge of θ before the data $\mathbf{x}$ are sampled. As a matter of fact, the form of (2.2) is suitable for sequentially updating: When the new data are obtained, the previous posterior can be treated as the prior and then multiplied with the likelihood of the new data as (2.2) to update the posterior.

By maximizing a-posterior (MAP) as equation (2.2), one can obtain another estimate of θ, i.e.,

$$\begin{aligned} \hat{\theta}_{MAP} &= \arg\max_{\theta} p(\theta|\mathbf{x}) = \arg\max_{\theta} \frac{p(\mathbf{x}|\theta)p(\theta)}{p(\mathbf{x})} \\ &= \arg\max_{\theta} p(\mathbf{x}|\theta)p(\theta) \end{aligned} \tag{2.5}$$

Both $p(x|\theta_{MLE})$ and $p(x|\theta_{MAP})$ can be used to make an inference on x. For a large number of the training set $\mathbf{x}$, θ_{MLE} and θ_{MAP} tend to converge to the same value; but for a limited number of the training set, θ_{MLE} may achieve better performance if the prior is appropriate.

As with the MAP method, the Bayesian inference techniques utilize a "posterior". However, the watershed between Bayesian and non-Bayesian inference is not drawn on whether a prior is used. A Bayesian inference does not first find the optimum parameter. Instead, it integrates all of the model parameters out, based on the belief (probability) of occurrence of each model. More precisely, the inference probability of $\tilde{x}$ is

$$p(\tilde{x}|\mathbf{x}) = \int p(\tilde{x}|\theta)p(\theta|\mathbf{x})d\theta. \tag{2.6}$$

There are two important factors that determine the application scope of Bayesian inference: One is the prior selection, and another one is the problem of how to efficiently compute the integral (over the parameter) of Bayesian inference.

In the following sections, two prior selection methods will be introduced, and then some of the inference methods will be explained in detail.

2.2 Prior Selection

In this section, two priors will be introduced: the conjugate prior and reference prior. The conjugate prior has the advantage of reducing the computational complexity of the posterior. The reference prior is an "uninformative prior" that has a maximum distance from the posterior under Kullback-Leibler divergence.

2.2.1 Conjugate Prior

The conjugate prior has the exact form as the posterior. For example, for the k dimension multinomial distribution

$$p(x|\boldsymbol{\theta}) = \frac{\Gamma(\sum_i x_i + 1)}{\prod_i^k \Gamma(x_i + 1)} \prod_{i=1}^{k} \theta_i^{x_i}, \tag{2.7}$$

its conjugate prior Dirichlet distribution has the form

$$\mathrm{Dir}(x|\boldsymbol{\alpha}) = \frac{1}{\mathrm{B}(\boldsymbol{\alpha})} \prod_{i=1}^{k} x_i^{\alpha_i - 1}, \tag{2.8}$$

where $\mathrm{B}(\cdot)$ is the Beta function, which can be written in terms of the gamma function

$$\mathrm{B}(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)}. \tag{2.9}$$

Since the prior and the posterior have the same form of distribution, to compute the posterior, one only needs to update the parameters of this distribution.

2.2.2 Reference Prior

Reference priors were firstly introduced by Jose Bernardo [31]. The idea of reference priors is that the prior should not strongly influence the posterior distribution, and such a prior is called the *uninformative prior* [32]. Divergences, such as the Kullback-Leibler (KL) divergence, can be used to measure the influence of the priors on the posteriors. Suppose the random variable X is parameterized by Θ, and $T(X)$ is the sufficient statistic of X.A reference prior $p(\theta)$ should be chosen that has a maximum distance to $p(\theta|t)$ under

KL divergence

$$\int p(\theta|t) \log \frac{p(\theta|t)}{p(\theta)} \, d\theta, \tag{2.10}$$

whose expectation over the distribution of T can be written as

$$\begin{aligned} I(\Theta; T) &= \int p(t) \int p(\theta|t) \log \frac{p(\theta|t)}{p(\theta)} \, d\theta \, dt \\ &= \int \int p(\theta, t) \log \frac{p(\theta|t)}{p(\theta)} \, d\theta \, dt \end{aligned}, \tag{2.11}$$

which is the mutual information between θ and t. Under (2.11), one wants to choose $p^*(\theta)$ such that

$$p^*(\theta) = \arg\max_{p(\theta)} I(\Theta, T). \tag{2.12}$$

For one-dimensional parameters, the reference priors are equal to Jeffreys priors, which have the form

$$p(\theta) \propto I(\theta)^{\frac{1}{2}}, \tag{2.13}$$

where $I(\theta)$ is the Fisher information, which can be expressed as

$$I(\theta) = -\mathbb{E}_\theta \left[\frac{d^2 \log p(\mathbf{x}|\theta)}{d\theta^2} \right] \tag{2.14}$$

.

2.3 Inference Methods

A factor graph can model the correlation of a large set of variables in an intuitive manner. The variables in the factor graph can be either hidden variables or observable variables, and they will be connected with the factors,

which take those variables as the arguments. Based on the factor graph, the message passing algorithm can be implemented, by which the beliefs of all the hidden variables can be derived. In this section, the factor graph will be introduced, and then the message passing algorithm in the factor graph will be derived.

2.3.1 Factor Graph

A factor graph represents the factorization of the joint distribution and is a bipartite graph composed of two different types of nodes: variable nodes and factor nodes. As one type of graphical model, the correlations among the variables can be easily observed. In a nutshell, variable nodes represent random variables in the problem, whereas factor nodes connecting immediately related variable nodes explain the correlations among them. Each factor node is assigned with a factor function that contains the variables connected to the factor node as arguments. The only assumption of a factor graph is that the joint probability of all the variables should be equal to the product of all the factor functions.

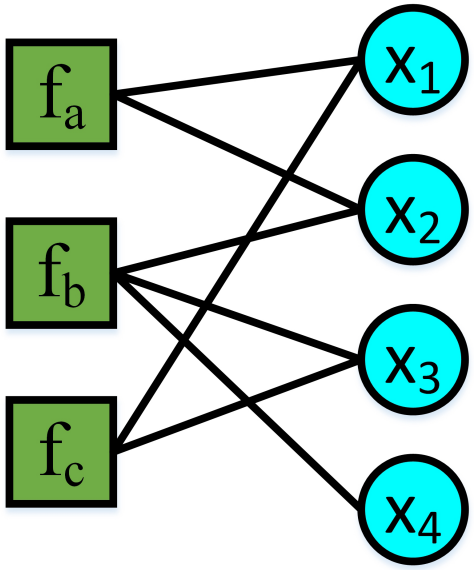


Figure 2.1: An Example of a Factor Graph.

Figure 2.1 shows an example of a factor graph. In this factor graph, there are 4 variables $\{x_1, x_2, x_4, x_4\}$, and 3 factors $\{f_a, f_b, f_c\}$. From the factor graph, all of the factors are multiplied to obtain the joint distribution

Table 2.1: Belief Propagation Algorithm.

Inputs: $f_1(\mathbf{x}_1)$, $f_2(\mathbf{x}_2)$, $\ldots$, $f_M(\mathbf{x}_M)$.
Initialization: initialize each message $m_{i\to a}(x_i)$ $\forall(a,i)$.
Repeat until all $m_{a\to i}(x_i)$ **converge or reach maximum iterations:**

- update all the messages $m_{a\to i}(x_i)$ via Equation (2.18);
- update all the messages $m_{i\to a}(x_i)$ via Equation (2.17);

Output: the beliefs $b(x_i)$ $\forall i$ via Equation (2.19).

$p(x_1, x_2, x_3, x_4)$ as

$$p(x_1, x_2, x_3, x_4) = \frac{f_a(x_1, x_2) f_b(x_2, x_3, x_4) f_c(x_1, x_3)}{Z}, \tag{2.15}$$

where Z is the normalization constant, and can be computed as

$$Z = \sum_{x_1, x_2, x_3, x4} f_a(x_1, x_2) f_b(x_2, x_3, x_4) f_c(x_1, x_3). \tag{2.16}$$

2.3.2 Belief Propagation Algorithm

In Bayesian inference, one usually needs to obtain the distributions of some specific random variables, which requires one to summarize/integrate the joint distribution over other random variables. However, due to the complex form of joint distribution, direct summations/integrations are usually infeasible. Belief propagation is an efficient algorithm that can obtain the beliefs (marginal distribution) of the random variables by implementing a sum-product algorithm in the factor graph.

In the factor graph, the belief propagation algorithm defines two types of message flows: One is the message from the factor to the variable, and

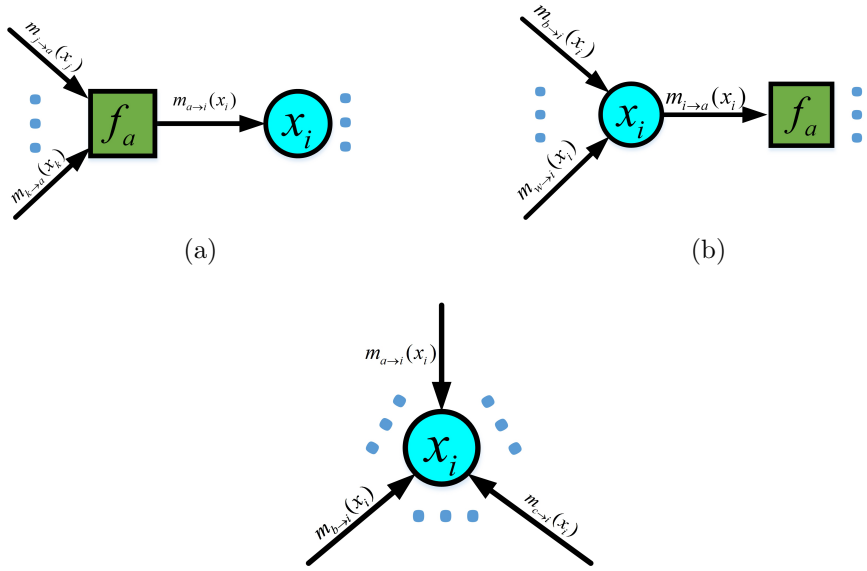


Figure 2.2: (a)message from factor to variable; (b) message from variable to factor; (c) variable belief.

another is the message from the variable to the factor. Suppose there are N random variables $\{X_1, \ldots, X_N\}$ and M factors $\{f_1(\mathbf{x}_1), \ldots, f_M(\mathbf{x}_M)\}$, where $\mathbf{x}_a \subset \{x_1, \ldots, x_N\}$. For the BP algorithm, each message in the factor graph will be updated according to the following rules: the message from the variable to the factor

$$m_{i\to a} \propto \prod_{b\in N(i)\backslash a} m_{b\to i}(x_i), \tag{2.17}$$

and the message from the factor to the variable

$$m_{a\to i} \propto \sum_{\mathbf{x}_a\backslash x_i} \left(f_a(\mathbf{x}_a) \prod_{j\in N(a)\backslash i} m_{j\to a}(x_j) \right), \tag{2.18}$$

where the notation $N(i)\backslash a$ denotes the set of all neighbors of node i excluding factor node a, and $\sum_{\mathbf{x}_a\backslash x_i}$ denotes a sum over all the variables in $\mathbf{x}_a$ except

x_i. One uses i, j as the index of set $\{1, \ldots, N\}$ and a, b as the index of set $\{1, \ldots, M\}$. Moreover, the belief of x_i is

$$b(x_i) \propto \prod_{a \in N(i)} m_{a \to i}(x_i). \tag{2.19}$$

Details of the BP algorithm are listed in Table 2.1.

2.3.3 Variational Bayes (VB)

The BP algorithm can only compute the discrete variables, or continuous variables with limited forms such as the Gaussian distribution. Even for discrete random variables, the computation complexity will increase exponentially with the size of variable's the parameters. To cope with these complex situations, one can approximate these true distributions.

Generally, there are two types of approximate methods: the random approximate method and the deterministic approximate method. The random approximate method usually utilizes sampling methods, such as the Gibbs sampling method [33] and Metropolis-Hastings [34] algorithms. Although the sampling method may theoretically approximate the true distribution with arbitrarily high precision as the number of samples increases, the complexity is usually much higher than that of the deterministic approximate method. In addition, the designs of the sampling methods are very tricky. For example, the Metropolis-Hastings method samples from a self-constructive Markov network, which determines the convergence speed and precision. However, there is no general design method for this Markov network. The deterministic approximate methods commonly have lower complexity. In addition, one may only concentrate on the main aspect of the inference variable, and the

deterministic approximate method will meet our demands with high speed.

Based on above reasons, deterministic approximate methods will be adopted to deal with the complex distribution in the factor graph of this dissertation. Two of the deterministic approximate methods will be utilized in this dissertation: variational Bayes (VB) and expectation propagation (EP).

Suppose one wants to make inferences on the hidden random variable Z given X, and there are N observations of X, which are $\mathbf{x} = \{x_1, \ldots, x_N\}$. Due to the complex form of the posterior $p(z|x)$, one will find a favorable distribution $q(z)$ to approximate $p(z|x)$ so that make inferences. Generally, one obtains the following form

$$\mathrm{KL}(q||p) = \ln p(x) + \mathcal{L}(q(z)), \tag{2.20}$$

where

$$\mathrm{KL}(q||p) = \int q(z) \ln \left(\frac{q(z)}{p(z|x)} \right) dz, \tag{2.21}$$

$$\mathcal{L}(q) = \int q(z) \ln \left(\frac{q(z)}{p(x,z)} \right) dz. \tag{2.22}$$

$\mathrm{KL}(q||p)$ is the Kullback-Leibler divergence operation, which measures the similarity of the distribution between q and p. $\mathrm{KL}(q||p)$ is always non-negative, and is equal to 0 when $q(x) = p(x)$. Comparing with the form of the KL divergence, it is known that $\mathcal{L}(q)$ is the KL divergence between $q(z)$ and $p(x, z)$, which indicates $\mathcal{L}(q) \geq 0$ and $\mathrm{KL}(q||p) \geq \ln p(x) = \text{Constant}$. Hence, $\ln p(x)$ is the lower bound of $\mathrm{KL}(q||p)$.

The optimization (minimization) function is (2.20). However, if one knows the exact posterior $p(z|x)$ in (2.21), then one can make inferences on Z directly. Instead, one can minimize (2.22). In the following, let's suppose

z is a vector, i.e. $\mathbf{z}$, and $q(\mathbf{z})$ can be factorized as

$$q(\mathbf{z}) = \prod_{i=1}^{M} q_i(\mathbf{z}_i), \tag{2.23}$$

where $\mathbf{z}_i \subset \mathbf{z}$ and $i \in \{1, \ldots, M\}$. Substituting (2.23) into (2.22) and minimizing $\mathcal{L}(q)$ with respect to each $q(\mathbf{z}_j)$, one can obtain the optimal solution $q^*(\mathbf{z}_j)$ as

$$q^*(\mathbf{z}_j) = \frac{\exp\left(\mathbb{E}_{i \neq j}[\ln p(\mathbf{x}, \mathbf{z})]\right)}{\int \exp\left(\mathbb{E}_{i \neq j}[\ln p(\mathbf{x}, \mathbf{z})]\right) d\mathbf{z}_j}, \tag{2.24}$$

where

$$\mathbb{E}_{i \neq j}[\ln p(\mathbf{x}, \mathbf{z})] = \int \ln p(\mathbf{x}, \mathbf{z}) \prod_{i \neq j} q_i d\mathbf{z}_i. \tag{2.25}$$

One can see from (2.25) that the solutions of all the $q^*(\mathbf{z}_j)$ are coupled with the others. To solve this issue, one can iteratively run through each of the $q^*(\mathbf{z}_j)$ until they all converge. In fact, this iteration algorithm is guaranteed to coverage [33].

2.3.4 Expectation Propagation (EP)

The joint distribution of $\mathbf{z}$ and N i.i.d. observations $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is given by

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z}) \prod_{i=1}^{N} p(\mathbf{x}_i | \mathbf{z}). \tag{2.27}$$

Let $t_0(\mathbf{z}) = p(\mathbf{z})$ and $t_i(\mathbf{z}) = p(\mathbf{x}_i|\mathbf{z})$. Then (2.27) can be written as

$$p(\mathbf{x}, \mathbf{z}) = \prod_{i=0}^{N} t_i(\mathbf{z}). \tag{2.28}$$

Table 2.2: Expectation Propagation Algorithm.

Inputs: $t_1(\mathbf{z}), t_2(\mathbf{z}), \ldots, t_N(\mathbf{z})$
Initialization: initialize $\tilde{t}_1(\mathbf{z}), \tilde{t}_2(\mathbf{z}), \ldots, \tilde{t}_N(\mathbf{z})$; initialize $q(\mathbf{z})$ according to (2.30).
Repeat until all $\tilde{t}_i$ converge or reach maximum iterations:

- Choose a $\tilde{t}_i$ to refine;
- Obtain the distribution $q^{\setminus i}(\mathbf{z})$:

$$q^{\setminus i}(\mathbf{z}) \propto \frac{q(\mathbf{z})}{\tilde{t}_i(\mathbf{z})}; \tag{2.26}$$

- Approximate $q^{\setminus i}(\mathbf{z})\tilde{t}_i(\mathbf{z})$ with new $q(\mathbf{z})$;
- Compute normalizer $Z_i = \int t_i(\mathbf{z}) q^{\setminus i}(\mathbf{z}) d\mathbf{z}$;
- Update $\tilde{t}_i = Z_i q(\mathbf{z}) / q^{\setminus i}(\mathbf{z})$

Output: $q(\mathbf{z})$.

The EP algorithm approximates the posterior $p(\mathbf{z}|D)$ with $q(\mathbf{z})$ under

$$\mathrm{KL}(p(\mathbf{x}|\mathbf{z})||q(\mathbf{z})) = \mathrm{KL}\left(\frac{\prod_{i=0}^{N} t_i(\mathbf{z})}{p(\mathbf{x})}||q(\mathbf{z})\right), \tag{2.29}$$

which has the reverse form of (2.21). However, the form of (2.29) is intractable. To make (2.21) tractable, the EP algorithm factorizes $q(\mathbf{z})$ as

$$q(\mathbf{z}) = \frac{\prod_{i=1}^{N} \tilde{t}_i(\mathbf{z})}{\int \prod_{i=1}^{N} \tilde{t}_i(\mathbf{z})}, \tag{2.30}$$

and let each $\tilde{t}_i(\mathbf{z})$ approximate each $t_i(\mathbf{z})$, and an approximate iteration algorithm will be listed in Table 2.2 [35].

The commonly used distribution $\tilde{t}_i$ is the exponential family distribution

$$f(\mathbf{z}|\theta) = h(\mathbf{z}) g(\theta) \exp\left(\theta^T \mathbf{u}(\mathbf{z})\right). \tag{2.31}$$

Once $q(\mathbf{z})$ is obtained, one can make inferences on Z and compute $p(\mathbf{x})$ as

$$p(\mathbf{x}) \approx \int q(\mathbf{z}) d\mathbf{z}. \tag{2.32}$$

CHAPTER 3

ADAPTIVE SLEPIAN-WOLF DECODING FOR TWO BINARY SOURCES

3.1 Introduction

The DSC with two sources problem can be modeled as a channel coding problem: The two sources of the DSC can be viewed as the input and output of a channel; The correlation of the two sources can be viewed as the property of the channel. In this chapter, the correlation of the two binary sources will be modeled as the Binary asymmetrical channel (BAC) problem, which has a wide application scope.

3.1.1 Related Work

Since the DSC problem can be modeled as a channel coding problem, the algorithms of the channel problem can be directly applied to the DSC problem. In [36,37], the DSC problem was modeled as a virtual BSC channel. To estimate the parameter (correlation in the DSC) in the BSC channel, various methods have been proposed. In [38–40], the crossover probability of the BSC is assumed to be constant in the context of the source coding. [18,41,42] make further progress and assume that the crossover probability changes over time. However, if the correlation of the two sources does not satisfy the symmetrical condition, the BSC model will lead to a poor result. To be more applicable in practice, the symmetrical condition is not assumed in this section, and a model of the correlation of the two sources as the binary asymmetrical channel (BAC) is used.

In the rest of this chapter, an adaptive DSC algorithm for two binary sources modeled as BAC is proposed. First, the BAC model is introduced. Then, the proposed algorithms are explained in detail; Finally, the experimental results and an analysis of the proposed algorithms are given.

3.2 Problem Formulation

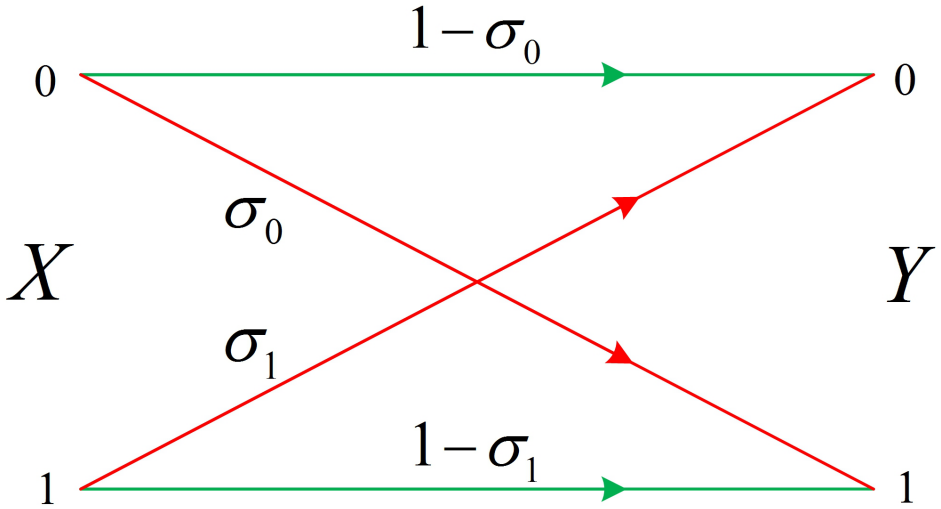


Figure 3.1: Binary asymmetric channel

In Figure 3.1, suppose the BAC channel has a probability σ_0 such that an input 0 will be flipped into a 1 and probability σ_1 such that an input 1 will be flipped into a 0.

Unlike with the BSC [4], the conditional probability of the received bit value, given the send bit $P(y|x)$, can be expressed as the following piecewise function:

$$f(x, y, \sigma_0, \sigma_1) = P(y|x, \sigma_0, \sigma_1) = \begin{cases} \sigma_0^y (1-\sigma_0)^{(1-y)} & : x = 0 \\ \sigma_1^{1-y} (1-\sigma_1)^y & : x = 1 \end{cases}. \tag{3.1}$$

The above piecewise function can be unified as

$$f(x, y, \sigma_0, \sigma_1) = \sigma_x^{x \oplus y} \left(1 - \sigma_x\right)^{1 \oplus x \oplus y}. \tag{3.2}$$

Suppose

$$p(x) = \begin{cases} \pi_0 & : x = 0 \\ \pi_1 & : x = 1 \end{cases}, \tag{3.3}$$

then the joint distribution $p(x, y)$ is given as

$$\begin{aligned} p(x, y) &= p(x)p(y|x) \\ &= \pi_x \sigma_x^{x \oplus y} \left(1 - \sigma_x\right)^{1 \oplus x \oplus y} \\ &= \left(\pi_0 \sigma_0^y (1 - \sigma_0)^{(1-y)}\right)^{(1-x)} \left(\pi_1 \sigma_1^{(1-y)} (1 - \sigma_1)^y\right)^x. \end{aligned} \tag{3.4}$$

In the following, the correlation between the sources X and Y is modeled as the BAC channel in Equation (3.2), and an adaptive source coding method will be proposed.

For simplicity, it is assumed that both π_0 and π_1 in (3.4) are known in advance, and that both are equal to 0.5. However, the proposed VB-based method will assume that π_0 and π_1 are unknown and will estimate them, and priors will be given for this algorithm for Bayesian inference.

3.3 Proposed Methods

To encode the source data $\mathbf{x}$ (vector form), the low-density parity-check code (LDPC) is utilized. LDPC is one of the linear block codes, which can be defined by a sparse parity-check matrix. Given an LDPC parity-check matrix $\mathbf{H}$, the syndrome $\mathbf{s}$ (codeword) can be easily obtained by $\mathbf{s} = \mathbf{H} \times \mathbf{x}$.

To decode the syndromes, a factor graph (see section 2.3.1) is first pro-

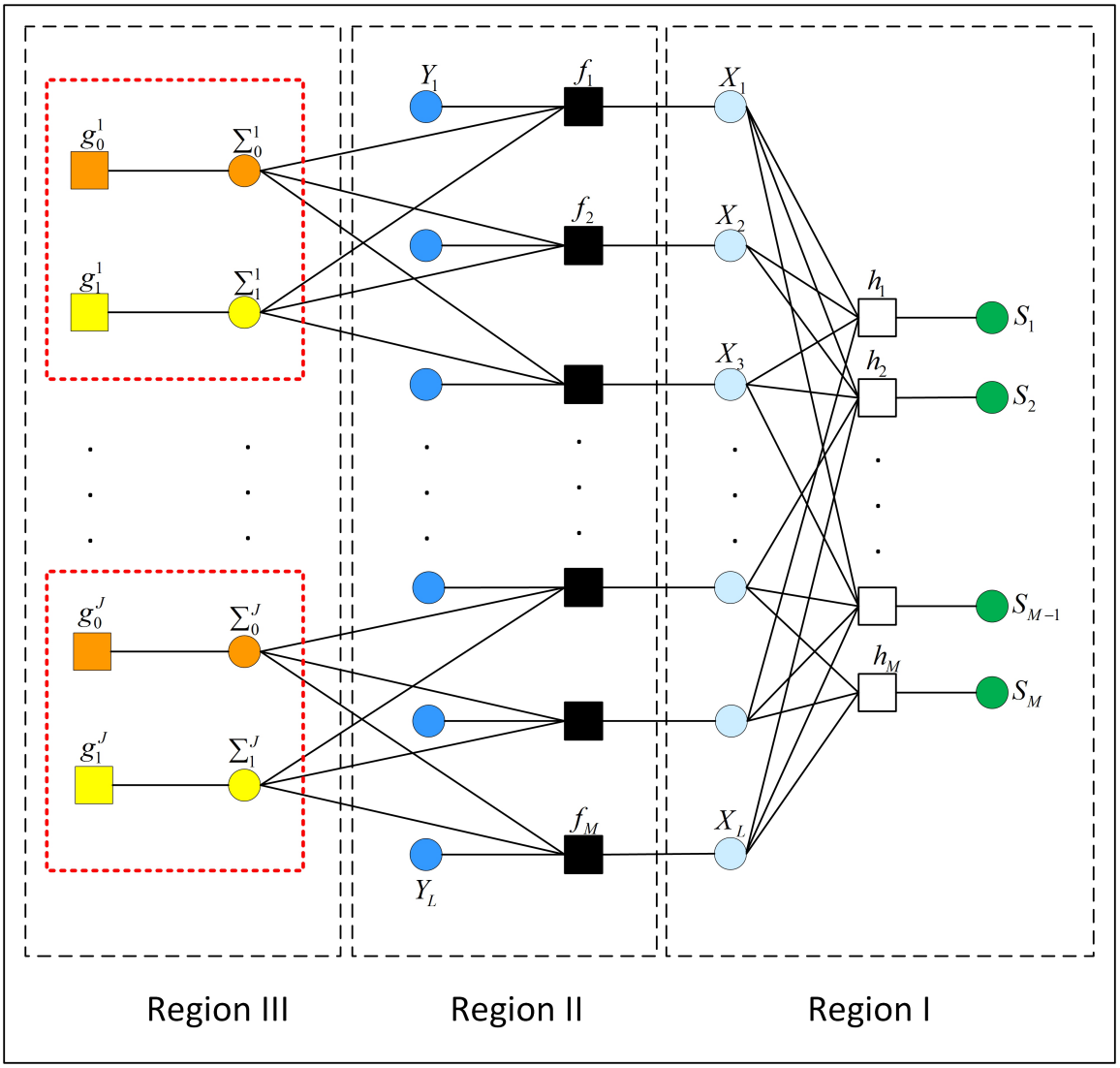


Figure 3.2: Factor graph of binary asymmetric channel (BAC) sources

posed in Figure 3.2. As shown in Figure 3.2, the factor graph can be divided into three regions: Region I is the low-density parity-check code (LDPC) area, which encodes the constraints between the sources $\{x_1, \ldots, x_L\}$ and the compressed syndromes $s_1, \ldots, s_K$. Region II is the correlation between the sources $\{x_1, \ldots, x_L\}$ and the side information (SI) $\{y_1, \ldots, y_L\}$. The BAC channel parameters σ_0 and σ_1 are estimated in the Region III. In the following sections, each of these three regions will be explained one by one.

Table 3.1: Summary of LDPC decoding using BP

Input: the syndromes $s_1, \ldots, s_K$.
Initialization:

$$L_{ai} = 0, \forall \text{ check node } a \tag{3.7}$$

.
Repeat until all messages $m_{a\to i}(x_i)$ and $m_{i\to a}(x_i)$ are converged or reach maximum number of iterations:

- update all the messages $m_{i\to a}(x_i)$ via Equation (3.10);
- update all the messages $m_{a\to i}(x_i)$ via Equation (3.11);

Output: the log-likelihood ratio for variable node i can be updated according to Equation (3.12).

3.3.1 Region I - LDPC Codes

In Region I of Figure 3.2, there are L-bit source and M-bit syndrome, which are connected by the check nodes $\{h_1, \ldots, h_M\}$, and the sparse parity-check matrix $\mathbf{H}$ has the size of $L \times M$, i.e., $\mathbf{H} \in \mathbb{F}_2^{L\times M}$.

To satisfy the LDPC constraint, each factor h_a ($a \in \{1, \ldots, M\}$) is defined as

$$h_a(\mathbf{x}_a, s_a) = \sum \mathbf{x}_a + s_a = 0. \tag{3.5}$$

All of the summations in Equation (3.5) are implemented in $\mathbb{F}_2$, and $\mathbf{x}_a$ denotes all of the sources that connect with the factor h_a. The vectorized form of (3.5) can be written as

$$\mathbf{H} \times \mathbf{x} + \mathbf{s} = \mathbf{0} \tag{3.6}$$

The BP algorithm (see section 2.3.2) will be applied to solve the LDPC decoding, and the detailed steps are listed in Table 3.1.

Initially, no information is obtained from the check factor node. Therefore

the message from those nodes should bear no information, i.e., $m_{ai}(1) = m_{ai} = 0.5$ for any factor node a to variable node i.

After assigning the initial values to the messages from factor to variable, we can update the messages from variable to factor and the messages from factor to variable alternatively as described in section 2.3.2. Since the unknown variables are binary, it is more convenient to represent the messages using likelihood or log-likelihood ratios. Define

$$\begin{aligned} l_{ai} &\triangleq \frac{m_{ai}(0)}{m_{ai}(1)}, \\ L_{ai} &\triangleq \log l_{ai} \end{aligned} \tag{3.8}$$

and

$$\begin{aligned} l_{ia} &\triangleq \frac{m_{ia}(0)}{m_{ia}(1)} \\ L_{ia} &\triangleq \log l_{ia} \end{aligned} \tag{3.9}$$

for any variable node i and factor node a.

Therefore, the message from variable node i to check node a can be updated as

$$L_{ia} \leftarrow \sum_{b \in N(i) \backslash a} L_{bi}. \tag{3.10}$$

For the message from the check node a to the variable node i, according to Equation (3.5), a simplified update rule can be divided as follow

$$L_{ai} \leftarrow \begin{cases} 2 \tanh^{-1} \left(\prod_{j \in N(a) \backslash i} \tanh \left(\frac{L_{ja}}{2} \right) \right) & : s_a = 0 \\ 2 \tanh^{-1} \left(- \prod_{j \in N(a) \backslash i} \tanh \left(\frac{L_{ja}}{2} \right) \right) & : s_a = 1 \end{cases}. \tag{3.11}$$

The belief of variable node i can be updated as

$$\beta_i = \sum_{a \in N(i)} L_{ai}. \tag{3.12}$$

Hard thresholding can be applied to the final belief to estimate the value of each bit. That is,

$$\hat{x}_i = \begin{cases} 0 & : \text{if } \beta_i \geq 0 \\ 1 & : \text{otherwise} \end{cases}. \tag{3.13}$$

3.3.2 Region II - Correlation between the sources and the side information

For the messages between the Region II and Region III, the deterministic approximate inference methods - the EP and VB algorithms will be applied to update these messages. The details of the updating rules will be derived in the next subsection. In this subsection, only the message sent from the Region II to Region I, i.e. $m_{f_i \to X_i}(x_i)$, will be derived.

For both of the deterministic algorithms, each message sent from the Region III to Region II has the form of Beta distribution. For example, the messages $m_{\Sigma_0^j \to f_i}(\sigma_0)$ and $m_{\Sigma_1^j \to f_i}(\sigma_1)$ can be given as

$$\begin{aligned} m_{\Sigma_0^j \to f_i}(\sigma_0) &= \text{Beta}(\sigma_0 | \alpha_{0,i}^j, \beta_{0,i}^j) = \frac{1}{\text{B}(\alpha_{0,i}^j, \beta_{0,i}^j)} \sigma_0^{\alpha_{0,i}^j - 1} (1 - \sigma_0)^{\beta_{0,i}^j - 1} \\ m_{\Sigma_1^j \to f_i}(\sigma_1) &= \text{Beta}(\sigma_1 | \alpha_{1,i}^j, \beta_{1,i}^j) = \frac{1}{\text{B}(\alpha_{1,i}^j, \beta_{1,i}^j)} \sigma_1^{\alpha_{1,i}^j - 1} (1 - \sigma_1)^{\beta_{1,i}^j - 1} \end{aligned}, \tag{3.14}$$

where $\text{B}(\cdot)$ is defined in (2.9).

Then according to Equation (2.18), we have

$$\begin{aligned} m_{f_i \to X_i}(x_i) &= \int m_{\Sigma_0^j \to f_i}(\sigma_0) m_{\Sigma_1^j \to f_i}(\sigma_1) f(x, y, \sigma_0, \sigma_1) d\sigma_0 d\sigma_1 \\ &= \int \mathrm{Beta}(\sigma_0)\mathrm{Beta}(\sigma_1) \sigma_x^{x \oplus y} \left(1 - \sigma_x\right)^{1 \oplus x \oplus y} d\sigma_0 d\sigma_1 \end{aligned} . \tag{3.15}$$

By using the following properties

$$\begin{aligned} \mathrm{B}(\alpha, \beta) &= \int \mathrm{Beta}(\sigma | \alpha, \beta) d\sigma \\ \mathrm{B}(\alpha + 1, \beta) &= \mathrm{B}(\alpha, \beta) \frac{\alpha}{\alpha + \beta} \\ \mathrm{B}(\alpha, \beta + 1) &= \mathrm{B}(\alpha, \beta) \frac{\beta}{\alpha + \beta} \end{aligned} , \tag{3.16}$$

The ratio form (see (3.8)) of $m_{f_i \to x_i}(x_i)$ in Equation (3.15) can be simplified as

$$m_{f_i \to X_i}(x_i) = \begin{cases} \frac{\beta_{0,i}}{\alpha_{1,i}} \frac{\alpha_{1,i} + \beta_{1,i}}{\alpha_{0,i} + \beta_{0,i}} : y = 0 \\ \frac{\alpha_{0,i}}{\beta_{1,i}} \frac{\alpha_{1,i} + \beta_{1,i}}{\alpha_{0,i} + \beta_{0,i}} : y = 1 \end{cases} . \tag{3.17}$$

3.3.3 Region III

To make the algorithm more adaptive, in Figure 3.2, the channel parameters σ_0 and σ_1 are assumed to be unknown, and both of these parameters will be estimated. In addition, it is assumed that the parameters σ_0 and σ_1 are changing over time j, i.e., σ_0^j and σ_1^j. Here, the number of factor nodes in Region II connecting to each variable node σ_k^j is called the connection ratio C, where $k \in \{0, 1\}$. In Figure 3.2, it is seen that the connection ratio is 3.

The Bayesian inference requires one to estimate the posterior distribution $p(\sigma_k^j | \mathbf{y}_j)$, where $\mathbf{y}_j = \{y_i | i \in N^{\backslash g_k^j}(\Sigma_k^j)\}$, and $N^{\backslash g_k^j}(\Sigma_k^j)$ denotes the set of all neighboring indices for a variable node σ_k^j excluding g_k^j. The posterior is given as

$$
\begin{aligned}
p(\sigma_k^j|\mathbf{y}_j) &= \frac{1}{Z_j} \prod_{i \in \mathcal{N}^{\setminus g_k^j(\Sigma_k^j)}} p(\sigma_k^j) p(y_i|\sigma_k^j) \\
&= \frac{1}{Z_j} \prod_{i \in \mathcal{N}^{\setminus g_k^j(\Sigma_k^j)}} \sum_{x_i} p(\sigma_k^j) p(x_i) p(y_i|x_i, \sigma_k^j) \\
&= \frac{1}{Z_j} g(\sigma_k^j) \prod_{i \in \mathcal{N}^{\setminus g_k^j(\Sigma_k^j)}} \sum_{x_i} m_{X_i \to f_i}(x_i) p(y_i|x_i, \sigma_k^j),
\end{aligned}
\tag{3.18}
$$

where $Z_j = \int \prod_{i \in \mathcal{N}^{\setminus g_k^j(\Sigma_k^j)}} p(\sigma_k^j) p(y_i|\sigma_k^j) d\sigma_k^j$ is a normalization constant, $p(\sigma_k^j) = g(\sigma_k^j)$, and $p(x_i)$ is captured by the message $m_{X_i \to f_i}(x_i)$ with binary sources x_i taking 0 or 1. The connection ratio C equals to $|\mathcal{N}^{\setminus g_k^j}(\Sigma_k^j)|$.

According to Equation (3.18), the exact form of $p(\sigma_k^j|\mathbf{y}_j)$ contains 2^C terms, which are infeasible to estimate when C is large, say 50. Considering the trade off between complexity and accuracy, deterministic approximate methods are introduced. Two deterministic approximate methods are adopted, and they are Expectation propagation (EP) (see Section 2.3.3) and Variation Bayesian (VB) (see Section 2.3.4). In the following, I will illustrate how to apply both of them to estimate the BAC parameters and derive each of the both updating equations for message passing.

Estimating parameters by EP

Only the estimate method for parameter σ_0^j $(1 \leq j \leq J)$ will be provided, and the estimate method for σ_1^j $(1 \leq j \leq J)$ can follow the same manners. In addition, all of the following steps are generalized for all j, hence the index j will be omitted. For example, σ_0^j is shorted as σ_0.

For σ_0, the Equation (3.18) can be rewritten as the following

$$p(\sigma_0|\mathbf{y}) = \frac{1}{Z} m_{g_0 \to \Sigma_0}(\sigma_0) \prod_{i \in \mathcal{N} \setminus g_0(\Sigma_0)} m_{f_i \to \Sigma_0}(\sigma_0). \tag{3.19}$$

The generalized EP algorithm is listed in Table 2.2. For this specific problem, $q_i(\sigma_0)$ will be used to estimate each $m_{f_i \to \Sigma_0}(\sigma_0)$, and confine all of $q_i(\sigma_0)$ into beta distribution.

The prior $g(\sigma_0)$ is given as the reference prior. Since the beta distribution is used to estimate the likelihood function (3.2), according to Equation (2.13), $g(\sigma_0)$ can be calculated as

$$g(\sigma_0) = \frac{1}{B(\alpha_0, \beta_0)} \sigma_0^{\alpha_0 - 1} (1 - \sigma_0)^{\beta_0 - 1}, \tag{3.20}$$

where $\alpha_0 = \beta_0 = 0.5$. In addition, it can be seen that the reference prior (3.20) has the same form as conjugate prior of beta distribution, which will lower the computational complexity.

In the process of estimating σ_0, the approximate distribution for σ_1 is fixed, and let's suppose

$$q(\sigma_1) = \frac{1}{\mathrm{B}(\alpha_1, \beta_1)} \sigma_1^{\alpha_1 - 1} (1 - \sigma_1)^{\beta_1 - 1}. \tag{3.21}$$

The following is the detail of the proposed EP algorithm for estimating σ_0.

1. Initialize the prior $g_0(\sigma_0)$ according to Equation (3.20).

2. Initialize the likelihood messages as

$$q_i(\sigma_0) = z_{0,i} \sigma_0^{\alpha_{0,i} - 1} (1 - \sigma_0)^{\beta_{0,i} - 1} \tag{3.22}$$

with $\alpha_{0,i} = 1$, $\beta_{0,i} = 1$ and $z_{0,i} = 1$.

3. Initialize the posterior distribution as

$$q(\sigma_0) = g_0(\sigma_0). \tag{3.23}$$

4. For 1 to the maximum number of iteration

   For $i \in \mathcal{N}^{\backslash g_0}(\sigma_0)$

   (a) Remove $q_i(\sigma_0)$ from the posterior $q(\sigma_0)$

$$\begin{aligned} \alpha_0^{'} &= \alpha_0 - (\alpha_{0,i} - 1); \\ \beta_0^{'} &= \beta_0 - (\beta_{0,i} - 1). \end{aligned} \tag{3.24}$$

   (b) Update α_0 and β_0 according to moment matching method, which are

$$\alpha_0 = \frac{m_1 (m_1 - m_2)}{m_2 - m_1^2}; \tag{3.25}$$

   and

$$\beta_0 = \alpha_0 \left(\frac{1}{m_1} - 1 \right), \tag{3.26}$$

   where

$$m_1 = \frac{P_1}{Q_1}, \tag{3.27}$$

$$m_2 = \frac{P_2}{Q_2}, \tag{3.28}$$

$$P_1 = \left(\alpha_0^{'} + y\right)\left(1 + \left(\frac{\alpha_1}{\beta_1}\right)^{2y-1}\right) Lr\left(x\right) + \left(\frac{\alpha_0^{'}}{\beta_0^{'}}\right)^{1-y}\left(\alpha_0^{'} + \beta_0^{'} + 1\right) \tag{3.29}$$

$$Q_1 = \left(\alpha_0^{'} + \beta_0^{'} + 1\right)\left(\left(1 + \left(\frac{\alpha_1}{\beta_1}\right)^{2y-1}\right) Lr\left(x\right) + 1 + \left(\frac{\beta_0^{'}}{\alpha_0^{'}}\right)^{2y-1}\right), \tag{3.30}$$

$$\begin{aligned} P_2 = &\left(\alpha_0^{'} + y\right)\left(\alpha_0^{'} + y + 1\right)\left(1 + \left(\frac{\alpha_1}{\beta_1}\right)^{2y-1}\right) Lr\left(x\right) + \left(\frac{\alpha_0^{'}}{\beta_0^{'}}\right)^{1-y} \\ &\left(\alpha_0^{'} + 1\right)\left(\alpha_0^{'} + \beta_0^{'} + 2\right) \\ &, \end{aligned} \tag{3.31}$$

$$\begin{aligned} Q_2 = &\left(\alpha_0^{'} + \beta_0^{'} + 1\right)\left(\alpha_0^{'} + \beta_0^{'} + 2\right) \\ &\left(\left(1 + \left(\frac{\alpha_1}{\beta_1}\right)^{2y-1}\right) Lr\left(x\right) + 1 + \left(\frac{\beta_0^{'}}{\alpha_0^{'}}\right)^{2y-1}\right), \end{aligned} \tag{3.32}$$

and $Lr(x) = \frac{m_{X_i \to f_i(0)}}{m_{X_i \to f_i(1)}}$

(c) Set approximated message

$$\begin{aligned} \alpha_{0,i} &= \alpha_0 - \left(\alpha^{'} - 1\right), \\ \beta_{0,i} &= \beta_0 - \left(\beta^{'} - 1\right). \end{aligned} \tag{3.33}$$

End For

End For

5. Output $q(\sigma_0)$

Then one can follow the similar above procedures to estimate σ_1, and all of the procedures are the same except the formulas of P_1, Q_1 in (3.27), and P_2, Q_2 in (3.28) should be modified as

$$P_1 = \left(\frac{\alpha_1^{'}}{\beta_1^{'}}\right)^{y} \left(\alpha_1^{'} + \beta_1^{'} + 1\right) Lr\left(x\right) + \left(\alpha_1^{'} + 1 - y\right) \left(1 + \left(\frac{\beta_0}{\alpha_0}\right)^{2y-1}\right), \tag{3.34}$$

$$Q_1 = \left(\alpha_1^{'} + \beta_1^{'} + 1\right) \left(\left(1 + \left(\frac{\alpha_1^{'}}{\beta_1^{'}}\right)^{2y-1}\right) Lr(x) + 1 + \left(\frac{\beta_0}{\alpha_0}\right)^{2y-1}\right), \tag{3.35}$$

$$\begin{aligned} P_2 = &\left(\frac{\alpha_1^{'}}{\beta_1^{'}}\right)^{y} \left(\alpha_1^{'} + 1\right) \left(\alpha_1^{'} + \beta_1^{'} + 2\right) Lr\left(x\right) + \left(\alpha_1^{'} + 1 - y\right) \\ &\left(\alpha_1^{'} + 2 - y\right) \left(1 + \left(\frac{\beta_0}{\alpha_0}\right)^{2y-1}\right) \end{aligned} \tag{3.36}$$

and

$$\begin{aligned} Q_2 = &\left(\alpha_1^{'} + \beta_1^{'} + 1\right) \left(\alpha_1^{'} + \beta_1^{'} + 2\right) \\ &\left(\left(1 + \left(\frac{\alpha_1^{'}}{\beta_1^{'}}\right)^{2y-1}\right) Lr(x) + 1 + \left(\frac{\beta_0}{\alpha_0}\right)^{2y-1}\right). \end{aligned} \tag{3.37}$$

The σ_0 and σ_1 can be estimated alternately until both σ_0 and σ_1 converge or reach the maximum number of iteration.

Estimating parameters by VB

The VB algorithm is also utilized to estimate σ_0 and σ_1. Before continuing, some notations should be classified:

- N is the number of data (connection ratio in Region III of Figure 3.2);
- Define 1-of-2 random variables $\mathbf{x}$ as $\mathbf{x} = (1-x, x)^T$, where x is the values of X in Figure 3.2. It can be seen that only one element of $\mathbf{x}$ equals to 1 and the other one equals to 0. x_{nk} is the $k-th$ element of $\mathbf{x}$ for data n. Note $k = \{0, 1\}$;
- let $\tilde{\sigma_0} = \sigma_0$, $\tilde{\sigma_1} = 1 - \sigma_1$, and $\tilde{\sigma} = [\tilde{\sigma_0}, \tilde{\sigma_1}]^T$.

Considering the dependent relations among $\mathbf{x}$, y, π and $\tilde{\sigma}$, their joint distribution can be factorized as

$$p(\mathbf{x}, y, \pi, \tilde{\sigma}) = p(y|\mathbf{x}, \tilde{\sigma})p(\mathbf{x}|\pi)p(\pi)p(\tilde{\sigma}). \tag{3.38}$$

Then one has

$$p(y|\mathbf{x}, \tilde{\sigma}) = \prod_{n=1}^{N}\prod_{k=0}^{1} \mathrm{Bern}(y_n|\tilde{\sigma_k})^{x_{nk}}, \tag{3.39}$$

where Bern is Bernoulli distribution.

For the distribution $p(\mathbf{x}|\pi)$, $p(\pi)$ and $p(\sigma)$, they are assumed as

$$p(\mathbf{x}|\pi) = \prod_{n=1}^{N}\prod_{k=0}^{1} \pi_k^{x_{nk}}, \tag{3.40}$$

$$p(\pi) = \mathrm{Dir}\left(\pi|\alpha_0, \alpha_1\right), \tag{3.41}$$

$$p(\tilde{\sigma}) = p(\tilde{\sigma}_0)p(\tilde{\sigma}_1) = \mathrm{Beta}(\tilde{\sigma}_0|\beta_{00}, \beta_{01})\mathrm{Beta}(\tilde{\sigma}_1|\beta_{10}, \beta_{11}). \tag{3.42}$$

The exact form of joint distribution (3.38) can be obtained by multiplying (3.39), (3.40), (3.41) and (3.42). However, this multiplication form has much higher complexity, or it is infeasible.

Under VB framework in Section 2.3.3, much more easier approximate distributions for $q(\mathbf{x})$, $q(\pi)$ and $q(\sigma)$ will be given in order to make inference on $\mathbf{x}$, π and σ respectively. The detail of the algorithm is listed in the following

1. Initialize the parameters α_0, α_1, β_{00}, β_{01}, β_{10} and β_{11}:

   Suppose $\{x_1, \ldots, x_N\}$ are current decoded words in Region I of Figure 3.2, then the parameters can be initialized as

$$\begin{aligned}
\alpha_0 &= \frac{\#\{x_n == 0\}}{N} P_\alpha \\
\alpha_1 &= (1 - \alpha_0) P_\alpha \\
\beta_{00} &= \frac{\#\{x_n == 0 \&\& y_n == 1\}}{\#\{x_n == 0\}} P_{\beta_0} \\
\beta_{01} &= (1 - \beta_{00}) P_{\beta_0} \\
\beta_{10} &= \frac{\#\{x_n == 1 \&\& y_n == 1\}}{\#\{x_n == 1\}} P_{\beta_1} \\
\beta_{11} &= (1 - \beta_{10}) P_{\beta_1}
\end{aligned}, \tag{3.43}$$

   where P_α, P_{β_0} and P_{β_1} are constants to adjust the values of the parameters of the distributions. The bigger values the constants are, the more influences the corresponding distributions will take effect on the final estimations.

2. Repeat the following steps until all $q(\cdot)$ are converged or reach the maximum number of iterations.

(a) Update $q(x)$:

$$q(x) = \prod_{n=1}^{N} \prod_{k=0}^{1} r_{nk}^{x_{nk}}, \tag{3.44}$$

where

$$\begin{aligned} r_{nk} &= \frac{\rho_{nk}}{\sum_{j=0}^{1} \rho_{nj}}, \\ \ln \rho_{nk} &= y_n \mathbb{E}[\ln \tilde{\sigma_k}] + (1 - y_n)\mathbb{E}[\ln(1 - \tilde{\sigma_k})] + \mathbb{E}(\ln \pi_k), \end{aligned} \tag{3.45}$$

where

$$\begin{aligned} \mathbb{E}[\ln \tilde{\sigma_k}] &= \psi(\beta_{k0}) - \psi(\beta_{k0} + \beta_{k1}) \\ \mathbb{E}[\ln(1 - \tilde{\sigma_k})] &= \psi(\beta_{k1}) - \psi(\beta_{k0} + \beta_{k1}) \\ \mathbb{E}(\ln \pi_k) &= \psi(\alpha_k) - \psi(\sum_{k=0}^{1} \alpha_k) \end{aligned}, \tag{3.46}$$

where $\psi(\cdot)$ is digamma function.

(b) Update $q(\pi)$:

$$q(\pi) = \mathrm{Dir}(\pi | \alpha_0', \alpha_1'), \tag{3.47}$$

where

$$\begin{aligned} {\alpha_k}' &= \alpha_k + N_k \\ N_k &= \sum_{n=1}^{N} r_{nk} \end{aligned}, \tag{3.48}$$

(c) Update $\tilde{\sigma}$:

$$q(\tilde{\sigma}) = \prod_{k=0}^{1} \mathrm{Beta}(\tilde{\sigma_k} | {\beta_{k0}}', {\beta_{k1}}'). \tag{3.49}$$

where

$$\begin{aligned} \beta_{k0}{}' &= S_k + \beta_{k0} \\ \beta_{k1}{}' &= T_k + \beta_{k1} \end{aligned}, \tag{3.50}$$

where

$$\begin{aligned} S_k &= \sum_{n=1}^{N} y_n r_{nk} \\ T_k &= \sum_{n=1}^{N} (1 - y_n) r_{nk} \end{aligned}. \tag{3.51}$$

3. Output σ_0 and σ_1 as

$$\begin{aligned} \sigma_0 &= \tilde{\sigma_0} \\ \sigma_1 &= 1 - \tilde{\sigma_1} \end{aligned}. \tag{3.52}$$

Both of the EP and VB algorithms can be applied to all of the pair nodes σ_0^j and σ_1^j so that provides the estimates for σ_0^j and σ_1^j to Region II to continue message passing algorithm.

3.4 Experimental Results

The proposed algorithms will be tested in two experiments, and the details of setups are listed in Table 3.2. All of the algorithms are implemented by Java and evaluated on an Intel 3.0 GHz CPU machine.

First, let's set both σ_0^t and σ_1^t as Gaussian distribution $\mathcal{N}(0.15, 0.01^2)$. The length of block N is 10000, and the connection ratio C equals 500. The crossover probabilities σ_0^t and σ_1^t for the proposed algorithm are initialized 0.1 away from the mean of the true probabilities. The algorithm will run 60

Table 3.2: Experimental setups

	Exp I	**Exp II**
σ_0^t	$\mathcal{N}(0.15, 0.01^2)$	$\mathcal{N}(0.2, 0.01^2)$
σ_1^t	$\mathcal{N}(0.15, 0.01^2)$	$\mathcal{N}(0.1, 0.01^2)$
N	10000	
C	500	
# of iterations per each trial	60	
learning period	every 40 iterations	
# of trials	100	

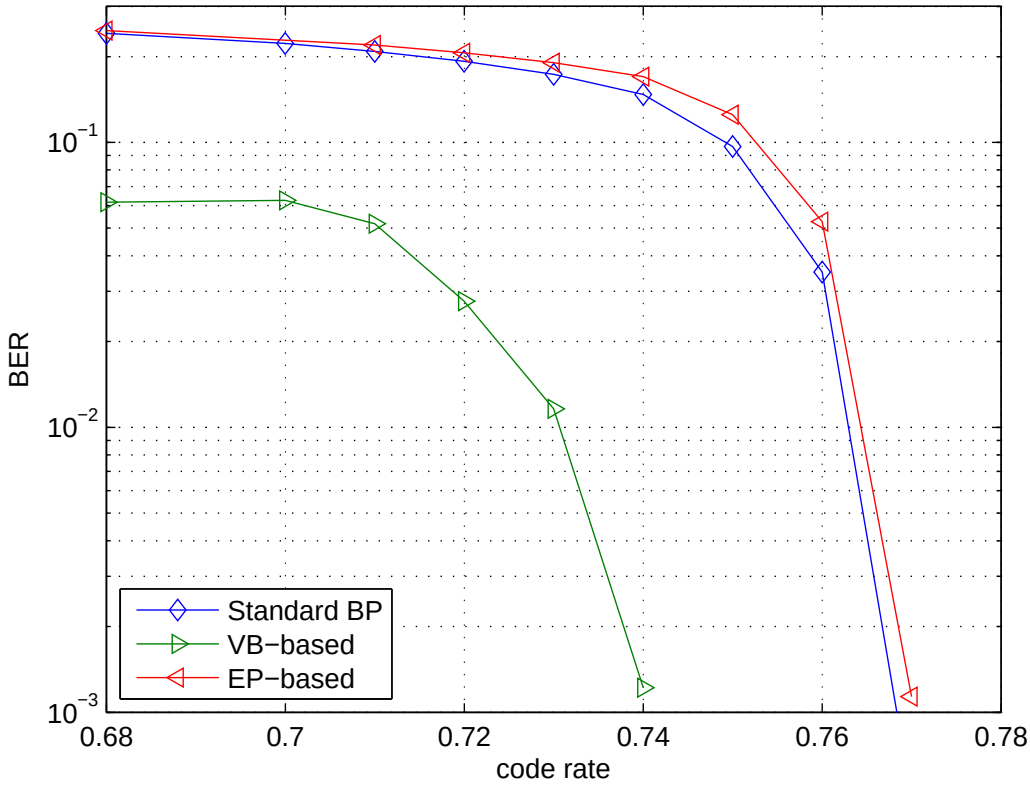


Figure 3.3: Bit error rates versus code rates curves of Exp I

times for each testing, and the VB/EP only run every 40 iterations. All the results are averaged over 100 trials. The bit error rate is plotted in Figure 3.3. To make comparison, standard BP algorithm with no parameter esti-

mation is also implemented. In Figure 3.3, It can be seen that the VB-based algorithm outperforms the standard BP algorithm and EP-based algorithm. The BP algorithm and EP-based algorithm achieve the same performance. The largest gap between the VB and the standard BP is more than 10 dB.

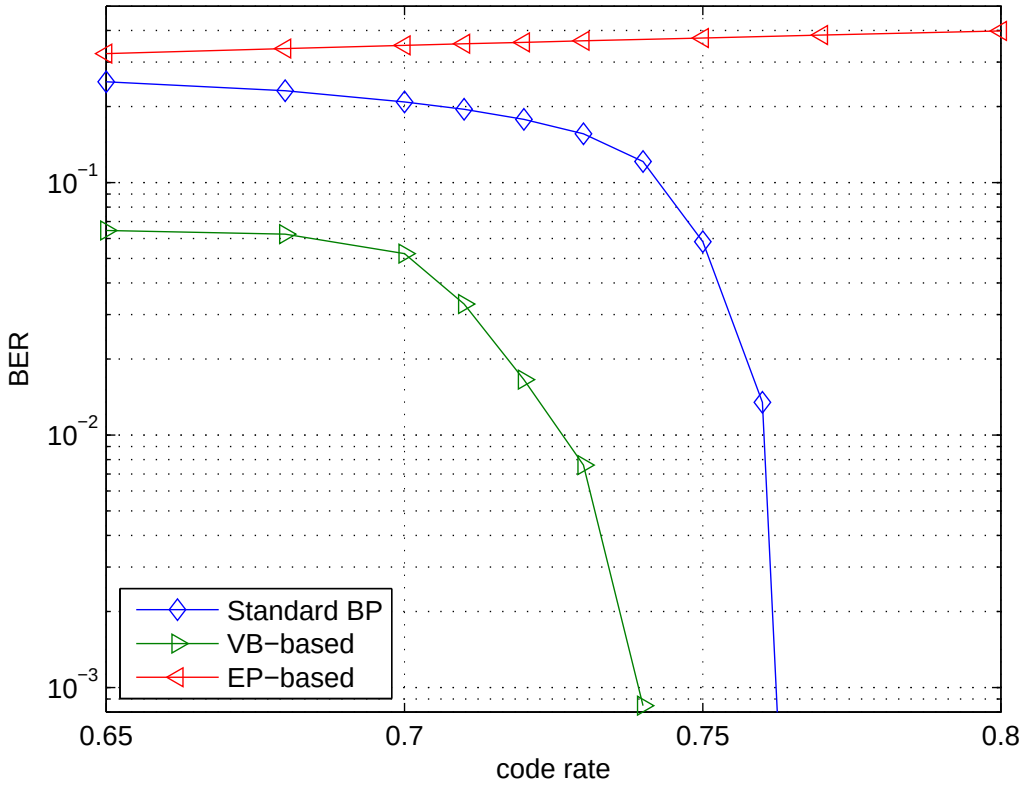


Figure 3.4: Bit error rates versus code rates curves of Exp II

Second, σ_0 and σ_1 are set as Gaussian distributions $\mathcal{N}(0.2, 0.01^2)$ and $\mathcal{N}(0.1, 0.01^2)$, respectively. Other parameters are the same as Exp I. In addition, standard BP is also implemented for comparison. The bit error rate versus bit rate is plotted in Figure 3.4. In Figure 3.4, it can be seen that the VB-based algorithm still achieves the best performance among all of the three algorithms. The EP algorithms even can not converge.

Contrary to the results of [23], the EP algorithm has the worst perfor-

mance in above experiments, which can be explained as follow:

1. The algorithm in [23] assumes the channel type as BSC; however, none of above experiments is BSC channel. As a matter of fact, the EP algorithm has a better performance in Experiment I than Experiment II. That's because the channel type in Experiment I more approaches to BSC than that of Experimental II.

2. The property of EP algorithm are more suitable to learn the single mode model problem, however, the estimation of BAC problem is a multiple-mode estimation problem.

CHAPTER 4

COMPRESSION OF CORRELATED TEMPERATURE DATA OF SENSOR NETWORK

4.1 Introduction

In the real world, to monitor the temperatures of different locations in a neighborhood, we usually install sensors at each location in order to obtain their readings periodically. However, due to the limited memory of each sensor, the readings should be sent quickly after they are sampled. As the frequency of the samplings increases, the channel between each sensor and the destination may not transmit the data correctly due to its limited capacity. Since the sensors are installed in the same neighborhood, there may exist spatial correlations among the sensors. In addition, for each sensor, temporal correlations may also exist between the consecutive readings. In this section, both spatial and temporal correlations are utilized to reduce the transmission data rate between each sensor and the destination. To verify the validity of the proposed algorithm, a small wireless network is established in the laboratory for testing.

4.2 Problem Formulation

A Crossbow temperature network [43] is installed in the laboratory. The network contains four sensors (senders) and one common destination (receiver). The four sensors are located in different locations and will send their readings to the destination every hour.

In the network, one of the data source is used as the side information (SI),

and the other three sources are compressed based on their correlations with the SI. This type of coding is called "asymmetric coding". Under this framework, this four-source DSC problem can be viewed as the three individual two-source DSC problems.

For simplicity, two sources, $X = [X_1, X_2, \ldots, X_L]$ and $Y = [Y_1, Y_2, \ldots, Y_L]$, are assumed in the problem, where X is the source to be compressed, Y is the SI, and each of both sources has L-length block data.

Suppose X and Y are jointly Gaussian, i.e.,

$$Y_i = X_i + \mathcal{N}(0, \sigma_s^2), \tag{4.1}$$

where $\mathcal{N}(0, \sigma_s^2)$ is a Gaussian distribution with 0 mean and σ_s^2 variance.

In addition, for each $X_i, 1 \leq i \leq L-1$, it is assumed that

$$X_{i+1} = X_i + \mathcal{N}(0, \sigma_t^2), \tag{4.2}$$

where $\mathcal{N}(0, \sigma_t^2)$ is a Gaussian distribution with 0 mean and σ_t^2 variance.

Data of the source X are first quantized into bits (its binary representation), and then the bits are encoded by an LDPC matrix (see section 3.3) in order to obtain the syndromes. These syndromes are then sent to the destination. Data from source Y (SI) are sent to the destination directly.

Once it receives the syndromes and the SI, the decoder will implement the message passing algorithm in the proposed factor graph to recover the data from the source X. In the next section, a detailed explanation of the proposed factor graph and how to implement the message algorithm in it will be given.

4.3 Proposed Method

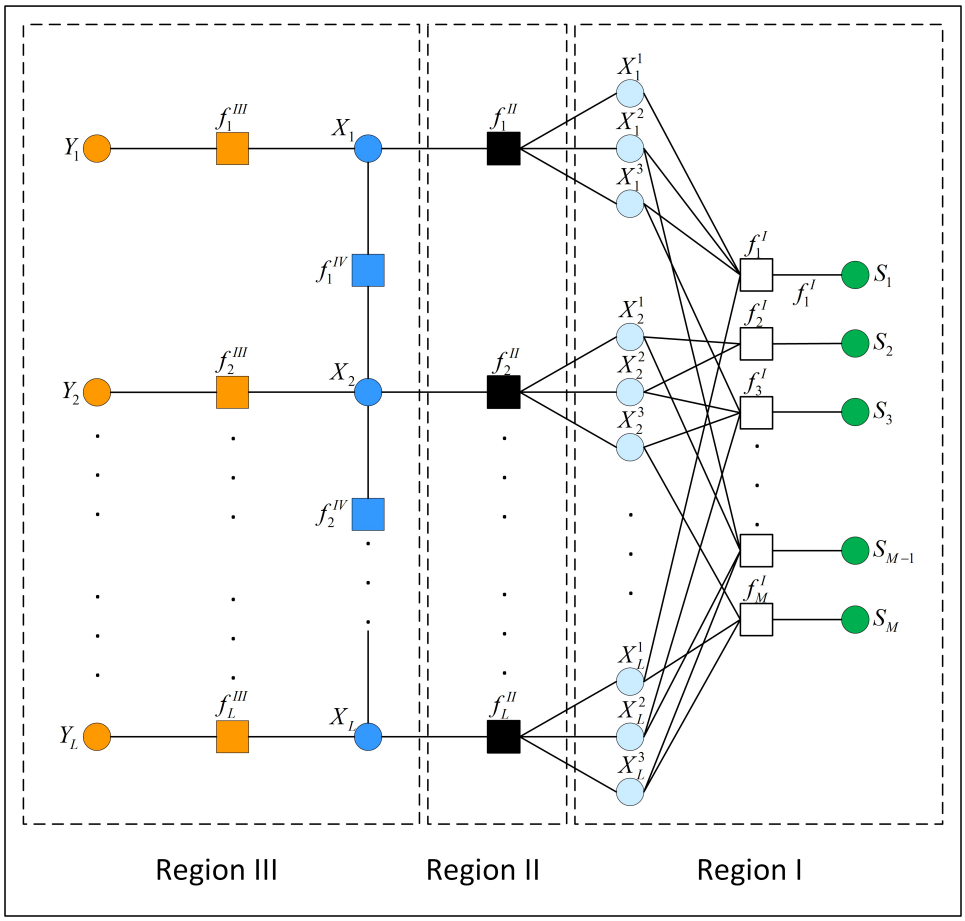


Figure 4.1: Factor Graph of Our Proposed Method

Figure 4.1 is the factor graph of the proposed algorithm. This factor graph captures all of the aforementioned correlations and constraints. Based on the types of correlations and constraints, the factor graph can be divided into three regions. The rest of this chapter will elaborate on them.

4.3.1 Region I

Region I captures the constraints of the LDPC algorithm. Suppose each x_i $(1 \leq i \leq L)$ are quantized into K bits (see Figure 4.1, where $K = 3$), and $x_{i,k}$ $(1 \leq i \leq L, 1 \leq j \leq K)$ denotes the $k-th$ bit of x_i. And all of the $x_{i,k}$ are

encoded together by the LDPC algorithm (see Section 3.3.1). Suppose the LDPC compression ratio is M/L. Then M syndromes $\mathbf{s} = \{s_1, s_2, \ldots, s_M\}$ are generated for transmission to the decoder. At the joint decoder, with the constraint imposed by the received syndromes, the factor f_m^I can be expressed as

$$f_m^I(\mathbf{X}_m, s_m) = \begin{cases} 1 & , s_m \oplus (\bigoplus \mathbf{X}_m) \\ 0 & , otherwise \end{cases}, \tag{4.3}$$

where $\mathbf{X}_m$ denotes the set of neighbors of the factor node f_m^I, $\oplus$ is the bit-xor operation, and $\bigoplus$ is bit-sum of the $\mathbf{X}_m$ set.

All of the messages in Region I can be updated according to the algorithm listed in Table 3.1.

4.3.2 Region II

In Region II, we use f_l^{II} to denote the relation between the source x_i and its quantized bits $\{x_{i1}, x_{i2}, \ldots, x_{iL}\}$. Noticeably, all the variables on the left of f_l^{II} are continuous, and due to the assumption of Gaussian distribution among them, the messages left to f_l^{II} are also Gaussian distributions. Since Gaussian distributions can be fully specified by their mean and variance, only two parameters are required to express these messages. Let $Q[X]$ denote one of the candidates of the quantized source $\{x_{i1}, x_{i2}, \ldots, x_{iL}\}$. The factor node f_l^{II} can then be given as

$$f_l^{II}(Q[X], \mu_l, \sigma_l^2) = \int_{P(Q[X])-\Delta}^{P(Q[X])+\Delta} \frac{1}{\sqrt{2\pi}\,\sigma_l} \exp\left(-\frac{(z-\mu_l)^2}{2\sigma_l^2}\right) dz, \tag{4.4}$$

where μ_l,σ_l^2 are the parameters of the message $m_{X_l \to f_l^{II}}(\mu_l, \sigma_l^2)$, $P(Q[X])$ is the de-quantized value of $Q[X]$, and Δ is half of the step size of the

quantization.

For message $m_{f_l^{II}\to X_{lk}}(x_{lk})$, $k = 1, 2, \ldots, K$, since x_{lk} is a binary variable, the BP algorithm in (2.18) can be implemented directly to update this type of messages. However, for message $m_{f_l^{II}\to X_l}(\mu_l, \sigma_l^2)$, there are 2^K items, which make it difficult or even infeasible for further implementation of message passing in Region III. To cope this difficulties, we can use a deterministic approximation method-EP algorithm, and the details will be given in the following section.

4.3.3 Region III

According to (4.1) and (4.2), the functions f_l^{III} and f_l^{IV} in Region III can be given as

$$f_l^{III}(x_l) = \frac{1}{\sqrt{2\pi}\,\sigma_s} \exp\left(\frac{-(x_l - y_l)^2}{2\sigma_s^2}\right) \tag{4.5}$$

and

$$f_l^{IV}(x_l, x_{l+1}) = \frac{1}{\sqrt{2\pi}\,\sigma_t} \exp\left(\frac{-(x_{l+1} - x_l)^2}{2\sigma_t^2}\right) \tag{4.6}$$

respectively.

For each of the variable nodes X_l, there are three types of messages that flow into this variable, and they are from factors f_l^{II}, f_l^{III} and f_l^{IV}. The message from the f_l^{II} contains 2^K items. The message from f_l^{III} can be viewed as the prior, which has a Gaussian form. The exact forms of the messages from f_l^{IV} are also Gaussian distributions. To continue implementing the message passing algorithm in region III, the EP algorithm in section 2.3.4 will be utilized to estimate all of the messages that flow out of the variable node X_l. The EP estimations of these messages are expressed as $\tilde{m}_{X_l\to f_l^{II}}(x_l)$, $\tilde{m}_{X_l\to f_l^{IV}}(x_l)$, and all of these messages are restricted into Gaussian distri-

bution. Then, the EP updating rules can be derived by using the following equations:

$$\mathcal{N}(x; m_1, \nu_1)\mathcal{N}(x; m_2, \nu_2) = \mathcal{N}(m_1; m_2, \nu_1 + \nu_1)\mathcal{N}(x; m, \nu), \tag{4.7}$$

where $v = \dfrac{\nu_1\nu_2}{\nu_1 + \nu_2}$, $m = v\left(\dfrac{m_1}{\nu_1} + \dfrac{m_2}{\nu_2}\right)$. The details of the algorithm can be found in [35].

The messages $\tilde{m}_{X_l \to f_l^{II}}(x_l)$ and $\tilde{m}_{X_l \to f_l^{IV}}(x_l)$ can then be fed into X_L's neighbor factors in order to update the messages that flows out of factors f^{II} (see Region II) and f^{IV}. The messages that flow out of the factor f_l^{IV} can be updated according to the equation

$$\int \mathcal{N}(x_1; m_1, \nu_1)\mathcal{N}(x_2; x_1, \nu_2)\ dx_1 \propto \mathcal{N}(x_2; m_1, \nu_1 + \nu_2). \tag{4.8}$$

Remark 4.1 *Actually, region III is mainly composed by a Kalman filter. The SI Y can be viewed as an observable value, and X is the hidden variable.*

In addition, when comparing with the prior work [44], there is no need to implement the procedure of de-quantization since the beliefs of the source X in Region III are continuous.

4.4 Results

In this section, the proposed algorithm will be applied to the network. For all of the 4 sensors, since the data from one sensor will be utilized as the SI, there are 3 combinations for testing. They can be called Group 1, Group 2, and Group 3. In addition, 50 readings are obtained from each sensor, i.e., the length of data block $L = 50$. The numbers of quantized bits are set as $K = 3, 4, 5, 6, 7$, and the range of quantization is $[0, 60]$. Hence, the

compression rate R can be calculated as $R = K * M/L$.

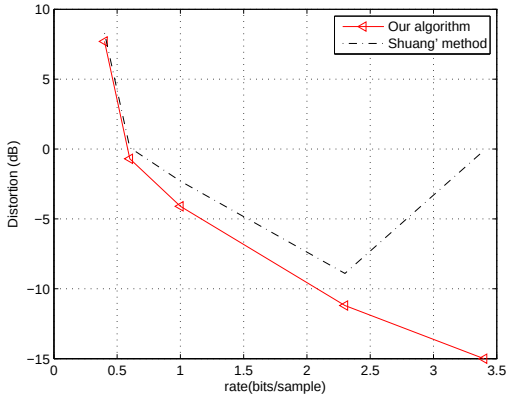
Table 4.1: The rate/bits of the three data groups given the number of bit planes (K)

	$K=3$	$K=4$	$K=5$	$K=6$	$K=7$
Group 1($\sigma_s = 0.72$,$\sigma_t = 1.1$)	0.4	0.6	1.0	2.3	3.4
Group 2($\sigma_s = 0.68$,$\sigma_t = 0.9$)	0.2	0.3	1.2	2.0	3.2
Group 3($\sigma_s = 0.65$,$\sigma_t = 1.0$)	0.2	0.4	1.2	2.3	3.5

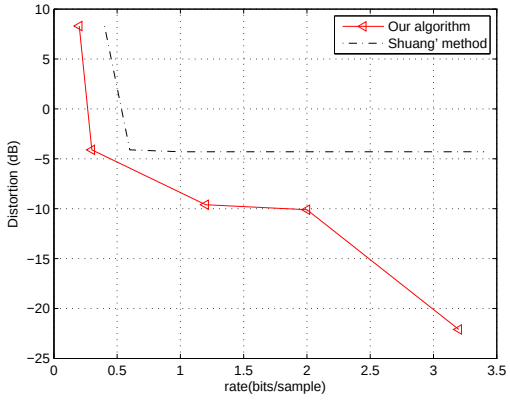
Since the message passing algorithm is an iterative algorithm, the stop criterion should be given in advance. In all of these experiments, there are two stop criteria: The first one is that all of the beliefs of the decoding bits $x_{i,k}$ successfully satisfy the LDPC syndrome constraints in (4.3). The second one is when the implementation reaches the maximum number of iterations. The maximum number of iterations is set to be 60. In addition, the EP algorithm will be implemented every 20 iterations.

For a given number of bit plane, the minimum rates R_{min} of compressed syndromes that can be successfully decoded for the proposed algorithm are calculated in Table 4.1. In addition, the spatial and temporal standard deviations of each data group are also listed.

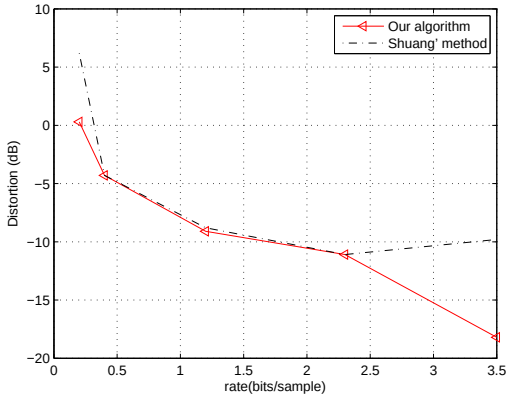
Figure 4.2 gives the distortions versus the rates listed in Table 4.1. Here, the distortion is defined as the mean square error, and it is given in decibels (dB) unit. To make a comparison, the algorithm in [44](by Wang et al.) is also implemented, and [44] only considers spatial correlations [44]. Unlike [44], the proposed algorithm assumes the correlations are priors and estimates them offline. For a fair comparison, the same assumption is also applied to [44]. From the results in Figure 4.2, the proposed algorithm outperforms the algorithm in [44]. In the Figure 4.2(a) and Figure 4.2(c),it is



(a)



(b)



(c)

Figure 4.2: Distortion versus the rate listed in Table 4.1: (a) Group 1; (b) Group 2; (c) Group 3

seen that the distortions of the algorithm in [44] increase as the rates of data increase. That's because, for a given bit plane and LDPC setting, the algorithm in [44] cannot decode successfully. However, since the correlations among the data have been fully utilized in the proposed algorithm, the probability of decoding successfully will increase.

CHAPTER 5

STREAMLINED GENOME SEQUENCE COMPRESSION USING DISTRIBUTED SOURCE CODING

5.1 Introduction

The vision of miniaturized sequencing devices is turning into reality with the emergence of MinION by Oxford Nanpore [45]. Such devices are promising in a variety of potential applications, ranging from studying of wildlife and clinical capture of sequenced genes, to food inspection for identifying pathogens. However, such portable devices are commonly subject to the constraints in processing capabilities, power budget, and storage and communication limitations. With these constraints, the traditional view of genome compression architecture as simple decoder and complex encoder needs to be changed. It is urgent to develop novel techniques to satisfy the emerging reality challenges. Data compression methods (for reducing the storage space with significantly lower computational complexity and memory requirements) become crucial for the efficient management of genomic data in portable devices.

In both situations (with or without reference sequences), traditional genome compression is computationally expensive at the encoder. The complexity is dominated by matching (approximately) repeated patterns of nucleotides—namely Adenine (A), Cytosine (C), Guanine (G) and Thymine (T)—between or within the DNA sequences. These patterns are also accompanied by insertions, deletions and substitutions of single nucleotides.

To date, a number of specialized DNA sequence compression algorithms have been proposed. In the spirit of Ziv and Lempel [46], Grumbach and

Tachi [47] proposed the first DNA sequence compressor, Biocompress, to compress the exact repeating patterns with a specially designed Fibonacci coder. The algorithm was then improved in Biocompress-2 [48] by introducing a Markov model for encoding the non-repeated regions. Chen et al. [49, 50] extended the earlier approach to cover approximated repeats by further exploiting the nature of DNA sequences. Meanwhile, the work in [51] introduced a combined CTW+LZ algorithm for searching approximate repeats and palindrome using hash and dynamic programming. Behzadi and Fessant [52] proposed a dynamic programming approach for the optimal selection of approximate repeats with promising compression efficiency being witnessed. However, such methods are heuristic as the underlying statistics of the sequence patterns are generally ignored. The authors in [53–55] proposed to combine the matching and substitution of approximate repeats and a specific normalized maximum likelihood model, obtaining a much higher compression ratio. Subsequently, statistical modeling for predicting the generation of symbols and arithmetic coding for such symbols in DNA sequences were proposed for more efficient compression. Cao *et al.* [56] proposed to estimate the probability distribution of symbols with a panel of "expert" to tackle the approximate repeat problem. Alternatively, finite context models are proposed to capture different aspects of statistical information along the sequence [57, 58], such reference free methods are plagued by their low compression rates (not greater than 6:1) and prohibitive computational consumption for large DNA sets.

Recognizing reference-free architectures do not fully utilizing information, a series of algorithms are proposed to compress sequences by matching approximate repeats with a reference sequence. The RLZ algorithm proposed

by Kuruppu *et al.* [59] performed relative Lempel-Ziv compression of DNA sequences with the collection of related sequences. Wang *et al.* [60] proposed the GRS compressor, that is able to compress a sequence with using a reference without any additional information. Applying the copy model into the matching of exact repeats in reference sequences, GReEn [61] achieved even larger gains when compared to [59] and [60]. Recently, reference-based algorithms [62, 63] achieved highly efficient compression performance for the fastq data format, by matching and comparing repeated subsequences in the reference sequences. Although the reference-based architectures can achieve hundreds of folds compression, the requirement of reference sequences makes it impractical for miniaturized devices, which have very limited storage space and communication bandwidth.

In this section, a novel and pioneering architecture for the genome compression application in miniaturized devices with limited processing capabilities, power budget, storage space and communication bandwidth is proposed. The contribution of the proposed method is three-fold.

1. To the best of our knowledge, the proposed architecture is the first practical one to meet the demands of miniaturized devices. Motivated by the distributed source coding (DSC) for sensor networks [64], the proposed scheme includes a simplified encoder without having access to reference sequences or communicating with other encoders, and a complex decoder that detects repeated subsequences in the stored reference sequences and decompress the received encoded bits with the specifically designed graphical model. Hence, the proposed compression system can successfully meet the constraints and requirements of the miniaturized sequencing devices.

2. A flexible encoding and decoding mechanism is proposed. Using feedback from the decoder, the encoder transmits either hashes conducting the detection of variable-size exact repeats in decoder or syndromes obtained with low-complexity Slepian-Wolf coding [65] of the non-repeated subsequences. The proposed encoder and decoder perform efficiently by taking consideration of both exact repeats and approximately repeated subsequences (e.g. insertion, deletion and substitution).

3. With the syndrome and reference sequences at the decoder [65], a novel factor graph model is constructed to tackle the challenge in detecting insertion, deletion and substitution between the reference and original source. Experimental results show that the proposed architecture can achieve an efficient compression performance with significantly low encoding complexity when compared to the benchmark compressor GRS.

The rest of this section is organized as follows. In Section 5.2, the proposed DSC based genome compression system is introduced, and the system includes the implementation details of the proposed hash based (exactly) repeated sequence coding with adaptive length and an overview of syndrome based non-repeated sequence coding. Then, in Section 5.3, a design of the syndromes based non-repeated sequence coding is proposed, and this design can handle the insertion, deletion, and substitution between sources and reference. The experimental results can be found in Sections 5.4.

5.2 System architecture

The block diagram of the proposed Genome compression framework is depicted in Figure 5.1. Suppose that there are two correlated DNA sequences

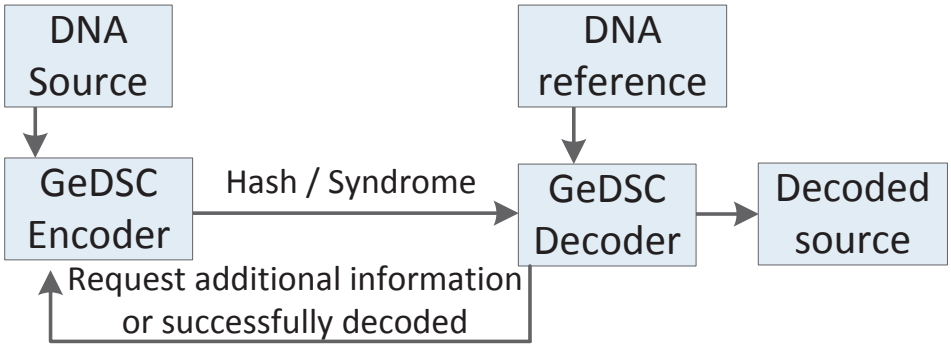


Figure 5.1: Workflow of genome compression based on DSC.

(i.e., source and reference sequences) available at the encoder and decoder, respectively, where the variations between two sequences are modeled by insertion, deletion and substitution. The alphabet of our studied DNA sequence is confined within the set $\{`A', `C', `G', `T', `N'\}$, where 'N' denotes an unknown base due to a low sequencing quality. Figure 5.2 shows the logical flow of the proposed framework.

At the encoder (see the left hand side of Figure 5.2), a streaming DNA sequence obtained from the portable sequencer will be first stored in the incoming data buffer for further processing. Second, a sub-sequence $\mathbf{x}_i^L$, which starts with the i-th to be compressed base in the source sequence, is extracted from the incoming data buffer, where its length L and the corresponding coding method are decided by the adaptive code length and types selection module. The compressed sequence can be either LDPC Accumulate (LDPCA[1]) syndromes $\mathbf{s}_{\mathbf{x}_i^L} = \mathbf{H}\mathbf{x}_i^L$ or hash bits $\mathbf{h}_{\mathbf{x}_i^L}$ depending on whether variations are presented between the source and the reference sequence, based on the decoder feedback, where $\mathbf{H}$ is the parity check matrix in LDPC codes (see section 3.3.1). Third, the encoded sequence will be temporally stored in

[1]LDPCA is an extension of LDPC in section 3.3.1, which has a feedback with the decoder so that the encoder can change the LDPC length of syndrome adaptively.

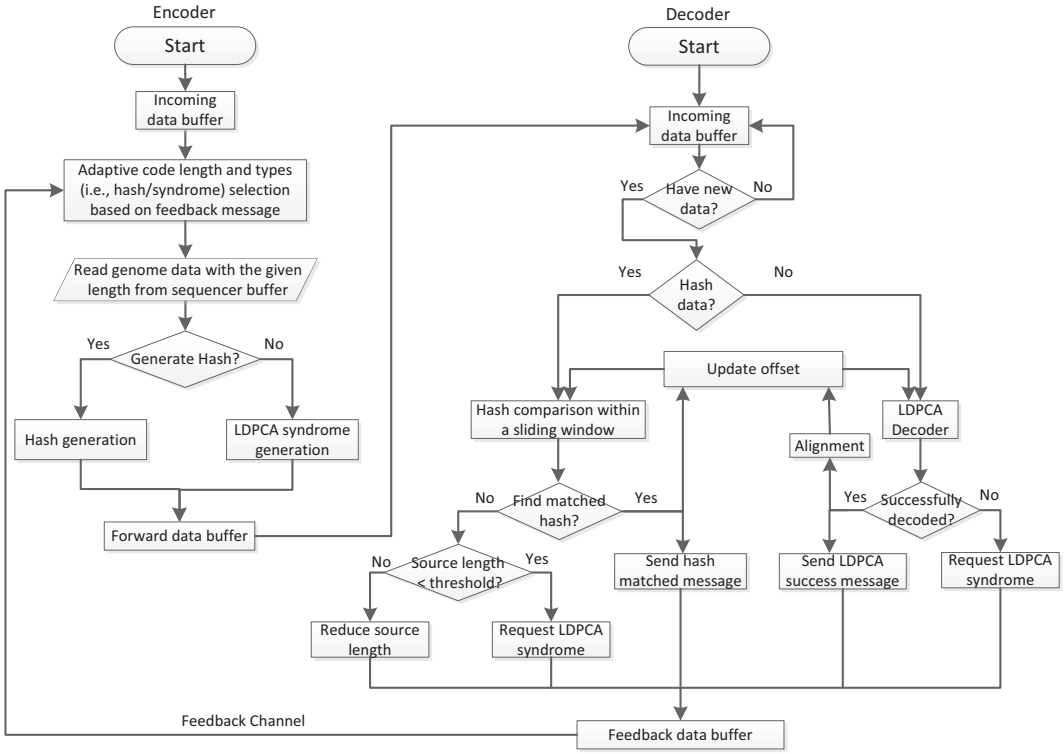


Figure 5.2: Workflow of genome compression based on DSC.

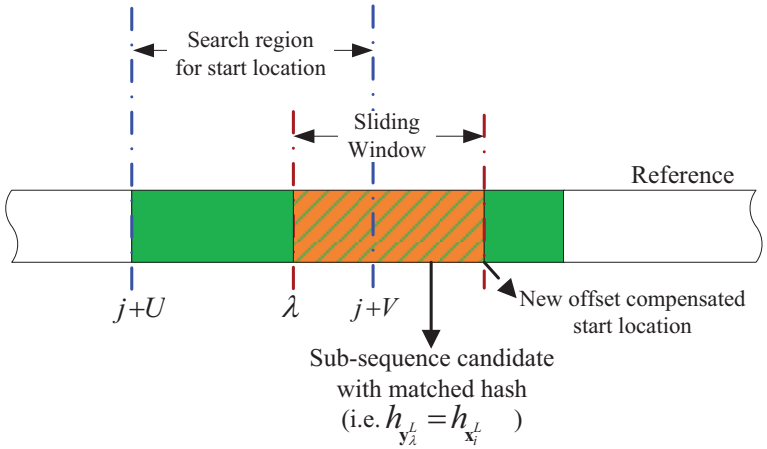


Figure 5.3: The diagram of hash-based coding

the forward data buffer and send to the decoder.

At the decoder (see the right hand side of Fig. 5.2), the received streaming data in the incoming data buffer will be processed by one of the following

modules based on the corresponding data compression mode (i.e., either hash bits or syndromes).

1. For the received hash data $\mathbf{h}_{\mathbf{x}_i^L}$, it will be compared with the hashes generated from a bunch of sub-sequence candidates $\mathbf{y}_{j+U}^L, \cdots, \mathbf{y}_{j+V}^L$ within the reference sequence for $V - U + 1$ total candidates, where j is the current offset compensated start location, and U and V are predefined lower and upper bounds of the search region for start locations. Then, the comparison result can be further processed as follows.

   (a) If a matched hash $\mathbf{h}_{\mathbf{y}_k^L}$ for $k = j + U, \cdots, j + V$ is detected (i.e., $\mathbf{h}_{\mathbf{y}_k^L} = \mathbf{h}_{\mathbf{x}_i^L}$), the next offset compensated start location of the sliding window can be updated as $j = k + L$ (see Fig. 5.3). Moreover, we claim that $\mathbf{y}_k^L$ will be identical to $\mathbf{x}_i^L$, if $\mathbf{h}_{\mathbf{y}_k^L}$ and $\mathbf{h}_{\mathbf{x}_i^L}$ are matched with each other, which is the fundamental assumption of our proposed system. Intuitively, the aforementioned assumption can be enforced by choosing a strong hash code with a small search region. The experimental results based on sequences [66, 67] with total more than 238 million bases demonstrate that a 16-bit cyclic redundancy check (CRC) hash code with a search region $U = -2$ and $V = 10$ provides a strong assertion of such assumption. In addition, the decoder will inform the success to the encoder and request a longer code length based on a predefined protocol as updating $L^{\text{current}} = bL_0$, where L_0 is a predefined initial length and the scaling factor b is updated as $b = b + d_b$, d_b is an incremental constant, and b is initialized as 0. For example, at the beginning, $L^{\text{current}} = L_0$, if a matched hash is detected, the adaptive length L^{current} will be updated as $L^{\text{current}} = d_b L_0$, as the scaling fac-

tor $b = 0 + d_b$. Similarly, if n_h number of successively matched hashes are detected, the adaptive length and its corresponding scale factor will be $L^{\text{current}} = n_h d_b L_0$ and $b = n_h d_b$, respectively.

(b) If no matched hash can be detected, the following two conditions will be checked.

i. if $L^{\text{current}} = L_0$, the decoder will inform the hash matching failure to the encoder and request syndromes from the encoder for further action.

ii. Otherwise, the decoder also informs the hash matching failure to the encoder, but requests a shorter code length by setting $L^{\text{current}} = L_0$.

2. For the received syndromes, the decoder will pass the syndrome to the proposed factor graph based LDPCA decoder with the capability of handling deletion, insertion and substitution between the source and the reference (see the next section for more implementation details). The following two conditions will be checked.

(a) If the decoded source $\hat{\mathbf{x}}_i^L$ satisfies both the parity check constraint (i.e., $\mathbf{s}_{\mathbf{x}_i} = H\hat{\mathbf{x}}_i^L$) and the hash constraint (i.e., $\mathbf{h}_{\mathbf{x}_i^L} = \mathbf{h}_{\hat{\mathbf{x}}_i^L}$), the decoder will send an LDPCA success message back to the encoder and update the offset compensated start location j through the Smith-Waterman local alignment between the reference and the decoded source. Moreover, the encoder will send hash codes to the decoder for the next sub-sequence.

(b) Otherwise, the decoder will request additional LDPCA syndromes from the encoder.

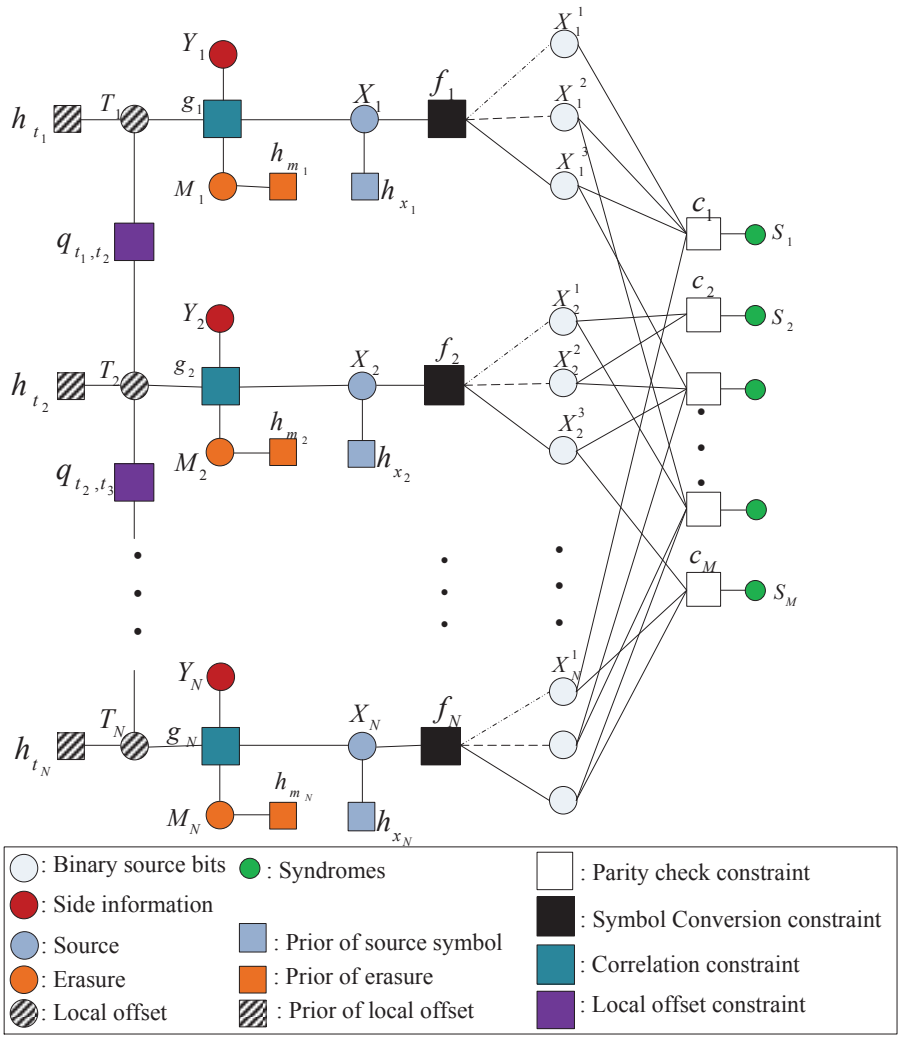


Figure 5.4: Factor graph of genome compression based on DSC.

5.3 Syndrome Based non-repeated sequence coding

As previously mentioned in our system architecture, if an exact repeat cannot be identified by hash coding, the decoder will request syndromes from the encoder through a feedback channel. In this section, the codec design of the proposed syndrome based non-repeated sequence coding is proposed.

5.3.1 Syndrome based non-repeated sequence encoding

The first step of the proposed syndrome based non-repeat encoder is to convert DNA data into a binary source, such that they can be compressed under a binary LDPCA encoder. Suppose the following mapping rule for the letters within the alphabet, i.e., 'A' $\rightarrow$ 000, 'C' $\rightarrow$ 001, 'G' $\rightarrow$ 010, 'T' $\rightarrow$ 011, 'N' $\rightarrow$ 100, a DNA subsequence $\mathbf{x}$ can be represented by the corresponding binary vector $\mathbf{x_b}$. For instance, given a DNA subsequence $\mathbf{x} = [\text{'A''T''G''C''T''N'}]^T$ with length $N = 6$, its corresponding binary vector will be $\mathbf{x_b} = [\mathbf{000\ 011\ 010\ 001\ 011\ 100}]^{\mathbf{T}}$ with length $3N$. Thus, for LDPC based Slepian-Wolf (SW) coding (i.e., lossless DSC), the compressed syndromes will be generated through $\mathbf{s_x} = \mathbf{H}\mathbf{x_b}$, where $\mathbf{H}$ is a sparse parity check matrix with size $M \times 3N$ and $M < 3N$. Thus, the resulting code rate can be expressed as $R = M/N$ bits per base. It is worth mentioning that the computational complexity of the aforementioned encoder is ultra-low, since the only operation is the bit-wise multiplication between the sparse matrix $\mathbf{H}$ and the original source. Moreover, LDPCA codes to implement rate adaptive decoding is employed, where the decoder can incrementally request additional LDPCA syndromes from the encoder through a feedback channel, when facing decoding errors.

5.3.2 Syndrome based non-repeated sequence decoding

To perform syndrome based decoding for non-repeat DNA subsequence $\mathbf{x}$ with the reference sequence as side information $\mathbf{y}$, the key factor is to be able to explore the variations between the source subsequence $\mathbf{x}$ and the reference sequence $\mathbf{y}$, where the variations are modeled by the insertion, deletion, and substitution between the source and reference. Moreover, a substitution can

be expressed as an insertion in the source sequence followed by a deletion in the corresponding location in the reference sequence. In this section, it is demonstrated that such variations can be effectively estimated through Bayesian inference on graphical models. The graphical model of the proposed syndrome based decoding with variation is depicted in Figure 5.4. In Figure 5.4, the variable nodes (usually depicted by a circle) denote variables such as source symbol, binary source bits, local offset introduced by variation, and syndromes. The detail of the proposed factor graph will be explained in the followings.

The parity check constraint imposed by the received syndromes is studied, where $s_1, \cdots, s_M$, the realization of variable node S_l, $l = 1, \cdots, M$, denotes the received syndromes in Figure 5.4. Similar as the LDPC codes in Section 3.3, the factor nodes c_l, $l = 1, \cdots, M$, take into account the parity check constraints, where the corresponding factor function can be expressed as

$$c_l(\mathbf{x}_{c_l}, s_l) = \begin{cases} 1, & \text{if } s_l \oplus \bigoplus \mathbf{x}_{c_l} = 0 \\ 0, & \text{otherwise} \end{cases}, \tag{5.1}$$

where $\mathbf{x}_{c_l}$ denotes the set of neighbors of the factor node c_l, and $\bigoplus \mathbf{x}_{c_l}$ denotes the binary sum of all elements of the set $\mathbf{x}_{c_l}$.

Moreover, x_i^1, x_i^2, x_i^3, the realization of variable node X_i^r with $i = 1, \cdots, N$, $r = 1, 2, 3$, are the binary representation for the i-th base x_i in the DNA sequence according to the mapping rule introduced in Section 5.3.1, where the mapping rule is captured by the factor node f_i, $i = 1, \cdots, N$ with corresponding factor function as follows

$$f_i(x_i, x_i^1, x_i^2, x_i^3) = \begin{cases} 1, & \text{if } \mathbf{map}(x_i^1, x_i^2, x_i^3) = x_i, \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

where $\mathbf{map}(\cdot)$ denotes the mapping from the binary bits "$\cdot$ "to a letter in the alphabet, e.g. the output of $\mathbf{map}(0, 1, 1)$ corresponds to the letter "T".

Moreover, since the alphabet is not uniformly distributed in an arbitrary DNA sequence, the prior distribution for the alphabet is captured by the factor node h_{x_i}, where learning prior through training DNA sequences will be discussed shortly in the results section.

Now, an additional erasure variable node M_i to capture the variation between reference y_i and source x_i is introduced, where the variable $m_i = 1$ indicates the presence of variations, $m_i = 0$ means the existence of matches $y_{i+t_i} = x_i$, where $t_i = -T, \cdots, T$ are all possible local offsets within the search region $[-T, T]$. Moreover, the corresponding prior distribution of variable m_i is captured by the factor node h_{m_i} with factor function defined as

$$h_{m_i}(m_i) = \begin{cases} 1 - p_e & \text{if exist matches } y_{i+t_i} = x_i \\ p_e, & \text{otherwise} \end{cases}, \tag{5.3}$$

where p_e can be learnt through training DNA sequence.

For the existence of matches $y_{i+t_i} = x_i$, the local offset t_i is captured by the variable node T_i and its corresponding prior is represented by the factor node h_{t_i} with $h_{t_i}(t_i) = p_{t_i}$, where p_{t_i} can be learnt through training DNA sequences. Furthermore, as the local offsets between adjacent DNA bases do not vary significantly in this work, it is expected that adjacent variables t_i will not differ much in value. Such characteristic is captured by the additional

factor node $q_{t_i,t_{i+1}}$, where the corresponding factor function is defined as

$$q_{t_i,t_{i+1}}(t_i, t_{i+1}, \alpha) = \frac{\alpha}{2} e^{-\alpha|t_i - t_{i+1}|}, \tag{5.4}$$

where α is the scale parameter of the Laplace distribution.

The factor node g_i and its corresponding factor function $g_i(m_i, y_i, x_i, t_i)$ are introduced to combine the impact imposed by the side information y_i, erasure m_i and local offset t_i. For $m_i = 0$, the factor function can be expressed as,

$$g_i(m_i = 0, y_i, x_i, t_i) = \begin{cases} 1 & \text{if } y_{i+t_i} = x_i, \\ 0, & \text{otherwise.} \end{cases} \tag{5.5}$$

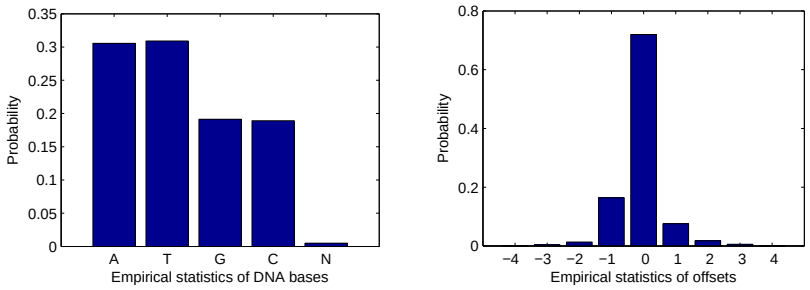
For $m_i = 1$, the variable nodes T_i and Y_i will be disconnected from the factor node g_i . Therefore, the simplified factor function $g_i(m_i = 1, x_i) = 1$ can be used to take the impact of erasure into account.

With the factor graph defined above in Figure 5.4, BP algorithm in Section 2.3.2 can be implemented so that obtain the posterior distribution (belief) of each variable. The original DNA sequence can be recovered by the posterior distribution of each source x_i.

5.4 Results

Two genome sequence data - The Arabidopsis Information Resrouce (TAIR) [27] and The Institute for Genomic Research (TIGR) [28] are adopted for testing in this experiment. These two database are collected by professional groups or institutes, and have been widely used by research communities.

The TAIR maintains a database of genetic and molecular biology data for the model higher plant Arabidopsis thaliana [27]. In this experiment, TAIR8 [66] dataset and TAIR9 [67] dataset are tested, where each dataset contains five chromosomes with over 238 millions bases in total. Moreover, the genome of TAIR9 is used for testing the compression performance with TAIR8 as reference only available at the decoder. For this experiment, all the hyper-parameters are initialized as follows, the initial code length $L_0 = 528$, the incremental constant $b_d = 3$, the scale parameter of Laplace distribution $\alpha = 1$, the maximum local offset search region $T = 4$ and the erasure probability $p_e = 0.01$. The proposed codec is implemented in MATLAB and evaluated on an Intel 3.0GHz CPU machine.

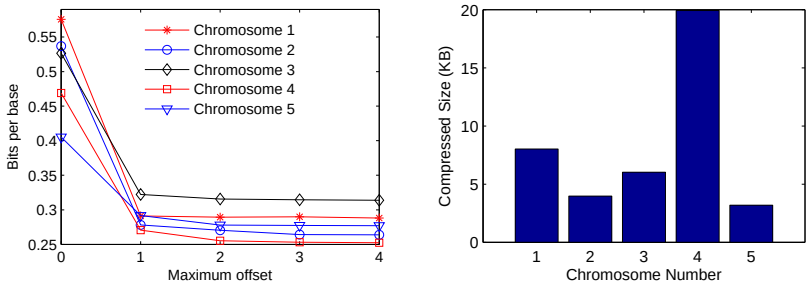


(a) (b)

Figure 5.5: The empirical statistics of (a) the DNA bases {'A', 'T', 'G', 'C', 'N'} and these of (b) the local offsets with the range from -4 to 4.

First, the empirical marginal statistics of the DNA bases {'A', 'T', 'G', 'C', 'N'} and these of the local offsets t_i within the range from -4 to 4 are shown in Figures 5.5(a) and 5.5(b), respectively, which will be used as the priors in the syndrome based non-repeated sequence decoding. In Figure 5.5(a), it is verified the assumption that the alphabets of DNA sequences are usually non-uniformly distributed. Moreover, Figure 5.5(b) depicts that the maximum local offset with $T = 4$ is sufficiently large for capturing shifts between the

reference and the source.



(a) (b)

Figure 5.6: Compression performance of the proposed codec on TAIR dataset, (a) the average code rates vs. the different maximum local offsets in syndrome coding; (b) the overall compression performance (i.e., hash bits + syndromes) for all 5 chromosomes.

Figure 5.6(a) illustrates the relationship between the average code rates and the different maximum local offsets in syndrome coding based on all 5 chromosomes. In Figure 5.6(a), it shows that the code rates decrease as the maximum local offsets increase, due to the fact that a larger maximum local offset offers a wider search region for exploring the reference. However, a larger maximum local offset may also result in a higher decoding complexity. Figure 5.6(b) shows the overall compression performance (i.e., hash bits + syndromes) for all 5 chromosomes in terms of compressed file size. Moreover, Figure 5.7 shows a side-by-side comparison of the compression rate and compression time. It shows that both the proposed method and GRS algorithm achieve significant file size reductions (i.e., up to 8252x file size reduction).

For the TIGR data, the chromosome 4 (35.8MB) of the TIGR5 dataset is tested using the chromosome 4 of the TIGR6 as the reference by varying the LDPC code length (i.e., 528, 1056, 1584, 2112 and 2640) as shown in Table 5.1. Moreover, GRS method is also implemented on this dataset, and the result was also listed in Table 5.2 as reference. It can be seen that the

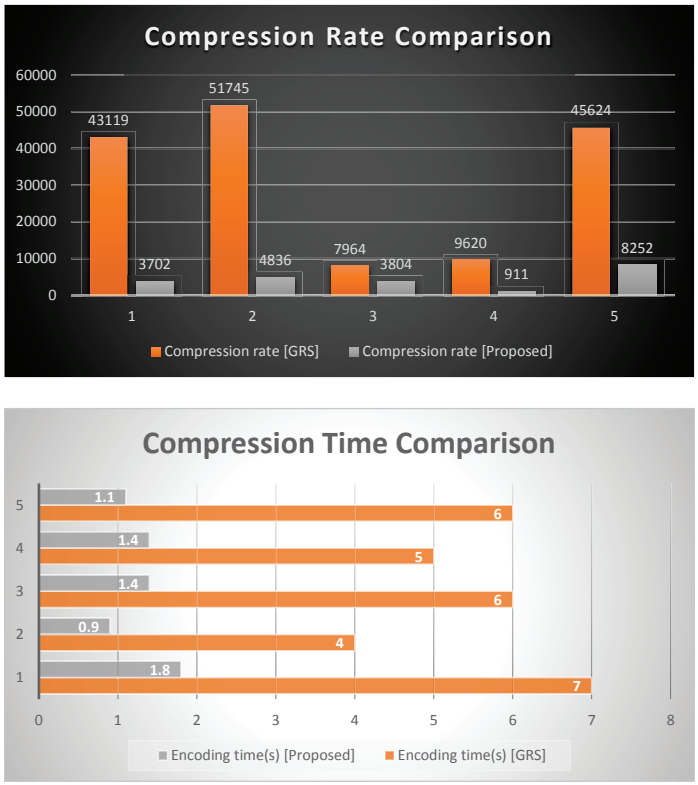


Figure 5.7: Performance comparison between GRS and our proposed codec on TAIR dataset.

compression performance decreases as the LDPC code length increases. The proposed algorithm achieved better compression performance comparing with GRS in this case. It is because that the reference chromosome 4 in TIGR6 includes a significant amount of insertions when comparing with the same chromosome in TIGR 5, where the insertion information in the reference chromosome has no contribution to the size of DSC compressed data.

The proposed encoder shows a significantly lower encoding complexity. It is worth mentioning that the proposed codec is implemented by MATLAB, where a potential performance boost is highly expected by using more efficient programming languages e.g., C/C++. To the best of my knowledge, this is the first study of DSC based genome compression. There is no doubt

Table 5.1: Performances of our proposed method on Chromosome 4 of TIGR5 (35.8 MB)

LDPC Length	Compression Size (KB)	Enc Time (s)	Dec Time (s)
528	3.68	0.04	298
1056	4.58	0.009	596
1584	5.67	0.01	787
2112	6.97	0.01	1102
2640	6.72	0.08	1374

Table 5.2: Performance of GRS on Chromosome 4 of TIGR5 (35.8 MB)

Compression Size (KB)	Encoding Time (Seconds)	Decoding Time (Seconds)
26.34	12	6

that it opens many possibilities for the portable miniaturized applications in which energy consumption and bandwidth usage are of paramount importance.

CHAPTER 6

CONCLUSIONS

DSC has attracted the interests of many researchers in recent years. The S-W and W-Z problems are two of the most important topics with regard to DSC. Although the theoretical compression limits of these two problems have been known for decades, a systematic method for solving both problems is still unavailable. Therefore, more and more researchers have focus on and tried to solve these problems. This dissertation has attempted to solve three main problems: adaptive Slepian-Wolf decoding for two binary sources (ASWDTBS) problem, the compression of correlated temperature data of sensor network (CCTDSN) problem and the streamlined genome sequence compression using distributed source coding (SGSCUDSC) problem.

ASWDTBS problem is an S-W problem. LDPC code was used to encode the source data. For the decoding, a three-region factor graph was established. The three regions are the LDPC region, the source data and SI correlation region, and the parameters estimation region. For the LDPC and correlation regions, the BP algorithm was applied to implement the message passing algorithm. The channel parameters in Region III were estimated by EP and VB algorithms. The proposed algorithm was tested with the synthetic data and compare with the standard BP algorithm with no parameter estimation. The experimental results show that the VB-based algorithm performed the best, but that the EP-based algorithm performed the worse.

CCTDSN can be viewed as an extension of the W-Z problem. A Distributed Source Coding algorithm, taking both spatial and temporal correlations into account, is proposed. To test this method, a wireless temperature

networks was established in our laboratory, and each sensor sent its temperature readings every hour. The results showed that the algorithm achieved much better results than the algorithm of prior work on the temperature network. In the future, more research will be done on the quantization step of this W-Z problem, and an optimization between the number of bit planes and each bit plane's compression ratio for a given rate will be sought.

Finally, a DSC based genome compression architecture was presented. This area of research can be classified as a S-W problem. To the best of our knowledge, the proposed framework is the first study of its kind. It is especially targeted at the low complexity genome encoding for miniaturized devices, which have limited processing capabilities, power budgets, storage space and communication bandwidth. Compared to the traditional reference based DNA compression algorithm (e.g., GRS), the proposed framework offers ultra-low encoding complexity (non-repeated subsequences are encoded using low complexity DSC encoding), while (exactly) repeated subsequences are compressed through adaptive length hash coding based on the decoder feedback. The customized factor graph based decoder tackles the challenges of detecting insertion, deletion and substitution between the reference and the original source, and it recovers the non-repeated subsequences based on received syndromes. Last but not least, our proposed genome compression framework incorporates LDPCA codes for rate adaptive decoding. Experimental results show that the proposed architecture could achieve an efficient compression performance with significantly lower encoding complexity when compared to the benchmark compressor GRS.

BIBLIOGRAPHY

[1] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inform. Theory*, vol. 19, pp. 471–480, July 1973.

[2] A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Trans. Inform. Theory*, vol. 22, pp. 1–10, Jan. 1976.

[3] A. D. Wyner, "The rate-distortion function for source coding with side information at the decoder-II: General sources," *Information and Control*, vol. 38, pp. 60–80, 1978.

[4] T. Cover and J. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[5] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (discus): design and construction," in *Proc. DCC*, pp. 158–167, 1999.

[6] A. Wyner, "Recent results in the Shannon theory," *IEEE Trans. Inform. Theory*, vol. 20, pp. 2–10, Jan. 1974.

[7] M. Grangetto, E. Magli, and G. Olmo, "Distributed arithmetic coding," *IEEE Communications Letters*, vol. 11, no. 11, p. 883, 2007.

[8] A. Aaron and B. Girod, "Compression with side information using turbo codes," in *Proc. DCC'02*, pp. 252–261, Snowbird, UT, Mar. 2002.

[9] A. Aaron, S. Rane, R. Zhang, and B. Girod, "Wyner-Ziv coding of video: applications to compression and error resilience," in *Proc. DCC'03*, (Snowbird, UT), Apr. 2003.

[10] D. Rebollo-Monedero and R. Zhang and B. Girod, "Design of optimal quantizers for distributed source coding," in *Proc. DCC'03*, (Snowbird, UT), Mar 2003.

[11] C. Tang, N. Cheung, A. Ortega, and C. Raghavendra, "Efficient inter-band prediction and wavelet based compression for hyperspectral imagery: a distributed source coding approach," in *Proc. DCC'05*, (Snowbird, UT), pp. 437–446, Mar 2005.

[12] D. Varodayan, A. Aaron, and B. Girod, "Rate-adaptive distributed source coding using low-density parity-check codes," in *43rd Asilomar Conference on Signals, Systems and Computers, 2005. Conference Record of the*, 2005.

[13] Q. Xu and Z. Xiong, "Layered Wyner–Ziv video coding," *Image Processing, IEEE Transactions on*, vol. 15, no. 12, pp. 3791–3803, 2006.

[14] D. Kubasov, J. Nayak, and C. Guillemot, "Optimal reconstruction in wyner-ziv video coding with multiple side information," in *Multimedia Signal Processing, 2007. MMSP 2007. IEEE 9th Workshop on*, pp. 183–186, IEEE, 2007.

[15] C. Brites and F. Pereira, "Correlation noise modeling for efficient pixel and transform domain wyner–ziv video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 9, pp. 1177–1190, 2008.

[16] S. Wang, L. Cui, S. Cheng, Y. Zhai, M. Yeary, and Q. Wu, "Noise Adaptive LDPC Decoding Using Particle Filtering," *Communications, IEEE Transactions on*, vol. 59, no. 4, pp. 913–916, 2011.

[17] D. Krithivasan and S. S. Pradhan, "Distributed source coding using abelian group codes: A new achievable rate-distortion region," *Information Theory, IEEE Transactions on*, vol. 57, no. 3, pp. 1495–1519, 2011.

[18] L. Cui, S. Wang, S. Cheng, and M. Yeary, "Adaptive Binary Slepian-Wolf Decoding using Particle Based Belief Propagation," *Communications, IEEE Transactions on*, vol. 59, no. 9, pp. 2337–2342, 2011.

[19] S. Wang, L. Cui, L. Stankovic, V. Stankovic, and S. Cheng, "Adaptive correlation estimation with particle filtering for distributed video coding." *IEEE Transactions on CSVT*, to appear.

[20] R. Ma and S. Cheng, "The universality of generalized hamming code for multiple sources," *IEEE Transactions on Communications*, pp. 1–7, Oct. 2011.

[21] X. Pan, R. Liu, and X. Lv, "Low-complexity compression method for hyperspectral images based on distributed source coding," *Geoscience and Remote Sensing Letters, IEEE*, vol. 9, no. 2, pp. 224–227, 2012.

[22] J. E. Barceló-Lladó, A. M. Pérez, and G. Seco-Granados, "Enhanced correlation estimators for distributed source coding in large wireless sensor networks," *Sensors Journal, IEEE*, vol. 12, no. 9, pp. 2799–2806, 2012.

[23] L. Cui, S. Wang, and S. Cheng, "Adaptive slepian-wolf decoding based on expectation propagation," *Communications Letters, IEEE*, vol. 16, no. 2, pp. 252–255, 2012.

[24] R. Ma and S. Cheng, "Zero-error slepian-wolf coding of confined-correlated sources with deviation symmetry," *Information Theory, IEEE Transactions on*, vol. 59, no. 12, pp. 8195–8209, 2013.

[25] N. Deligiannis, A. Munteanu, S. Wang, S. Cheng, and P. Schelkens, "Maximum likelihood laplacian correlation channel estimation in layered wyner-ziv coding," *Signal Processing, IEEE Transactions on*, vol. 62, no. 4, pp. 892–904, 2014.

[26] M. Vaezi and F. Labeau, "Distributed source-channel coding based on real field bch codes," 2014.

[27] "Tair," *http://www.arabidopsis.org/*.

[28] "Tigr rice genome annotation," *http://www.arabidopsis.org/*.

[29] F. Chen, M. Rutkowski, C. Fenner, R. C. Huck, S. Wang, and S. Cheng, "Compression of distributed correlated temperature data in sensor networks," in *Data Compression Conference (DCC), 2013*, pp. 479–479, IEEE, 2013.

[30] S. Wang, X. Jiang, F. Chen, L. Cui, and S. Cheng, "Streamlined genome sequence compression using distributed source coding," *Cancer informatics*, vol. 13, no. Suppl 1, p. 123, 2014.

[31] J. M. Bernardo, "Reference Posterior Distributions for Bayesian-Inference," *Journal of the Royal Statistical Society Series B-Methodological*, vol. 41, no. 2, pp. 113–147, 1979.

[32] C. P. Robert, *The Bayesian choice: a decision-theoretic motivation*. Springer-Verlag, 1994.

[33] C. Bishop, *Pattern recognition and machine learning*, vol. 4. Springer New York, 2006.

[34] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[35] T. P. Minka, "Expectation propagation for approximate bayesian inference," in *Uncertainty in Artificial Intelligence*, vol. 17, pp. 362–369, 2001.

[36] D. Schonberg, K. Ramchandran, and S. S. Pradhan, "Distributed code constructions for the entire Slepian-Wolf rate region for arbitrarily correlated sources," in *Data Compression Conference, 2004. Proceedings. DCC 2004*, pp. 292–301, 2004.

[37] S. Pradhan and K. Ramchandran, "Generalized coset codes for distributed binning," *Information Theory, IEEE Transactions on*, vol. 51, no. 10, pp. 3457–3474, 2005.

[38] Y. Fang, "Crossover probability estimation using mean-intrinsic-llr of ldpc syndrome," *IEEE Communications Letters*, vol. 13, no. 9, pp. 679–681, 2009.

[39] A. Zia, J. Reilly, and S. Shirani, "Distributed parameter estimation with side information: A factor graph approach," in *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pp. 2556–2560, IEEE, 2007.

[40] V. Toto-Zarasoa, A. Roumy, and C. Guillemot, "Maximum likelihood bsc parameter estimation for the slepian-wolf problem," *Communications Letters, IEEE*, vol. 15, no. 2, pp. 232–234, 2011.

[41] L. Cui, S. Wang, and S. Cheng, "Online snr statistic estimation for ldpc decoding over awgn channel using laplace propagation," in *Global Telecommunications Conference (GLOBECOM 2012), 2012 IEEE*, IEEE, 2012.

[42] S. Wang, L. Cui, and S. Cheng, "Noise adaptive ldpc decoding using expectation propagation," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pp. 1–5, IEEE, 2011.

[43] "Crossbow sensors," *http://www.moog-crossbow.com/*.

[44] S. Wang, L. Cui, S. Cheng, L. Stankovic, and V. Stankovic, "Onboard low-complexity compression of solar images," in *Proceedings of IEEE ICIP*, (Brussels, Belgium), pp. 2641–2644, Sept. 2011.

[45] "MinION: A miniaturised sensing instrument," Feb. 2012.

[46] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inform. Theory*, vol. 24, pp. 530–536, Sept. 1978.

[47] S. Grumbach and F. Tahi, "Compression of DNA sequences," in *Proc. Data Compression Conf.*, (Snowbird, Utah, USA), pp. 340–350, Mar. 1993.

[48] S. Grumbach and F. Tahi, "A new challenge for compression algorithms: Genetic sequences," *J. Inf. Process. Manage.*, vol. 30, pp. 876–887, Nov. 1994.

[49] X. Chen, S. Kwong, and M. Li, "A compression algorithm for DNA sequences," *IEEE Eng. Med. Biol. Mag.*, vol. 20, pp. 61–66, July 2001.

[50] X. Chen, M. Li, B. Ma, and J. Tromp, "DNACompress: fast and effective DNA sequence compression," *Bioinformatics*, vol. 18, pp. 1696–1698, Dec. 2002.

[51] T. Matsumoto, K. Sadakane, and H. Imai, "Biological sequence compression algorithms," *Genome Informatics*, vol. 11, pp. 43–52, Dec. 2000.

[52] B. Behzadi and F. L. Fessant, "DNA compression challenge revisited: A dynamic programming approach," *Combinatorial Pattern Matching*, vol. 3537, pp. 85–96, June 2005.

[53] I. Tabus, G. Korodi, and J. Rissanen, "DNA sequence compression using the normalized maximum likelihood model for discrete regression," in *Proc. Data Compression Conf.*, (Snowbird, Utah, USA), pp. 253–262, Mar. 2003.

[54] G. Korodi, I. Tabus, J. Rissane, and J. Astola, "DNA sequence compression - Based on the normalized maximum likelihood model," *IEEE Signal Processing Mag.*, vol. 24, pp. 47–53, Jan. 2007.

[55] G. Korodi and I. Tabus, "Normalized maximum likelihood model of order-1 for the compression of DNA sequences," in *Proc. Data Compression Conf.*, (Snowbird, Utah, USA), pp. 33–42, Mar. 2007.

[56] M. D. Cao, T. I. Dix, L. Allison, and C. Mears, "A simple statistical algorithm for biological sequence compression," in *Proc. Data Compression Conf.*, (Snowbird, Utah, USA), pp. 43–52, Mar. 2007.

[57] A. J. Pinho, A. Neves, C. Bastos, and P. Ferreira, "DNA coding using finite-context models and arithmetic coding," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process*, (Taipei), pp. 1693–1696, Apr. 2009.

[58] D. Pratas and A. Pinho, "Compressing the human genome using exclusively Markov models," in *5th Int. Conf. on Practical Applications of Comput. Biol. Bioinformatics (PACBB 2011)*, pp. 213–220, Mar. 2011.

[59] S. Kuruppu, S. Puglisi, and J. Zobel, "Relative lempel-ziv compression of genomes for large-scale storage and retrieval," *String Process. and Inf. Retrieval*, vol. 6393, pp. 201–206, Oct. 2010.

[60] C. Wang and D. Zhang, "A novel compression tool for efficient storage of genome resequencing data," *Nucleic Acids Res.*, vol. 39, p. e45, Apr. 2011.

[61] A. Pinho, D. Pratas, and S. Garcia, "GReEn: A tool for efficient compression of genome resequencing data," *Nucleic Acids Res.*, vol. 40, p. e27, Feb. 2012.

[62] C. Kozanitis, C. Saunders, S. Kruglyak, V. Bafna, and G. Varghese, "Compressing genomic sequence fragments using SlimGene," *J. Comput. Biol.*, vol. 18, pp. 401–413, Mar. 2011.

[63] M. H. Y. Fritz, R. Leinonen, G. Cochrane, and E. Birney, "Efficient storage of high throughput DNA sequencing data using reference-based compression," *Genome Res.*, vol. 21, pp. 734–740, May 2011.

[64] Z. Xiong, A. D. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Processing Mag.*, vol. 21, pp. 80–94, Sept. 2004.

[65] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): Design and construction," *IEEE Trans. Inform. Theory*, vol. 49, pp. 626–643, Mar. 2003.

[66] E. Huala, A. Dickerman, M. Garcia-Hernandez, D. Weems, L. Reiser, F. LaFond, D. Hanley, D. Kiphart, M. Zhuang, W. Huang, *et al.*, "The arabidopsis information resource (tair): a comprehensive database and web-based information retrieval, analysis, and visualization system for a model plant," *Nucleic acids research*, vol. 29, no. 1, pp. 102–105, 2001.

[67] S. Rhee, W. Beavis, T. Berardini, G. Chen, D. Dixon, A. Doyle, M. Garcia-Hernandez, E. Huala, G. Lander, M. Montoya, *et al.*, "The arabidopsis information resource (tair): a model organism database providing a centralized, curated gateway to arabidopsis biology, research materials and community," *Nucleic acids research*, vol. 31, no. 1, pp. 224–228, 2003.