Master's thesis at Chalmers University of Technology, Dept. of Signals and Systems

On Source Coding With Fountain Codes

Report No. EX068/2008

Pooyan Abouzar Djirandehi

CHALMERS

Supervised by
Prof. Amin Shokrollahi
Laboratoire d'algorithmique
Laboratoire des mathematiques algorithmiques
School of Computer and Communication Sciences
Swiss Federal Institute of Technology (EPFL)
and
Prof. Erik Agrell
Department of Signals and Systems
Chalmers University of Technology

October 20, 2008

Abstract

Source and channel coding using fountain codes(also known as rateless erasure codes) have been of much interest during the last years. The capacity achieving property of these codes along with the low decoding complexity have greatly contributed to these codes becoming more frequently applied in both compression and protection against channel disruptions. One of the major problems of compression and source coding is the *Slepian-Wolf* problem. The problem gained importance because of suggesting that, no matter of seperate or jointly encoding of the sources X and Y, the sufficient rate for reconstructing them remains the same. The report is mostly devoted to fountain code approach of *Slepian-Wolf* problem. Suitable raptor code design helps us to achieve any arbitrary point of *Slepian-Wolf* region. Two different points of the *Slepian-Wolf* region are achieved with being only 3.8% and 5.4% off the *Slepian-Wolf* limit respectively. This fountain code approach is expected to outperform the classical code approach such as Low-Density-Parity-Check(LDPC) codes in terms of decoding complexity and proximity to desired *Slepian-Wolf* point. In a later part of the thesis, the Belief Propagation (BP) is applied to *LT-Markov* sub-graphs. Gilber-Eliot (GE) channel is the assumed Markov channel in this work. We come up with the overhead of 16.7% while applying the former Binary Symmetric Channel(BSC) design to *LT-Markov* sub-graphs.

Acknowledgements

There are some individuals I would like to thank and appreciate their cooperation and influence at different stages of the thesis.

First of all, I would like to thank Professor Shokrollahi for his clear and concise discussions on different parts of my work and especially appreciate his hints and ideas on Section 7 which turns out to be the most novel part of my work.

I would also like to thank Bertrand Ndzana Ndzana for his consistent and regular supervision and particularly appreciate his ideas on the opening three sections and the $8th$ section of this report.

I thank Martin Isaksson and Shirin Saeedi Bidokhti for proofreading some important parts of the report which appeared to find typos and flaws I would certainly have missed out.

I greatly thank Amin Mazloumian who helped me with Java programming.

I would really like to thank Mahdi Cheraghchi for his interruptions in the lab and his remarkable patience listening to the questions which were not in his area! and would also thank Mohammad Mahmoody for the football and movie debates!

Many thanks also go to my family for their support.

Contents

List of Figures

1 Introduction

Source and channel coding with Fountain codes does not have emerged for more than a decade. *Slepian-Wolf* problem has always been of great interest in the meanwhile. Interesting point about *Slepian-Wolf* is that, different correlated sources can be encoded seperately at rates equal or larger than their conditional entropy and decode them jointly to come up with intact source messages. The requirement is that their entropies add up to the joint entropy of all sources. In this work and other works, authors are mostly concerned with case of two correlated sources. However it is hypothetically expandable to multiple source case. *Slepian-Wolf* problem has been already solved by block codes such as LDPC codes [11]. Although in this work, the *Slepian-Wolf* problem is addressed using fountain codes. The advantage of fountain codes over block codes is their remarkably lower decoding complexity. Besides, we expect to get closer to the *Slepian-Wolf* limit. In this work, we first explain and discuss some important concepts of source and channel coding during the opening three sections which leads to a better perception of the upcoming chapters. The general scheme of *Slepian-Wolf* simultaneous decoding for memoryless sources has been brought up and explained in Section 5 and the raptor code design of section 7 has been implemented and simulated finally. The latter tested design, relies on first decoding the XOR of two sources and then retrieving one of them to finally have the two sources recovered. This approach saves us from horrible calculations and design of the general scheme depicted in Section 5 which would have required nonlinear programming. A brief explanation along with channel capacity calculation details of *GE* channels which are a particular kind of Markov channels have been introduced in Section 6. Section 8 is mostly devoted to Density Evolution (DE) and Sum-Product-Algorithm (SPA) on *GE-LDPC* subgraphs and then it has been generalized to *GE-LT* subgraphs. No particular design has been done on *GE-LT* subgraphs; instead we have tested the design of *Slepian-Wolf* corner point achieving problem on Binary Symmetric Channels (BSC) of the same capacity and have made a comparison of the amount of required overhead to reach the minimum Bit Error Rate (BER).

2 Data Compression

The majority of the first 3 chapters have been extracted and taken from [1] and [2]. The purpose of data compression is to remove the redundancy of data as much as possible. Although different methods have been devised to meet this end, almost all follow the same philosophy. That is to assign longer description sequences to less frequent input symbols and shorter descriptions to more frequent symbols. In fact, a source code $\mathcal{C}$ for a random variable X and realization x is a mapping from x to $D^\star$, which is a set of finite length strings of symbols from a D-ary alphabet. D denotes the number of alphabets of the codebook. Huffman code is known and proved to be the most efficient compressing code. In the following we would briefly look at the key definitions and proof of important theorems that will come to much use in our further research.

2.1 Basic Definitions and Concepts

2.1.1 Non-Singular codes

A non-singular code is a code which maps each entry of the range of X to a different sequence in D*. Non-singularity only grants the successful coding of the source, but does not certainly yield a decodable code.

2.1.2 Extension of the code $\mathcal{C}$

Extension of the code $\mathcal{C}$ is nothing but concatenating the strings of the range $\mathcal{C}$ sequences assigned to finite length strings of X.

$$\mathcal{C}(x_1x_2x_3\dots x_n) = \mathcal{C}(x_1)\mathcal{C}(x_2)\dots\mathcal{C}(x_n). \tag{2.1}$$

2.1.3 Uniquely decodable codes

If extension of a code is non-singular, the code is then uniquely decodable. In other words, the extension of a uniquely decodable code would be decoded to only one individual string of input source alphabets.

2.1.4 Prefix codes

A prefix code or instantaneous code is a codebook in which, none of the codewords are prefix to one another. Such a code is instantaneously decodable which means that each specific codeword can be decoded at the same time one reaches the end of it. Huffman code is the most popular prefix code.

2.1.5 Optimal codes

Optimality of a codeword is usually measured with the average length of it. In other words, a codebook $\mathcal{C}$ is supposed to be optimal if it has less average length comparing to other codes. Let l_i and p_i be the length and probability of the ith codeword respectively, the average codeword length,L, is then defined by

$$L = \sum_{i=1}^{m} p_i l_i. \tag{2.2}$$

2.1.6 D-adic probability distributions

A probability distribution which is D-adic with respect to D, is assigned the equal probability of D^{-n} to all of its codewords.

2.2 Kraft Inequality

Kraft Inequality is a useful tool for a swift check of whether a specific code could be a prefix code or not. One can not conclude that a code for which the Kraft Inequality holds is definitely a prefix code, but it is possible to construct a prefix code using codewords lengths of that code. A code for which the Kraft Inequality does not hold, not only is a non-prefix code but even also not a uniquely decodable code. The inequality is as following:
for any instantaneous code over an alphabet of size D, the codeword lengths $l_1, l_2, ..., l_m$ should satisfy (2.3).

$$\sum_{i=1}^{m} D^{-l_i} \leq 1. \tag{2.3}$$

2.2.1 Extended Kraft Inequality

This inequality mentions that for countably infinite prefix codewords, the codeword lengths satisfy the extended Kraft Inequality.

$$\sum_{i=1}^{\infty} D^{-l_i} \leq 1. \tag{2.4}$$

2.3 Optimal codelength

It is a quite interesting fact that the shortest description length of a code lies within one bit of the source alphabet's entropy and would never drop below the source entropy.

$$H(X) \leq L < H(X) + 1. \tag{2.5}$$

Since (2.5) plays a paramount role in data compression, it is worth including the proof in this report. Let p_i denote the transmission probability of the ith codeword and l_i denote the length of the ith codeword. According to optimality which was defined earlier, we would always wish to minimize the code average length $L = \sum_{i=1}^{m} p_i l_i$ and we also know that integers $l_1, l_2, \ldots, l_m$ satisfy the Kraft Inequality because a non-uniquely-decodable code is of no practical use. Firstly we prove a lemma and then proceed with the proof of the main theorem.

Lemma 2.1 *The expected length of a prefix D-ary code for a random variable X is greater than or equal to the entropy $H_D(X)$.*

$$L \geq H_D(X). \tag{2.6}$$

Equality holds only for $p_i = D^{-l_i}$.

Proof:

$$\begin{aligned} L - H_D(X) = \sum_i p_i l_i - \sum_i p_i \log_D \left(\frac{1}{p_i}\right) = \\ -\sum_i p_i \log_D \left(D^{-l_i}\right) + \sum_i p_i \log_D (p_i). \end{aligned} \tag{2.7}$$

Introducing two other variables $r_i = D^{-l_i} / \sum_{j=1}^{m} D^{-l_j}$ and $c = \sum_{i=1}^{m} D^{-l_i}$ leads to

$$
\begin{aligned}
L - H = \sum_i p_i \log_D \left(\frac{p_i}{r_i}\right) - \log_D (c) = \\
D(\mathbf{p}\|\mathbf{r}) + \log_D \left(\frac{1}{c}\right) \geq 0.
\end{aligned} \tag{2.8}
$$

■

The latter inequality follows from non-negativity of relative entropy and also from the Kraft Inequality that make both of the logarithms positive. Now we turn our attention back to the desired main theorem.

As proved earlier only $l_i = \left[\log_D \left(\frac{1}{p_i}\right)\right]$ would yield $L = H$ but it is not granted for $\left[\log_D \left(\frac{1}{p_i}\right)\right]$ to be integer for all values of i (l_i can only take on integer values). In case $\left[\log_D \left(\frac{1}{p_i}\right)\right]$ is not an integer we have to inevitably round it upward to $\left\lceil\log_D \left(\frac{1}{p_i}\right)\right\rceil$ in order to ensure that l_i is an integer. These uprounded codewords would also satisfy the Kraft Inequality.

$$
\sum_{i=1}^{m} D^{-\left\lceil \log\left(\frac{1}{p_i}\right)\right\rceil} \leq \sum_{i=1}^{m} D^{-\log\left(\frac{1}{p_i}\right)} = \sum_{i=1}^{m} p_i = 1. \tag{2.9}
$$

According to the above discussion, the choice of codeword lengths would hold in (2.10).

$$
\log_D \left(\frac{1}{p_i}\right) \leq l_i < \log_D \left(\frac{1}{p_i}\right) + 1. \tag{2.10}
$$

Just multiplying all the terms of the above inequality by p_i and summing over i, would yield the desired result,

$$
H_D(X) \leq L < H_D(X) + 1. \tag{2.11}
$$

Now, we will continue another interesting corollary of the above theorem.

Corollary 2.2 *Using large block lengths of input symbols gets us closer to the lower bound $H_D(X)$.*

The intuitive reason is that encoding one symbol gives us at most 1 bit of overhead since $\log\left(\frac{1}{p_i}\right)$ is not an integer. Whereas encoding a larger block length at once would distribute this overhead among the symbols of the block. In other words if we define $l(x_1, x_2, ..., x_n)$ to be the length of the codeword assigned to $(x_1, x_2, ..., x_n)$, then the expected length of a codeword per symbol L_n would be

$$L_n = \frac{1}{n}\sum p(x_1 x_2, \ldots, x_n) l(x_1, x_2, \ldots, x_n) = \frac{1}{n} E\left[l(X_1, X_2, \ldots, X_n)\right]. \tag{2.12}$$

According to (2.5), (2.13) will be achieved.

$$H(X_1, X_2, ..., X_n) \leq E\left[l(X_1, X_2, \ldots, X_n)\right] < H(X_1, X_2, \ldots, X_n) + 1. \tag{2.13}$$

Assuming that $X_1, X_2, ..., X_n$ are drawn i.i.d from the source alphabet's distribution along with (2.12) and (2.13) leads to (2.14).

$$H(X) \leq L_n < H(X) + \frac{1}{n}\cdot \tag{2.14}$$

The inequality (2.14) indicates that by merging input symbols and taking them as new symbols to be encoded, the average codeword length per symbol approaches the entropy. As $n \to \infty$,

$$\frac{1}{n} H(X_1, X_2, ..., X_n) \xrightarrow[n \Rightarrow \infty]{} H(X). \tag{2.15}$$

If we cast aside the assumption of $X_1, X_2, ..., X_n$ being i.i.d, then

$$\frac{H(X_1, X_2, ..., X_n)}{n} \leq L_n < \frac{H(X_1, X_2, ..., X_n)}{n} + \frac{1}{n}\cdot \tag{2.16}$$

Moreover we can say that if $X_1, X_2, ..., X_n$ form a stationary stochastic process, the average length per symbol tends to $H(X)$ as n increases.

$$L_n^\star \xrightarrow[n \Rightarrow \infty]{} H(X). \tag{2.17}$$

■

Another useful theorem whose proof is not given here, expresses the amount of penalty we pay if an inaccurate distribution of symbols is used instead. Let $p(x)$ and $q(x)$ be the correct and estimated probability distribution of input symbols respectively and $l(x) = \left\lceil \log\left(\frac{1}{q(x)}\right) \right\rceil$. Then, we have the following important equation,

$$H(p) + D(p\|q) \leq E_p\left[l(X)\right] < H(p) + D(p\|q) + 1. \tag{2.18}$$

$D(p\|q)$ bits is the penalty paid due to inaccuracy in input symbols distribution.

3 Channel Capacity And Channel Coding

In the previous part, source coding was briefly explained. In this section, we focus on channel coding. There is a close duality between source and channel coding. In source coding, the aim is to remove redundancy of the data as much as possible, whereas in channel coding redundancy is added to input data to make it robust against physical channel and probable data bits corruptions.

3.1 Information channel capacity and some examples

Channel, in communications, refers to the medium used to convey information from a sender to a receiver.

Definition 3.1 *Let X represent the space of signals that can be transmitted, and Y the space of signals received, during a block of time over the channel. Information channel capacity of a discrete memoryless channel is the maximum mutual information of X and Y over the distribution $p(x)$.*

$$C = \max_{p(x)} I(X;Y). \tag{3.19}$$

Here, the maximum is taken over all distributions $p(x)$ on the input alphabet. Channel capacity would be expressed as the maximum number of bits that can be sent at each use of the channel with an arbitrarily small error probability. For a noiseless binary channel for instance, the sent bit would be received intact at the other side of the channel. Hence every bit could be received exactly as it was sent, therefore the capacity would be simply 1. The noteworthy point on channel capacity calculation is that, the output distribution highly depends on that of the input and thus can not take on any arbitrary value. In other words, it is not allowed to assign the maximum value of $H(Y)$ in order to get the maximum value of $I(X;Y)$ achieved. Now we will continue with some examples which would be useful at better perception of the concept.

3.1.1 Binary Symmetric Channel

A binary symmetric channel is a channel in which, either of the transmitted input symbols are likely to be mistaken for the other one with equal probability p. We can calculate the capacity as (3.20).

$$\begin{aligned} I(X;Y) = H(Y) - H(Y \| X) = \\ H(Y) - \sum p(x) H(Y \| X = x) = \\ H(Y) - \sum p(x) H(p) = \\ H(Y) - H(p) \leq \\ 1 - H(p). \end{aligned} \tag{3.20}$$

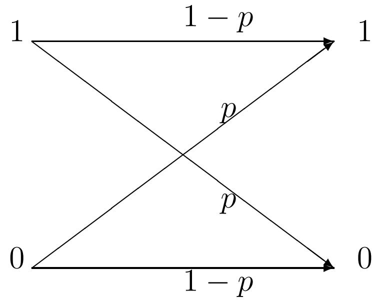

Figure 3.1: *The probability transition diagram of the binary symmetric channel.*

3.1.2 Binary Erasure Channel

The binary erasure channel is a channel in which, the transmitted bit would either be received intact at the receiver or gets corrupted and lost with probabilities $1 - \alpha$ or α respectively. Let X be a binary random variable With $Pr(X = 1) = \pi$, the capacity could be calculated as

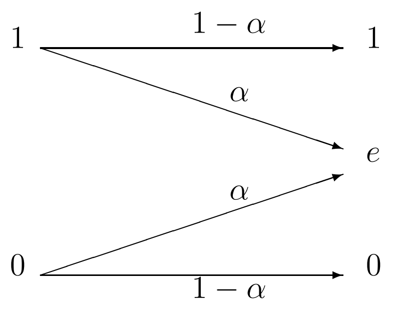

Figure 3.2: *The probability transition diagram of the binary erasure channel.*

$$C = \max_{p(x)} H(Y) - H(Y|X) = \max_{p(x)} H(Y) - H(\alpha). \tag{3.21}$$

H(Y) could be calculated like in (3.22).

$$\begin{aligned} H(Y) = Pr(Y=0)\log\frac{1}{Pr(Y=0)} \\ +Pr(Y=e)\log\frac{1}{Pr(Y=e)} \\ +Pr(Y=1)\log\frac{1}{Pr(Y=1)}. \end{aligned} \tag{3.22}$$

Then we have $Pr(Y=0)=(1-\pi)(1-\alpha)$, $Pr(Y=1)=\pi(1-\alpha)$ whereas $Pr(Y=e)=\alpha$, substituting these values in $H(Y)$ and then applying $H(Y)$ in (3.21) would yield

$$C = \max\,(1-\alpha)H(\pi) = 1-\alpha. \tag{3.23}$$

3.1.3 Symmetric Channels

A symmetric channel is identified with a symmetric $P(y|x)$ matrix. The element at the ith row and jth column of the matrix represents the conditional probability of having y_j received given x_i be the desired transmitted symbol. Sum of the row elements of the matrix is obviously one, but one can not deduce that the column sum is also one (we only need to take a look at the binary erasure channel). For a symmetric channel,

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) \\ &= H(Y) - H(\mathbf{r}) \\ &\leq \log |y| - H(\mathbf{r}), \end{aligned} \tag{3.24}$$

in which $\mathbf{r}$ is a row of a transition matrix.

3.1.4 Weakly Symmetric Channels

Despite the symmetric channels in which both the rows and columns are permutable with each other, weakly symmetric channel is a channel whose transition matrix is solely row wise permutable. An example of a weakly symmetric channel is shown in (3.25).

$$p(y|x) = \begin{pmatrix} 0.4 & 0.1 & 0.5 \\ 0.1 & 0.4 & 0.5 \end{pmatrix}. \tag{3.25}$$

I(X;Y) is a continuous concave function of $p(x)$ over a closed convex set and that is why we can look for a global maximum on $I(X;Y)$ versus $p(x)$. Now we have the tools required for proceeding to channel coding theorem which fundamentally underlies coding theorem.

3.2 Joint Typicality

The basic requirement for a receiver to be able to decode successfully is that no two input sequences produce an identical output sequence, y. According to joint typicality, for each n-sequence, there exist $2^{nH(Y|X)}$ possible Y sequences. What is needed for a unique successful decoding is that no two input codes lead up to the same output at the other side of the channel. There are approximately $2^{nH(Y)}$ Y sequences, thus the number of disjoint regions on Y region is approxitely $2^{n(H(Y)-H(Y|X))} = 2^{nI(X;Y)}$ which imply that nearly $2^{nI(X;Y)}$ distinguishable sequences can be sent. At this time, we can briefly detail the whole process.

At the transmitter side, a sequence message W would be drawn from index $1, 2, ..., M$ then would be encoded to a n-sequence $X^n(W)$ and transmitted over the channel which would be received as a random sequence Y^n, Then receiver decides that the index say $\hat{W} = g(Y^n)$ was sent. An error occurs in case $W \neq \hat{W}$. A memoryless channel is a channel whose outputs are independent of the inputs at previous time samples. A channel without feedback is a channel whose inputs do not depend on the past output symbols in other words,

$$p(x_k|x^{k-1}, y^{k-1}) = p(x_k|x_{k-1}). \tag{3.26}$$

Definition 3.2 *An (M, n) code for the channel $\{x, p(y|x), y\}$ consists of an index set $\{1, 2, ..., M\}$, an encoding function $X^n : \{1, 2, ..., M\} \rightarrow \mathcal{X}^n$ that gives us codewords $X^n(1), X^n(2), ..., X^n(M)$ and a decoding function*

$$g : \mathcal{Y}^n \rightarrow \{1, 2, ..., M\}. \tag{3.27}$$

which assigns a guessed index to each received vector. Any error occurrence could be expressed with

$$\begin{aligned} \lambda_i &= Pr\left[g(Y^n) \neq i | X^n = X^n(i)\right] \\ &= \sum_{y^n} p\left[y^n | x^n(i)\right] I\left[g(y^n \neq i)\right]. \end{aligned} \tag{3.28}$$

Where $I(\cdot)$ is the indicator function. (3.28) *indicates the error probability given that $X^n(i)$ was sent. The maximum error probability is therefore given by*

$$\lambda^n = \max \lambda_i, \tag{3.29}$$

and the average error probability would be then defined as

$$P_e^{(n)} = \frac{1}{M} \sum_{i=1}^{M} \lambda_i. \tag{3.30}$$

Definition 3.3 *The rate R of an (M, n) code is defined as*

$$R = \frac{\log M}{n} \textit{ bits per transmission.} \tag{3.31}$$

The rate R is said to be achievable if a $(\lceil 2^{nR} \rceil, n)$ exists such that $\lambda^{(n)}$ goes to zero as n tends to infinity.

Now, we are at the position to define joint typical sequences which will come to a great use when elaborating channel coding theorem later.

3.2.1 Joint Typical Sequences

Joint typicality is the method that is usually used to decode the received sequences because the aposteriori decoding is a demanding task to be performed at the receiver. We normally say that the received vector Y corresponds to codeword $X^n(i)$ if they are jointly typical.

Definition 3.4 *The set $A_\epsilon^{(n)}$ of jointly typical sequences $\{(x^n, y^n)\}$ with respect to the distribution $p(x, y)$ is the set of n-sequences with empirical entropies $\epsilon-$close to the true entropies.*

$$A_\epsilon^{(n)} = (x^n, y^n) \in \mathcal{X}^n \times \mathcal{Y}^n : \tag{3.32}$$

$$\left| -\frac{1}{n} \log p(x^n) - H(X) \right| < \epsilon, \tag{3.33}$$

$$\left| -\frac{1}{n} \log p(y^n) - H(Y) \right| < \epsilon, \tag{3.34}$$

$$\left| -\frac{1}{n} \log p(x^n, y^n) - H(X,Y) \right| < \epsilon, \tag{3.35}$$

and $p(x^n, y^n) = \prod_{i=1}^{n} p(x_i, y_i)$ *since the channel is memoryless.*

3.2.2 Joint AEP Theorem:

Theorem 3.5 *let (X^n, Y^n) be sequences of length n drawn i.i.d from the distribution $p(x^n, y^n) = \prod_{i=1}^{n} p(x_i, y_i)$, then following results are deducable.*

1. $Pr((X^n, Y^n) \in A_\epsilon^{(n)}) \to 1 \quad as \quad n \to \infty$
2. $|A_\epsilon^{(n)}| \leq 2^{(nH(X,Y)+\epsilon)}$
3. $(\tilde{X}^n, \tilde{Y}^n) \sim p(x^n)p(y^n)$ *then*

$$Pr((\tilde{X}^n, \tilde{Y}^n) \in A_\epsilon^{(n)}) \leq 2^{-n(I(X;Y)-3\epsilon)}, \tag{3.36}$$

and for sufficiently large n, we have

$$Pr((\tilde{X}^n, \tilde{Y},^n) \in A_\epsilon^{(n)}) \geq (1-\epsilon)2^{-n(I(X;Y)+3\epsilon)}. \tag{3.37}$$

3.3 Channel Coding Theorem

This theorem is the most basic theorem of information theory and underlies the later attempts taken in order to devise channel codes which could take us as close as possible to capacity of channel. Shannon infact set an ultimate target for the achievable rate on a physical channel. The ideas shannon introduced in order ro transmit data reliably over a channel include:

1. Allowing an arbitrarily non-zero probability of error.
2. Using the channel for so many times in row so that the law of large numbers comes true.
3. Calculating the average error probability over a large number of random codebooks to have the probability symmetrized and finally to find a good code.

It was mentioned earlier that the probability of a n-sequence codeword Y be jointly typical with a codeword X is approximately $2^{-nI(X;Y)}$. Hence, having 2^{nI} codewords almost guarantees an error-free transmission over the channel. Now we will bring up and prove the most important theorem of information theory history.

3.4 Definition and Proof Of The Channel Coding Theorem

Theorem 3.6 *All rates below capacity C are achievable. Specifically, for every rate $R < C$ there exists a sequence of $(2^{nR}, n)$ codes with maximum probability of error $\lambda^{(n)} \to 0$. Conversely, any set of $(2^{nR}, n)$ codes with $\lambda^{(n)} \to 0$ must have $R \leq C$.*

Now, we proceed to the proof of theorem. At first the achievability part will be proved and then comes the converse section.
We generate a code of $(2^{nR}, n)$ randomly drawn from distribution $p(x)$ and we also generate every alphabet of these 2^{nR} codewords randomly.

$$p(x^n) = \prod_{i=1}^{n} p(x_i). \tag{3.38}$$

Now, these codewords can be indicated within a matrix such that each row of the matrix shows one of the codewords,

$$C = \begin{bmatrix} x_1(1) & x_2(1) & \dots & x_n(1) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(2^{nR}) & x_2(2^{nR}) & \dots & x_n(2^{nR}) \end{bmatrix}. \tag{3.39}$$

Noting the discussion, each entry in the matrix is generated i.i.d according to the distribution $p(x)$. So the probability of a particular codebook could be expressed as in (3.40).

$$Pr(\mathcal{C}) = \prod_{\omega=1}^{2nR} \prod_{i=1}^{n} p(x_i(\omega)). \tag{3.40}$$

Now, $\mathcal{C}$ is assumed to have been generated as above and also revealed to both sender and receiver. Both sender and receiver are also aware of the transition matrix $p(y|x)$ for the channel. A message indexed W is then selected according to the uniform distribution,

$$Pr(W = \omega) = 2^{-nR}, \omega = 1, 2, \dots, 2^{nR}, \tag{3.41}$$

in which ω corresponds to the ωth row of the matrix C. The receiver then receives a sequence according to the channel distribution $p(y^n|x^n)$.

$$p(y^n|x^n(\omega)) = \prod_{i=1}^{n} p(y_i|x_i(\omega)), \tag{3.42}$$

and then the receiver takes a guess upon which index was sent according to the joint typicality concept. It means that the receiver announces $\hat{W}$ as the sent index if:

1. $(X^n(\hat{W}), Y^n)$ is jointly typical.
2. There is no other index k, such that $(X^n(k), Y^n) \in A_\epsilon^{(n)}$.

An error occurs in case $\hat{W} \neq W$ which could be called by the event $\mathcal{E} = \left\{\hat{W} \neq W\right\}$.
For calculating the probability of error, one has to take error probability average over all the codewords of a codebook and then over all codebooks, whereas taking average over codewords could be disregarded because of symmetry in distribution. In other words we can expect all the codewords to introduce more or less the same error probability and therefore the corresponding error probability for one of them could represent the average over all the codewords.

$$\begin{aligned} Pr(\mathcal{E}) &= \frac{1}{2^{nR}} \sum_{\omega=1}^{2^{nR}} \sum_{\mathcal{C}} P(\mathcal{C}) \lambda_{\omega}(\mathcal{C}) && (3.43) \\ &= \sum_{\mathcal{C}} P(\mathcal{C}) \lambda_1(\mathcal{C}) && (3.44) \\ &= Pr(\mathcal{E}|W=1). && (3.45) \end{aligned}$$

Now the event E_i is defined as the probability of $X^n(i)$ and Y^n to be jointly typical.

$$E_i = \{(X^n(i), Y^n) \in A_{\epsilon}^n\}, i \in \left\{1, 2, \ldots, 2^{nR}\right\}. \qquad (3.46)$$

Where E_i is obviously the event of ith codeword and Y^n being jointly typical. Hence the probability of error could be rephrased as below.

$$\begin{aligned} Pr(\mathcal{E}|W=1) &= P(E_1^c \cup E_2 \cup E_3 \cup \ldots \cup E_{2^{nR}}) && (3.47) \\ &\leq P(E_1^c) + \sum_{i=2}^{2^{nR}} P(E_i), && (3.48) \end{aligned}$$

The latter inequality follows from the fact that the events $E_1^c, E_2, \ldots, E_{2^{nR}}$ are not disjoint, hence the probability of union of them is not sum of their probabilities. Now according to the joint typical theorem AEP, the following relation holds,

$$P(E_1^c) \leq \epsilon, \qquad \text{for n sufficiently large.} \qquad (3.49)$$

According to codeword generation process, given that $X^n(1)$ is sent, Y^n and $X^n(i)$ are independent for $i \neq 1$ so are $X^n(1)$ and $X^n(i)$. Hence the probability of Y^n and $X^n(i)$ for $i \neq 1$ being jointly typical is $\leq 2^{-n(I(X;Y)-3\epsilon)}$, therefore

$$
\begin{aligned}
P(\mathcal{E}) = P(\mathcal{E}|W=1) &\leq P(E_1^c) + \sum_{i=2}^{2^{nR}} P(E_i) \\
&\leq \epsilon + \sum_{i=2}^{2^{nR}} 2^{-n(I(X;Y)-3\epsilon)} \\
&= \epsilon + (2^{nR}-1)2^{-n(I(X;Y)-3\epsilon)} \\
&\leq \epsilon + 2^{3n\epsilon}2^{-n(I(X;Y)-R)} \\
&\leq 2\epsilon.
\end{aligned}
\tag{3.50}
$$

■

The latter inequality would only be achieved if $R \leq I(X;Y) - 3\epsilon$, which implies that in case $R \leq I(X;Y)$, ϵ can be chosen such that error probability is less than 2ϵ Which tops off the proof of the achievability section. In the following section we will discuss the Fano's inequality and consequently the proof to the converse section of channel coding theorem.

3.4.1 Fano's Inequality

Theorem 3.7 *For a discrete memoryless channel with a codebook $\mathcal{C}$, uniformly distributed input messages and $P_e^{(n)} = Pr(W \neq g(Y^n))$, there exists an inequality for $H(X^n|Y^n)$.*

$$H(X^n|Y^n) \leq 1 + P_e^{(n)}nR. \tag{3.51}$$

Lemma 3.8 *Letting Y^n be the result of passing X^n through a discrete memoryless channel. Then we have*

$$I(X^n;Y^n) \leq nC \quad \textit{for all } p(x^n). \tag{3.52}$$

Proof:

$$\begin{aligned} I(X^n;Y^n) = H(Y^n) - H(Y^n|X^n) = & \quad (3.53) \\ H(Y^n) - \sum_{i=1}^{n} H(Y_i|X_i) \leq & \quad (3.54) \\ \sum_{i=1}^{n} H(Y_i) - \sum_{i=1}^{n} H(Y_i|X_i) = & \quad (3.55) \\ \sum_{i=1}^{n} I(X_i;Y_i) \leq & \quad (3.56) \\ nC. & \quad (3.57) \end{aligned}$$

■

The equality in (3.54) follows from the channel lack of memory and the inequality (3.55) comes from the fact that the codeword alphabets might be dependent on each other. Hence, $H(Y^n)$ is upperbounded by $\sum_{i=1}^{n} H(Y_i)$.

This theorem simply implies that using the channel for so many times would not increase the information capacity in bits per transmission.

Now, we are prepared to prove the converse section of the channel coding theorem which says that any sequence of $(2^{nR}, n)$ codes with $\lambda^{(n)} \to 0$ must have $R \leq C$.

As long as maximal probability goes to zero, it is easy to conclude that the average probability also tends to zero. Since, the index W has been taken from a uniform distribution over $\{1, 2, \ldots, 2^{nR}\}$, we have

$$\begin{aligned} nR = H(W) &= H(W|Y^n) + I(W;Y^n) && (3.58) \\ &\leq H(W|Y^n) + I(X^n(W);Y^n) && (3.59) \\ &\leq 1 + P_e^n nR + nC, && (3.60) \end{aligned}$$

in which, (3.59) follows from the natural fact that $W \to X^n(W) \to Y^n \to \hat{W}$ form a markov process. The inequality (3.60) follows from the Fano's inequality discussed earlier. Dividing the latter inequality by n yields (3.61).

$$R \leq P_e^{(n)} R + \frac{1}{n} + C, \tag{3.61}$$

by letting $n \to \infty$ and noting that $P_e^{(n)}$ goes to zero for large n according to the above assumptions. Thus

$$R \leq C. \tag{3.62}$$

Another deducable inequality could be

$$P_e^{(n)} \geq 1 - \frac{C}{R} - \frac{1}{nR}. \tag{3.63}$$

It is clear from the above relation that for $R > C$, the error probability bound would get away from zero. Now we will continue with some basic concepts of channel coding.

4 Encoding Of Correlated Sources

In data compression, $R \geq H(X)$ is sufficient to encode a source X. For the purpose of encoding two sources with the joint probability distribution of $p(x, y)$ together, the rate of $H(X, Y)$ would obviously suffice. Slepian and Wolf have shown that even in case of seperate encoding of the sources X and Y, total rate of $R \geq H(X, Y)$ would be sufficient to reconstruct the sources, whereas at the first sight one might think of $R = R_x + R_y \geq H(X) + H(Y)$ to be the minimum rate for that end. Letting $((2^{nR_1}, 2^{nR_2}), n)$ be the distributed source code for the joint source (X, Y) with encoder maps,

$$f_1 : \mathscr{X}^n \to \left\{1, 2, \ldots, 2^{nR_1}\right\}, f_2 : \mathscr{Y}^n \to \left\{1, 2, \ldots, 2^{nR_2}\right\}, \tag{4.64}$$

and decoder map,

$$g : \left\{1, 2, \ldots, 2^{nR_1}\right\} \times \left\{1, 2, \ldots, 2^{nR_2}\right\} \to \mathscr{X}^n \times \mathscr{Y}^n, \tag{4.65}$$

in which, $f_1(X^n)$ is the index for X^n and $f_2(Y^n)$ is the index corresponding to Y^n where (R_1, R_2) is the rate pair of the code.

Theorem 4.1 *Slepian-Wolf: For the distributed source coding problem for the source (X, Y) drawn i.i.d $\sim p(x, y)$, the achievable rate region would be then characterized by*

$$\begin{aligned} R_1 &\geq H(X|Y), \\ R_2 &\geq H(Y|X), \\ R_1 + R_2 &\geq H(X, Y). \end{aligned} \tag{4.66}$$

4.1 Achievability of the Slepian-Wolf Theorem

If for each sequence X^n, an index is drawn from $\left\{1, 2, \ldots, 2^{nR}\right\}$ uniformly at random, the set of sequences X^n with the same index would form a bin. The whole procedure can be viewed as first, setting up some bins and then throwing the sequences X^n into the bins. To decode the source, we look inside the bin to find a typical X^n sequence. In case there is only one typical sequence in the bin, it could be declared as the estimate $\hat{X}^n$ of the source sequence. Otherwise, error has occured. Generally, if the number of bins is much greater than the number of typical sequences, the likeliness of more than one typical sequences existing in a bin is small. Hence the probability that a typical sequence results in an error is very small. Letting $f(X^n)$ be the bin index corresponding to X^n and $\mathbf{g}$ be the decoding function, the probability of error is

$$\begin{aligned}
P(g(f(X)) \neq X) &\leq P(X \notin A_\epsilon^{(n)}) + \sum_x P(\exists x' \neq x : x\prime \in A_\epsilon^{(n)}, f(x\prime) = f(x))p(x) \\
&\leq \epsilon + \sum_x \sum_{x' \in A_\epsilon^{(n)}, x' \neq x} P(f(x') = f(x))p(x) \\
&\leq \epsilon + \sum_x \sum_{x' \in A_\epsilon^{(n)}} 2^{-nR} p(x) \\
&= \epsilon + \sum_{x' \in A_\epsilon^{(n)}} 2^{-nR} \sum_x p(x) \\
&\leq \epsilon + \sum_{x' \in A_\epsilon^{(n)}} 2^{-nR} \\
&\leq \epsilon + 2^{n(H(X)+\epsilon)} 2^{-nR} \\
&\leq 2\epsilon.
\end{aligned} \tag{4.67}$$

It is easy to observe that if $R > H(X) + \epsilon$ and n is sufficiently large, the probability of error could be arbitrarily small. Now we can proceed to top off the theorem's proof.

We assume to have partitioned the space of $\mathscr{X}^n$ into 2^{nR_1} bins and the space of $\mathscr{Y}^n$ into 2^{nR_2} bins. Since we are concerned with distributed encoding, we presume that sender 1 has sent the index of the bin X belongs to as m and sender 2 has sent the index of the bin Y belongs to as n. On the decoder side then, (x, y) would be declared as the sent sequence, where $f_1(x) = m$, $f_2(y) = n$. Now the probability of error could be comprised of four events, E_0, E_1, E_2 and E_{12}.

$$\begin{aligned}
&E_0 = \left\{(X, Y) \notin A_\epsilon^{(n)}\right\}, \\
&E_1 = \left\{\exists x' \neq X : f_1(x') = f_1(X) \quad and \quad (x', Y) \in A_\epsilon^{(n)}\right\}, \\
&E_2 = \left\{\exists y' \neq Y : f_2(y') = f_2(Y) \quad and \quad (X, y') \in A_\epsilon^{(n)}\right\}, \\
&E_{12} = \{\exists (x', y') : x' \neq X, y' \neq Y, f_1(x') = f_1(X), f_2(y') \neq f_2(Y) \\
&\ and \quad (x', y') \in A_\epsilon^{(n)}\}.
\end{aligned} \tag{4.68}$$

Union of events bound yields

$$\begin{aligned} P_e^{(n)} &= P(E_0 \cup E_1 \cup E_2 \cup E_{12}) \\ &\le P(E_0) + P(E_1) + P(E_2) + P(E_{12}). \end{aligned} \tag{4.69}$$

According to AEP, $P(E_0)$ tends to zero, and for $P(E_1)$ we can say that

$$\begin{aligned} P(E_1) &= P\left\{\exists x' \neq X : f_1(x') = f_1(X), \quad \textit{and} \quad (x', Y) \in A_\epsilon^{(n)}\right\} \\ &= \sum_{(x,y)} p(x,y) P\left\{\exists x' \neq x : f_1(x') = f_1(x), (x', y) \in A_\epsilon^{(n)}\right\} \\ &\le \sum_{(x,y)} p(x,y) \sum_{x' \neq x, (x',y) \in A_\epsilon^{(n)}} P(f_1(x') = f_1(x)) \\ &= \sum_{x,y} p(x,y) 2^{-nR_1} |A_\epsilon(X|y)| \\ &\le 2^{-nR_1} 2^{n(H(X|Y)+\epsilon)}. \end{aligned} \tag{4.70}$$

The latter tends to zero in case $R_1 > H(X|Y)$. For large n and $R_2 > H(Y|X)$, $R_1 + R_2 > H(X,Y)$, $P(E_2)$ and $P(E_{12})$ will also go to zero. The proof is therefore complete.

■

5 Simultaneous Decoding Of LT Codes For The Slepian-Wolf Problem

5.1 Problem Description

Two memoryless sources have been encoded seperately and oblivious of each other. Their symbols have been drawn *i.i.d* from a joint probability distribution function $p(x_i, y_i)$. We encode both sources X and Y with LT codes of output node distributions Ω_1 and Ω_2 respectively. At the decoder, simultaneous decoding is required to be done such as to operate at any desired point of the optimum Slepian-Wolf line:

$$\begin{cases} R_x + R_y = H(X,Y) \\ R_x \geq H(X|Y) \\ R_y \geq H(Y|X). \end{cases} \tag{5.71}$$

In the following, a major scheme and then Belief Propagation messages are proposed.

5.2 Scheme of simultaneous decoding for LT codes

In this section, we will briefly explain what has been done in [7]. Firstly, we state some assumptions which help to simplify the problem. Regarding the Slepian-Wolf region we intend to achieve, two decoders help each other to exploit the correlation between the two sources. Two sources are assumed to have been correlated with a BSC. It is possible to assume a $BSC(p_{est})$ between the sources X and Y, so that

$$p(Y_i = y_i | X_i = x_i) = \begin{cases} p_{est} & x_i \neq y_i \\ 1 - p_{est} & x_i = y_i \end{cases} . \tag{5.72}$$

For the marginal probability ditributions we have

$$\begin{aligned} P_{X_i}(x_i) &= \sum_{y_i} P_{X_i,Y_i}(x_i, y_i), \\ P_{Y_i}(y_i) &= \sum_{x_i} P_{X_i,Y_i}(x_i, y_i). \end{aligned} \tag{5.73}$$

Since LT codes are a class of erasure rateless codes [4], we would begin collecting the output symbols to catch up with the desired Slepian-Wolf point. With the assumption of moving on the line (5.71), we need to apply the rates $rH(X|Y)$ and $sH(Y|X)$ so that r and s be two real numbers greater than 1 satisfying

$$rH(X|Y) + sH(Y|X) = H(X,Y). \tag{5.74}$$

Let the entropy of X given Y be denoted by $H(X|Y) = h(p_{est_1})$ and $H(Y|X) = h(p_{est})$. $h(\cdot)$ is the binary entropy function defined by

$$h(p) = p\log(\frac{1}{p}) + (1-p)\log(\frac{1}{1-p}). \tag{5.75}$$

It is easy to obtain p_{est} and p_{est_1} since we are given with $p_{X,Y}(x,y)$ as the joint probability distribution,

$$\begin{aligned} p_{est} &< p \\ p_{est_1} &< p_1. \end{aligned} \tag{5.76}$$

p and p_1 are the smallest values that allow the error free decoding of sources X and Y so that

$$rh(p) + sh(p_1) > H(X,Y). \tag{5.77}$$

Now letting n be the output length of sources X and Y, we begin with collecting $nrh(p_{est})$ encoding symbols at Y decoder and $nsh(p_{est_1})$ encoding symbols at X decoder. Due to the fact that our estimations may be away from the values p and p_1 which would lead up to the correct decoding, there would probably remain some unsatisfied output node (checknode) equations at the end of the message passing decoding. Thus, p_{est} and p_{est_1} should be updated step by step to come up with yet unknown values p and p_1. Let p^i_{est} denote the ith update of p_{est} and $p^i_{est_1}$ be the ith update of p_{est_1}. Then at ith update step, $nr(h(p^i_{est_1}) - h(p^{i-1}_{est_1}))$ and $ns(h(p^i_{est}) - h(p^{i-1}_{est}))$ extra bits are sent to X and Y decoders to compensate for the information shortage. Belief Propagation (BP) algorithm is also after each update. Let's assume that $nrh(p_{est_1})(1+\epsilon_1)$ and $nsh(p_{est})(1+\epsilon_2)$ symbols have been finally collected at decoders X and Y respectively. ϵ_1 and ϵ_2 could then be the representing values for the error or the amount we are off the desired Slepian-Wolf region. It is noteworthy that the LT symbols have been produced from intermediate nodes which have been provied out of the input symbols with an invertible matix G,

$$(i_1, \ldots, i_n) = G(y_1, \ldots, y_n). \tag{5.78}$$

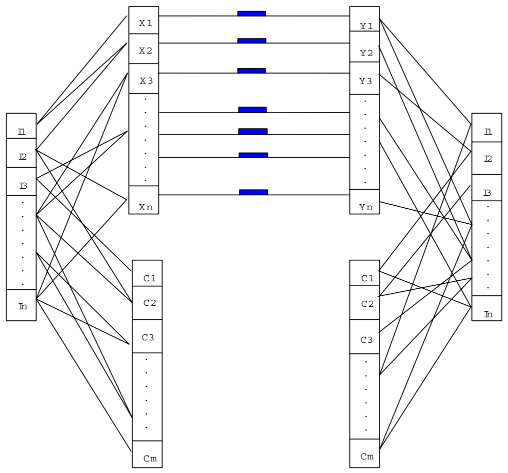

Figure 5.3: *Simultaneous decoding of the LT encoded sources X and Y using inter-subgraph and linking channel message passing*

5.3 Belief Propagation messages

BP algorithm would be used for the simultaneous decoding of previously defined LT codes. As known, the updates in lth round of the BP algorithm are as

$$\begin{aligned} \tanh\left(\frac{m_{o,i}^{(l)}}{2}\right) &= \prod_{i' \neq i} \tanh\left(\frac{m_{i'o}^{(l)}}{2}\right), \\ m_{i,o}^{(l+1)} &= m_o + \sum_{o' \neq o} m_{o',i}^{(l)}. \end{aligned} \tag{5.79}$$

The set of variable nodes is comprised of intermediate nodes and the set of checknodes is made up of input nodes and payload bits. $m_{i,o}$ represents the messages sent from varibale node i to check node o whereas $m_{o,i}$ indicates the messages leaving check nodes to variable nodes. m_o is the initial Log

Likelihood Ratio (LLR) of every check bit o. In other words the LLR of a received symbol Y is defined as

$$\ln \frac{Pr[X=0|Y]}{Pr[X=1|Y]}\ , \tag{5.80}$$

where X is the symbol sent over the channel. The initial reliabilities of $(c_1, \ldots, c_m)$ and $(i_1, \ldots, i_n)$ are ∞ and 0 respectively, whereas the initial LLR of y_i is also zero assuming that x_i has not been decoded.

Hence, the complete knowledge of payload bits is used to decode totally unknown intermediate symbols $(i_1, \ldots, i_n)$. Prior probabilities are clearly determined by (5.73) whereas extrinsic information is a function of source symbol messages which are passed to the corresponding symbols of the other decoder. By an abuse of notation used in [8] , extrinsic messages are defined as

$$P_{X_i}^{(E)} = \psi(x_i, m_i^{(x)})\,, \tag{5.81}$$

where $\psi : \{0,1\} \times \mathcal{R} \to [0,1]$ represents the function $\psi(a,b) = \frac{1}{2} + \frac{1}{2}\sigma(a)\tanh(\frac{b}{2})$, in which $\sigma : \{0,1\} \to \{+1,-1\}$ converts between two different kinds of binary symbols, where $\sigma(0) = +1$ and $\sigma(1) = -1$. Then the link node message from x_i to y_i can be expressed as

$$c_i^{(y)} = \log \frac{\sum_{x_i} p_{X_i,Y_i}(x_i, y_i = 0)\psi(x_i, m_i^{(x)})}{\sum_{x_i} p_{X_i,Y_i}(x_i, y_i = 1)\psi(x_i, m_i^{(x)})}. \tag{5.82}$$

The same approach would yield $c_i^{(x)}$ as the sent message from y_i to x_i,

$$c_i^{(x)} = \log \frac{\sum_{y_i} p_{X_i,Y_i}(x_i = 0, y_i)\psi(y_i, m_i^{(y)})}{\sum_{y_i} p_{X_i,Y_i}(x_i = 1, y_i)\psi(y_i, m_i^{(y)})}. \tag{5.83}$$

Given the initial reliabilities of a fraction of x messages, BP begins to run between nodes. Messages $c_i^{(y)}$ and $c_i^{(x)}$ are also sent from x_i to y_i and the other way round at each iteration.

6 Sources With Memory

In this section, first the properties of Gilbert-Elliot(GE) channels and some details on calculating the capacity of these channels are briefly explained. In the last part of the section, the *BP* messages on the scheme of simultaneous decoding of the sources that have *GE* channel as their correlation channel will be shortly introduced.

6.1 Gilbert-Elliott channels

In this section, bold upper case letters represent vectors and plain lower case letters along with subindices indicate a specific component of a vector. Now we will assume that a Markov channel underlies our BSC channel between x and y, which means that the channel state s_i changes at each time step and that translates to the transition of channel parameter θ_i that defines Markov-Modulated Binary Symmetric Channel (MMBSC). A specific type of MMBSC is the *GE* channel which is a binary-input, binary-output two-state hidden Markov process. This channel has state sequence $\mathbf{S} \in \mathcal{S} := \{B, G\}^n$ where B represents a bad BSC whereas G represents a good BSC with a lower crossover probability. Let $\mathbf{Y} \in \{0,1\}^n$ be the channel output vector in response to the channel input vector $\mathbf{X} \in \{0,1\}^n$,

$$\begin{aligned} &\mathbf{Y} = \mathbf{X} \oplus \mathbf{Z} \\ &Pr(Z_i = 1 | S_i = s_i) = \eta_{s_i}, \end{aligned} \tag{6.84}$$

where $\mathbf{Z}$ is the noise sequence and

$$\eta_G \leq \eta_B \leq 1/2. \tag{6.85}$$

The state transition probabilities are given by $b := Pr(S_{i+1} = B | S_i = G)$ and $g := Pr(S_{i+1} = G | S_i = B)$. The steady-state probablities can be derived by $Pr(S_i = B) = b/(b+g)$ and $Pr(S_i = G) = g/(b+g)$ respectively.

6.2 Capacity of the Gilbert-Elliot channels

The procedure on how to calculate the capacity of a GE channel has been discussed in [13]. This method gives a fast recursive algorithm to estimate the capacity when P_G, P_B and ρ are given. ρ is the *good-to-bad* ratio of the channel and is given as

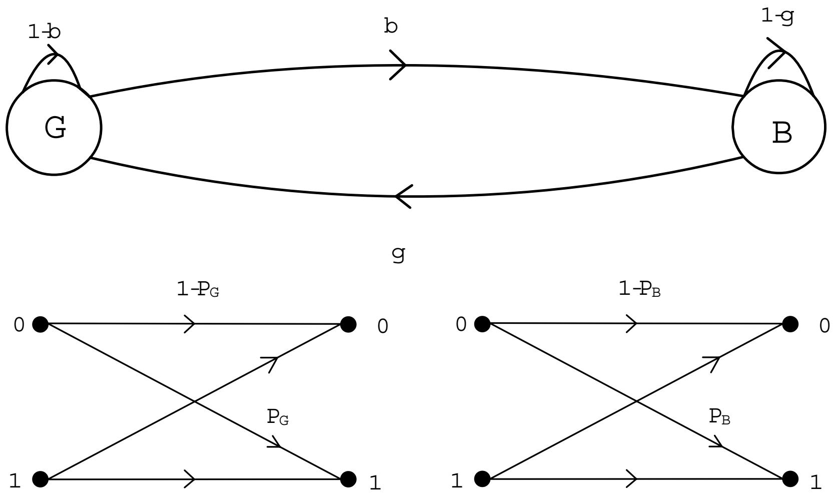

Figure 6.4: *GE channel model is depicted in the Figure. P_G and P_B indicate crossover probabilities in Good and Bad states respectively, g and b are the between states transition probabilities.*

$$\rho \triangleq Pr[s_l = G]/Pr[s_l = B] = g/b.$$

$\rho = 0$ and $\rho = \infty$ represent a channel that only has one state. According to [13], for $\xi \in \{G, B\}$ and using induction on l, it can be shown that

$$Pr[s_l = \xi | s_0 = \xi] - Pr[s_l = \xi | s_0 \neq \xi] = (1 - g - b)^l, \tag{6.86}$$

which yields the definition of the *channel memory parameter* μ,

$$\mu \triangleq 1 - g - b.$$

$\mu = 1$ holds when we are dealing with a fixed state channel, whereas $\mu = -1$ holds for regularly alternating channels. If $\mu < 0$, the channel has a *oscillatory* memory, $\mu > 0$ pertains to the case when channel has a *persistent* memory. Z is also defined as $Z = X \oplus Y$, $z_i = x_i \oplus y_i$. It is easy to see that **pmf** [1] of Z is the same as $P(Y|X)$, where the vectors $\mathbf{X}$ and $\mathbf{Y}$ are input and output vectors of the *GE* channel respectively. Now, the definition of another parameter will be given.

[1]Probability Mass Function

Definition 6.1 *$q_l^\star(z_{l-1}, s_0)$ denotes the probability of channel error at the lth use conditioned on the initial state and previous channel errors,*

$$q_l^\star(z_{l-1}, s_0) = Pr[z_l = 1 | z_{l-1}, s_0].$$

$q_l^\star(z_{l-1})$ denotes the channel error probability only conditioned on previous channel errors,

$$q_l(z_{l-1}) \triangleq E[q_l^\star(z_{l-1}, s_0) | z_{l-1}] = Pr[z_l = 1 | z_{l-1}],$$

where $E(\cdot)$ denotes expected value.

Skipping details which could be read in [13], we proceed with defining recursions which will come to use while calculating the channel capacity:

$$\begin{aligned} q_{l+1}^\star(z_l, s_0) &= \nu(z_l, q_l^\star(z_{l-1}, s_0)) \\ q_{l+1}(z_l) &= \nu(z_l, q_l(z_{l-1})), \end{aligned} \tag{6.87}$$

where $\nu(\cdot,\cdot)$ is defined by

$$\nu(0,q) \triangleq \begin{cases} P_G + b(P_B - P_G) + \mu(q - P_G)\left[\frac{1-P_B}{1-q}\right], & P_B \neq 1 \\ (1-b)P_G + b & P_B = 1, q \neq 1 \end{cases} \tag{6.88}$$

and

$$\nu(1,q) \triangleq \begin{cases} P_G + b(P_B - P_G) + \mu(q - P_G)(P_B/q), & P_G \neq 1 \\ (1-g)P_B & P_G = 0, q \neq 0 \end{cases} \tag{6.89}$$

for $P_G \leq q \leq P_B$. The initial values for the recursions are

$$q^\star(s_0) = \begin{cases} P_G & s_0 = G \\ P_B & s_0 = B \end{cases}.$$

As known, capacity can be written as follows:

$$C = \lim_{l\to\infty} \frac{1}{l} \max_{p(x_l)} I(x_l; y_l). \tag{6.90}$$

The capacity of the *GE* channel in bits per channel use is given by

$$C = 1 - \lim_{l\to\infty} E\left[H(q_l)\right] = 1 - \lim_{l\to\infty} E\left[H(q_l^\star)\right], \tag{6.91}$$

where $H(\cdot)$ is the binary entropy function. The sequence in (6.91) converges to capacity of the *GE* channel as l tends to infinity. A particular case is when $P_G = 0.01$, $P_B = 0.22$, $\rho = 1$ and $\mu = 0.98$ which implies $g = 0.01$ and $b = 0.01$. Capacity of this channel turns out to be 0.55 *bits per channel use.* This particular *GE* channel will become of more interest when *SPA* on *GE-LT* graphs will be analysed later in Section 8.

6.3 BP messages for simultaneous decoding of correlated sources

For the decoding of sources which are correlated according to the proposed model of the previous section, two other subgraphs are required to be added to the major scheme proposed in [7]. According to [9], each of these extra subgraphs would add four extra messages represented as α, β, χ and ζ to the scheme. Messages α, β and ζ would be sent out from channel factor nodes to the future state, current state and symbol-variable node respectively and χ is the extrinsic message sent from LT subgraph to the channel subgraph. We let $\alpha(s_i)$ denote the message from the current state variable node to the channel factor node and $\beta(s_{i+1})$ represent the message from the future state variable node to the channel factor node.
While decoding, the operations in LT subgraphs are the same as in the memoryless case, however there are two new messages passed through the Markov chain [8]. These messages for the decoder of source y are calculated by

$$\begin{aligned}
\alpha(s_{i+1}) &= K \sum_{x_i, y_i, s_i} p_{X_i,Y_i}(x_i, y_i|s_i)\alpha(s_i) \\
&\qquad .p_{S_{i+1}}(s_{i+1}|s_i) p_{X_i}^{(E)}(x_i) p_{Y_i}^{(E)}(y_i) \\
\beta(s_i) &= K \sum_{x_i, y_i, s_{i+1}} p_{X_i,Y_i}(x_i, y_i|s_i)\beta(s_{i+1}) \\
&\qquad .p_{S_{i+1}}(s_{i+1}|s_i) p_{X_i}^{(E)}(x_i) p_{Y_i}^{(E)}(y_i),
\end{aligned} \tag{6.92}$$

where K denotes a normalization constant. ζ for source decoder y would be given by

$$\zeta = \log \frac{\sum_{x_i, \mathbf{s} \in S} p(y_i=0, x_i|s)p(s).p_{X_i}^{(E)}(x_i)}{\sum_{x_i, \mathbf{s} \in S} p(y_i=1, x_i|s)p(s).p_{X_i}^{(E)}(x_i)} \quad ,$$

where the prior density $p(s)$ is used in the initial step, followed by values of α in the next steps, whereas $p_{X_i}^{(E)}(x_i)$ at the initial step would be replaced by the prior probability $p(x_i)$. Message χ can be calculated in the same manner as in memoryless case.

Messages (5.82) and (5.83) would then be modified as

$$c_i^{(y)} = \log \frac{\sum_{x_i,s_i,s_{i+1}} p_{X_i,Y_i}(x_i, y_i = 0|s_i)\alpha(s_i)\beta(s_{i+1}).p_{S_{i+1}}(s_{i+1}|s_i)p_{X_i}^{(E)}(x_i)}{\sum_{x_i,s_i,s_{i+1}} p_{X_i,Y_i}(x_i, y_i = 1|s_i)\alpha(s_i)\beta(s_{i+1}).p_{S_{i+1}}(s_{i+1}|s_i)p_{X_i}^{(E)}(x_i)}$$

$$c_i^{(x)} = \log \frac{\sum_{y_i,s_i,s_{i+1}} p_{X_i,Y_i}(x_i = 0, y_i|s_i)\alpha(s_i)\beta(s_{i+1}).p_{S_{i+1}}(s_{i+1}|s_i)p_{Y_i}^{(E)}(y_i)}{\sum_{y_i,s_i,s_{i+1}} p_{X_i,Y_i}(x_i = 1, y_i|s_i)\alpha(s_i)\beta(s_{i+1}).p_{S_{i+1}}(s_{i+1}|s_i)p_{Y_i}^{(E)}(y_i)}. \tag{6.93}$$

7 Joint Source-Channel Code Design

Now, we will step in to the venture of designing suitable LT codes for the proposed problems. We will almost use the notation of [15] for this part. At first, we will begin with the memoryless case. In order to lower the design complexity, the scheme in Figure 5.3 will be thought of as classical LT codes scheme. This scheme comprises of intermediate nodes as variable nodes and the set of input symbol nodes and payload bits as LT checknodes or output symbols. As a general joint source-channel code design, payload bits have been assumed to be sent over a Binary Input Additive White Gaussian Noise (BIAWGN) channel. The scheme for the decoder of the source x which can be easily genaralized to that of source y due to symmetry, will be explained now.

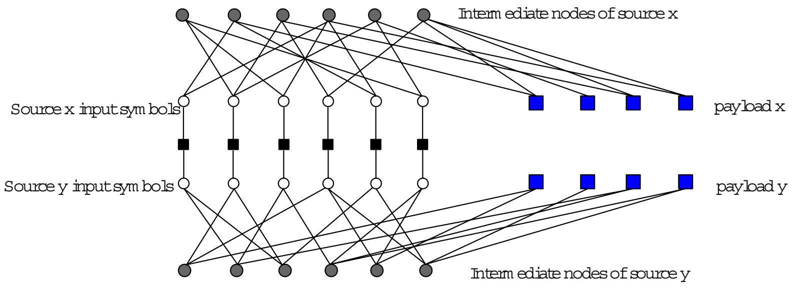

Figure 7.5: *Simultaneous decoding of two sets of payload bits. Input symbols and payload bits could be thought of as to have been produced through LT coding of intermediate nodes*

Let n denote the input length, m_1 represent the payload length, α_1 and α_2 denote the average input node degree (intermediate node degree) from the perspective of edges emanating from the variable nodes and payload bits respectively. β_1 and β_2 represent node degree distributions of input symbols and payload bits. Counting the edges at both sides yields

$$\begin{aligned} &\alpha_1 n + \alpha_2 n = \beta_1 n + \beta_2 m \\ &\frac{m}{n} = \frac{\alpha_1 + \alpha_2 - \beta_1}{\beta_2} \geq \frac{H(X|Y)}{C_1}\,, \end{aligned} \tag{7.94}$$

where $\alpha_1 = \beta_1$ and C_1 is in terms of [bits per channel use] and is the capacity of the channel over which payload is sent. The inequality in the latter equation arises from the fact that always, at least as many bits as the amount of source entropy are required to be sent over the channel. In addition to that, given with the input length ,n, the joint source-channel code sent over the channel is of length m, and the source entropy is H [bits/symbol], then $mC \geq nH$ must always hold. This will impose a constraint on the node average parameters so that,

$$\frac{\alpha_2}{\beta_2} \cdot C \geq H(X|Y), \tag{7.95}$$

which translates to the linear optimization problem of getting the term $\frac{\alpha_2}{\beta_2}$ as close to $\frac{H(X|Y)}{C}$ as possible where $\frac{\alpha_2}{\beta_2}$ is the code rate and is determined according to the desired Slepian-Wolf point. One example could be $R_1 = H(X)$ and $R_2 = H(Y|X)$ that would yield a corner point on the line (5.71). Other points could be also obtained by an appropriate selection of R_1 and R_2. Let $\omega_d^{(2)}$ indicate the probability that an edge connected to payload bits, has $d-1$ neighboring edges of that kind. We know that

$$\beta_2 = \frac{1}{\sum_d \frac{\omega_d^{(2)}}{d}} \ ,$$

which would alter the term subject to minimization to

$$\alpha_2 \sum_d \frac{\omega_d^{(2)}}{d}. \tag{7.96}$$

Besides, there are some constraints. Let $\omega_d^{(1)}$ represent the probability that an edge connected to the set of input symbols, has $d-1$ neighboring edges of the same kind,

$$\begin{cases} \sum_d \omega_d^{(1)} = 1 & 0 \leq \omega_d \leq 1 \quad \text{for } d = 1, \ldots, d_{\max} \\ \sum_l \omega_l^{(2)} = 1 & 0 \leq \tilde{\omega}_l \leq 1 \quad \text{for } l = 1, \ldots, \tilde{d}_{\max} \end{cases} . \tag{7.97}$$

As noted earlier, with only a slight change of notation, the same could be done for the source y decoder. It is again noteworthy that rates for the sources x and y are different. Let σ denote the channel parameter, in all calculations, C_1 and C_2 are the capacities of BIAWGN(σ_1) and BIAWGN(σ_2) respectively and could be easily set to 1 in case we want to do a pure source coding design.

7.1 Joint Source-Channel Coding with side information

Here, firstly the problem for the simple case of joint channel-source coding with side information available at the decoder will be solved. What is meant by side information is that the source Y encoded at the rate $\frac{H(Y)}{C}$ is totally available at the joint decoder and source X will be then decoded at rate $\frac{H(X|Y)}{C}$. Again, note that due to this module being extracted from our bigger scheme which was explained earlier, the set of our checknodes comprises of two subsets of input symbols and payload bits respectively. Since our two subsets have different observation LLRs, two types of edges between variable nodes and check nodes are considered. Moreover, the payload bits are assumed to have been sent over a BIAWGN(σ) channel. Nevertheless, the input codeword $\mathbf{x}$ is assumed to be correlated with $\mathbf{y}$ through the conditional probability $p(\mathbf{y}|\mathbf{x})$.

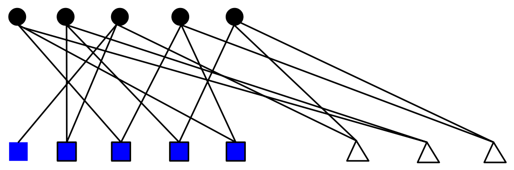

Figure 7.6: *Input symbols(squares), payload bits(triangles)as checknodes and intermediate nodes as variable nodes*

7.1.1 Full Gaussian Approximation

The problem is simplified by assuming that all the messages sent back and forth along the edges are Gaussian like the method proposed in [15]. For the sake of simplicity, the terms cirles, squares and triangles are used instead of intermediate nodes, input symbols and payload bits respectively. The messages are denoted by four random variables $X^l_{(1)}$, $X^l_{(2)}$, $Y^l_{(1)}$ and $Y^l_{(2)}$. $X^l_{(1)}$ and $X^l_{(2)}$ denote the messages from a circle to a square and a triangle respectively at the lth iteration, and $Y^l_{(1)}$ and $Y^l_{(2)}$ represent the messages from a square to a circle and from a triangle to a circle respectively. Then we can proceed with Density Evolution(DE) of our proposed scheme,

$$E\left[X_1^{(l+1)}\right] = \sum_{D=1}^{D_{\max}} \left(\sum_{d=1}^{D} \iota_d^{(1)}(d-1)E[Y_1^l] + I_{D-d}^{(2)}(D-d)E\left[Y_2^l\right] \right) \tag{7.98}$$

$$E\left[X_2^{(l+1)}\right] = \sum_{D=1}^{D_{\max}} \left(\sum_{d=1}^{D} \iota_d^{(2)} (d-1) E[Y_2^l] + I_{D-d}^{(1)} (D-d) E\left[Y_1^l\right] \right), \tag{7.99}$$

where $\iota_d^{(1)}$ represents the probability that a randomly chosen edge from a circle to a square has $d-1$ neighboring edges which are also connected to squares. $\iota_d^{(2)}$ is the probability that a randomly chosen edge from a circle to a square has $d-1$ neighboring edges which are also connected to the set of squares. $I_d^{(1)}$ is the probability that a randomly taken circle is connected to d squares and $I_d^{(2)}$ denotes the probability that a randomly taken circle is connected to d triangles. The mean of the messages $Y_{(1)}^l$ and $Y_{(2)}^l$ could be updated as

$$E\left[\tanh\left(\frac{Y_{(1)}^l}{2}\right)\right] = E\left[\tanh\left(\frac{Z_1}{2}\right)\right] \omega^{(1)} \left(E\left[\tanh\left(\frac{X_{(1)}^l}{2}\right)\right]\right) \tag{7.100}$$

$$E\left[\tanh\left(\frac{Y_{(2)}^l}{2}\right)\right] = E\left[\tanh\left(\frac{Z_2}{2}\right)\right] \omega^{(2)} \left(E\left[\tanh\left(\frac{X_{(2)}^l}{2}\right)\right]\right), \tag{7.101}$$

where $\omega^{(1)}$ and $\omega^{(2)}$ are the characteristic functions of edge degree distributions at the squares and triangles respectively. Z_1 and Z_2 are the observation LLR s of those sets. From now on, Gaussian approximation will be used to calculate the above messages [15]. Assuming that X is a Gaussian with mean μ and variance $\sigma^2 = 2\mu$, then

$$E\left[\tanh\left(\tfrac{X}{2}\right)\right] = \tfrac{1}{2\sqrt{\pi\mu}} \int_{-\infty}^{\infty} \tanh\left(\tfrac{u}{2}\right) e^{-\frac{(u-\mu)^2}{4\mu}}\ du.$$

Defining $\varphi(x)$ for $x \in [0, \infty)$ as

$$\varphi(x) = 1 - \frac{1}{2\sqrt{\pi x}} \int_{-\infty}^{\infty} \tanh\left(\frac{u}{2}\right) e^{-\frac{(u-x)^2}{4x}} du, \tag{7.102}$$

then

$$\begin{aligned} E\left[m_{i,o_1}^{(l+1)} | \deg(i) = D\right] &= (d-1).E\left[m_{o_1,i}^{(l)}\right] + (D-d).E\left[m_{o_2,i}^{(l)}\right] \\ E\left[\tanh\left(\frac{m_{i,o_1}^{(l+1)}}{2}\right)\right] &= 1 - \varphi\left(E\left[m_{i,o_1}^{(l+1)}\right]\right), \end{aligned} \tag{7.103}$$

where o_1 and o_2 represent the the set of square and triangle checknodes. Before proceeding to following relations, it is noteworthy to mention that the sum of two independent Poisson random variables remains Poisson. Hence, the only reason for using different notations is to clarify whether the input degree distribution corresponds to the input bits or the payload bits. Then we have

$$\begin{aligned} & E\left[\tanh\left(\frac{m_{i,o_1}^{(l+1)}}{2}\right)\right] = \\ & 1 - \sum_{D=1}^{D=D_{\max}} \sum_{d=1}^{D} \iota_d^{(1)} I_{D-d}^{(2)} \cdot \varphi\left((d-1) \cdot E\left[m_{o_1,i}^{(l)} | \deg(o_1,i) = d\right] + \right. \\ & \left. (D-d).E\left[m_{o_2,i}^{(l)} | \deg(o_2,i) = D-d\right]\right), \end{aligned} \tag{7.104}$$

where $\deg(o_1, i)$ is the the number of edges between the input node i and the set of squares. $\deg(o_2, i)$ indicates the number of edges between the input node i and the set of triangles, and there is a similar equality for $E\left[\tanh\left(\frac{m_{i,o_2}^{(l+1)}}{2}\right)\right]$,

$$\begin{aligned} & E\left[\tanh\left(\frac{m_{i,o_2}^{(l+1)}}{2}\right)\right] = \\ & 1 - \sum_{D=1}^{D=D_{max}} \sum_{d=1}^{D} \iota_d^{(2)} I_{D-d}^{(1)} . \varphi\left((d-1).E\left[m_{o_2,i}^{(l)} | \deg(o_2,i) = d\right] + \right. \\ & \left. (D-d).E\left[m_{o_1,i}^{(l)} | \deg(o_1,i) = D-d\right]\right). \end{aligned} \tag{7.105}$$

Now we will proceed with the update rule at output nodes. Again with abuse of notation in [15] and by denoting the expectation $E\left[\tanh(\frac{Z_1}{2})\right]$ by z_1 and $E\left[\tanh(\frac{Z_2}{2})\right]$ by z_2,

$$\begin{aligned} E\left[m_{o_1,i}^{(l+1)}\right] = \sum_b \omega_b^{(1)} \varphi^{-1} \Bigg(1 - z_1 \Bigg[1 - \sum_{D=1}^{D=D_{\max}} \sum_{d=1}^{D} \iota_d^{(1)} I_{D-d}^{(2)} \varphi\Big((d-1)E\left[m_{o_1,i}^{(l)}\right] + \\ (D-d)E\left[m_{o_2,i}^{(l)}\right]\Big)\Bigg]^{b-1}\Bigg) \end{aligned} \tag{7.106}$$

$$E\left[m_{o_2,i}^{(l+1)}\right] = \sum_b \omega_b^{(2)} \varphi^{-1}\left(1 - z_2\left[1 - \sum_{D=1}^{D=D_{\max}} \sum_{d=1}^{D} \iota_d^{(2)} I_{D-d}^{(1)} \varphi\left((d-1)E\left[m_{o_2,i}^{(l)}\right] + (D-d)E\left[m_{o_1,i}^{(l)}\right]\right)\right]^{b-1}\right). \tag{7.107}$$

According to [17], for successful decoding, we need to guarantee that,

$$\begin{aligned} y_1 &< \sum_b \omega_b^{(1)} \varphi^{-1}\left(1 - z_1\left[1 - \sum_{D=1}^{D=D_{max}} \sum_{d=1}^{D} \iota_d^{(1)} I_{D-d}^{(2)} \varphi\left((d-1)y_1 + (D-d)y_2\right)\right]^{b-1}\right) \\ y_2 &< \sum_b \omega_b^{(2)} \varphi^{-1}\left(1 - z_2\left[1 - \sum_{D=1}^{D=D_{max}} \sum_{d=1}^{D} \iota_d^{(2)} I_{D-d}^{(1)} \varphi\left((d-1)y_2 + (D-d)y_1\right)\right]^{b-1}\right). \end{aligned} \tag{7.108}$$

Regarding the monotonicity of φ^{-1}, the above inequalities can not hold for $y_1 \geq \varphi^{-1}(1-z_1)$ and $y_2 \geq \varphi^{-1}(1-z_2)$ respectively. However, these inequalities are required to be valid for y_1 and y_2 around zero [15]. With taking derivation of above equations around 0, assigning y_2 and y_1 with zero in the former and latter inequalities respectively, and using the fact that the distributions ι_d^2 ,$I_{D-d}^{(1)}$ and also ι_d^1 ,$I_{D-d}^{(2)}$ are independent of each other, upper bounds are obtained on $\omega_2^{(1)}$ and $\omega_2^{(2)}$,

$$\begin{aligned} \omega_2^{(1)} &\geq \frac{1}{\alpha_1 z_1}, \\ \omega_2^{(2)} &\geq \frac{1}{\alpha_2 z_2}, \end{aligned} \tag{7.109}$$

where $\alpha_1 = \sum_d (d-1)\iota_d^{(1)}$ and $\alpha_2 = \sum_d (d-1)\iota_d^{(2)}$. Although the noted method is quite straightforward to use, in practice there are some flaws laying on the path of using it. The problem is that the messages passed from output symbols to input symbols are far from being Gaussian as suggested in [15].

7.2 Semi-Gaussian Approximation

By proposing this scheme, the channel capacity and a corner point of the Slepian-Wolf region are simultaneously aimed to be achieved. Full Gaussian DE was explained in the previous section. Nonetheless, it doesn't appear to provide us with reasonable results since the density of the messages leaving checknodes could hardly be approximated by Gaussian. Moreover, Probability density function (pdf) of these messages is sharper in the middle. The more the degree of a check node increases, this middle apex gets sharper. It makes the full Gaussian approximation more grave. Therefore, in order to design the code, the semi Gaussian approximation was taken on. It means to approximate the messages leaving variable nodes (intermediate nodes) with Gaussian density and then using the exact expression term of (5.79) to track the mean of $m_{o,i}$ messages. The variable nodes Gaussian messages can then be fully expressed by their means since they are assumed to be symmetric $\sigma^2 = 2m$. Hence, our DE simulations are all based on Semi-Gaussian approximation.

7.3 DE constraints

Since the full DE is too complicated to analyse, the Semi-Gaussian approximation of the DE is considered. This means that the messages which go from variable nodes to check nodes are approximated by Gaussian distribution function. That is due to the fact that these messages are obtained as sum of random variables of the same type [15]. In order to evaluate DE, the all zero codeword is assumed to have been sent. Hence, the mean of the sent message from the input symbol nodes is required to increase at every iteration. The mean of the messages passed from input bits to output bits at $l+1$th iteration could be expressed by $\alpha \sum_d \omega_d f_d(\mu)$. This mean is subject to be more than μ. Whereas, $C.\alpha \sum_{d=1}^{D} \frac{\omega_d}{d}$ is the target function that is to be minimized [15]. Thus the constraint set could be expressed by

1. $\forall i = 0, \ldots, N-1 : \quad \alpha \sum_{d=1}^{D} \omega_d f_d(\mu_i) > \mu_i$
2. $\sum_{d=1}^{D} \omega_d = 1$
3. $\forall d = 1, \ldots, D : \omega_d \geq 0$,

where $f_d(\mu_i)$ for a given degree d is expressed as

$$2\mathrm{arctanh}\left(\tanh\left(\frac{\mathrm{Z_1}}{2}\right)\prod_{\mathrm{i=1}}^{\mathrm{d-1}}\tanh\left(\frac{\mathrm{X_i}}{2}\right)\right). \tag{7.110}$$

For our specific problem, as long as the checknodes have been sent on two different channels,

$$\alpha_1 \sum_d \omega_d^{(1)} f_d^{(1)}(\mu) + \alpha_2 \sum_d \omega_d^{(2)} f_d^{(2)}(\mu) > \mu, \tag{7.111}$$

where

$$\begin{aligned} f_d^{(1)}(\mu) &\equiv 2E\left[\mathrm{arctanh}\left(\tanh\left(\frac{\mathrm{Z_1}}{2}\right)\prod_{\mathrm{i=1}}^{\mathrm{d-1}}\tanh\left(\frac{\mathrm{X_i}}{2}\right)\right)\right], \\ f_d^{(2)}(\mu) &\equiv 2E\left[\mathrm{arctanh}\left(\tanh\left(\frac{\mathrm{Z_2}}{2}\right)\prod_{\mathrm{i=1}}^{\mathrm{d-1}}\tanh\left(\frac{\mathrm{X_i}}{2}\right)\right)\right]. \end{aligned} \tag{7.112}$$

The solution to the optimization problem of (7.96) is addressed with either Differential Evolution or linear programming of which the latter has been chosen due to simplicity. The optimization problem comprises of $\omega_d^{(1)}$ and $\omega_d^{(2)}$ as parameter vectors, with fixed α_1 and α_2. Our objective is then to get (7.96) as close to $\frac{R_1}{C}$ as possible. $\alpha_1 \sum_d \omega_d^{(1)} = 1$ along with $\sum_d \omega_d^{(1)} = 1$ and $\sum_d \omega_d^{(2)} = 1$ provide us with three hard constraints.

7.4 A specific joint source-channel code design for a Slepian-Wolf corner point problem

In this section, we will focus on designing a joint source-channel code with side information at the decoder. Assuming that $\mathbf{y}$ has been received intact and is observed at the decoder, the target is to decode $\mathbf{x}$ using its payload bits which have been sent over a BIAWGNC(0.97)[2]. The correlation between sources X and Y are presumed to be modelled by a BSC with crossover probability of $\epsilon = 0.0417$. This results in 0.25 *bits per channel use* for $H(X|Y)$. These assumptions specify getting the term $\alpha_2 \sum_d \frac{\omega_d}{d}$ as close to 0.5 as possible while α_1 and α_2 have been fixed and assigned with 5 and 3 respectively. Taking the constraint (7.111), μ is discretized with steps of 0.02, $\mu \in [0, 10]$. Linear programming provided us with $\omega_d^{(1)}$ and $\omega_d^{(2)}$ and as already known, $\Omega_d = \frac{\beta\omega_d}{d}$. The latter equality is straightforward to obtain from $\omega(x) = \frac{\Omega'(x)}{\Omega'(1)}$ and yields

$$\Omega^{(1)}(x) = 0.0091x + 0.3669x^2 + 0.5843x^3 + 0.0398x^{63} \tag{7.113}$$

$$\begin{aligned}\Omega^{(2)}(x) = {} & 0.2617x^2 + 0.1301x^3 + 0.1453x^4 + \\ & 0.1651x^5 + 0.1121x^8 + 0.0745x^9 + 0.1112x^{19}.\end{aligned} \tag{7.114}$$

Now, in order to measure the error probability with 10% of overhead, the target lower bound of 0.5 for the optimization problem of $\alpha_2 \sum_d \frac{\omega_d}{d} \geq 0.5$ is changed to 0.55. The calculated $\Omega^1(x)$ and $\Omega^2(x)$ go as

$$\begin{aligned}\Omega^1(x) = {} & 0.01x + 0.9094x^2 + 0.0069x^4 + 0.0246x^8 + 0.0051x^{42} + \\ & 0.0049x^{46} + 0.0086x^{62} + 0.0140x^{63} + 0.0165x^{66},\end{aligned} \tag{7.115}$$

$$\begin{aligned}\Omega^2(x) = {} & 0.1664x^2 + 0.3022x^3 + 0.1321x^4 + 0.1501x^5 + \\ & 0.1019x^8 + 0.0667x^9 + 0.0796x^{19},\end{aligned} \tag{7.116}$$

and mean of the messages m_{io} leaving the intermediate nodes versus the number of iterations are depicted for both cases in the Figures 7.7a and 7.7b.

Let m be the number of collected output bits and k be the source length. System information analysis at the decoder side yields

$$\begin{aligned} & k(1 - h(\epsilon)) + mC = k, \\ & m = k/2. \end{aligned} \tag{7.117}$$

In other words, receiver needs to collect $m = \frac{k}{2}(1 + \epsilon)$ bits to decode the intermediate nodes. Assigning σ with 0 changes the problem to a pure source coding problem. For the linear programming approach, α_1 and α_2 were

[2]Binary Input Additive White Gaussian Noise Channel

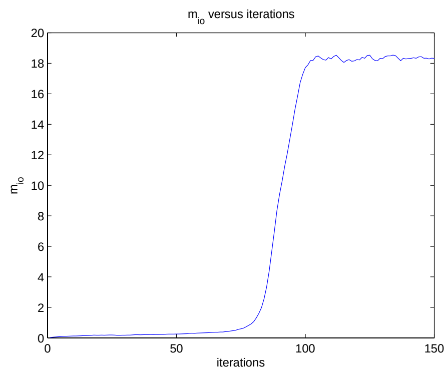

Figure 7.7a: *Mean of the intermediate node messages versus iterations while having no overhead*

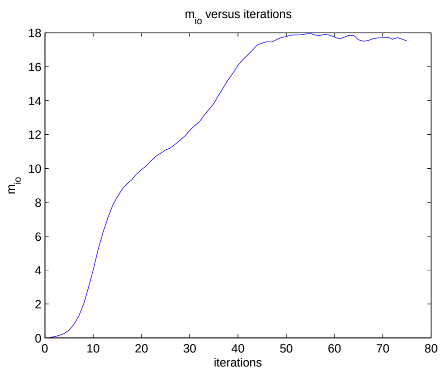

Figure 7.7b: *Mean of the intermediate node messages versus iterations while having 10 % of overhead*

assigned and fixed with 5 and 3 respectively. Another simulation was then run for the pure source coding problem, thereby resulting in $C = 1$ with the correlation between sources modelled by a BSC with the similar crossover probability of 0.0417. This translates into $H(X|Y) = 0.25$ bit and therefore getting $\alpha_2 \sum_d \frac{\omega_d^{(2)}}{d}$ as close to 0.25 as possible. The same discussion again provides us with $\Omega^{(1)}(x)$ and $\Omega^{(2)}(x)$,

$$\begin{aligned}\Omega^{(1)}(x) =& 0.015x + 0.8480x^2 + 0.1370x^{24}, \\ \Omega^{(2)}(x) =& 0.1978x^4 + 0.3302x^5 + 0.2242x^8 + 0.1489x^9 + 0.0988x^{65},\end{aligned} \tag{7.118}$$

when we are concerned with 0 percentage of overhead and

$$\begin{aligned}\Omega^{(1)}(x) =& 0.015x + 0.8480x^2 + 0.1370x^{24}, \\ \Omega^{(2)}(x) =& 0.2148x^4 + 0.3238x^5 + 0.2198x^8 + 0.1460x^9 + 0.0956x^{65},\end{aligned} \tag{7.119}$$

for 4% of overhead. The maximum attainable $m_{i,o}$ is obviously $\alpha_1 z_1 + \alpha_2 z_2$. For the joint source-channel code case, it is straightforward to conclude that $\alpha_1 z_1 + \alpha_2 z_2 \simeq 20.7$. This consequently results in an error probability of approximately 6.10^{-4} while taking the messages leaving intermediate nodes to be Gaussian. While testing the pair of $\Omega_d^{(1)}$ and $\Omega_d^{(2)}$ in (7.118), intermediate node message mean got stuck at 4.5 that implies the quite poor error probability of 6.7%. Whereas the pair in (7.119) yielded the mean of approximately 75 and error proability of 4.10^{-10}.

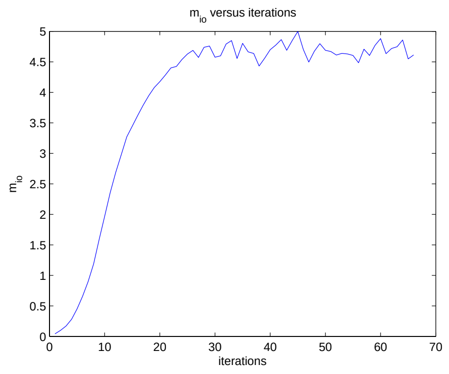

Figure 7.8a: *Mean of the intermediate node messages versus iterations while having no overhead*

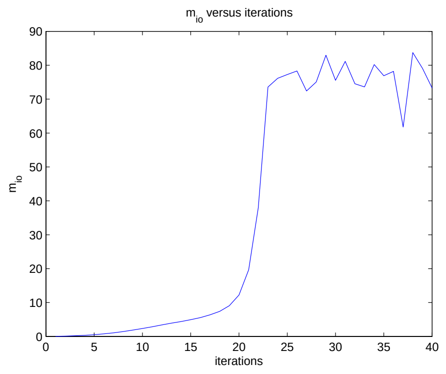

Figure 7.8b: *Mean of the intermediate node messages versus iterations while having 4 % of overhead*

7.5 Slepian-Wolf problem with arbitrary rate allocation

Up to now, the only concern was achieving the corner points of the *Slepian-Wolf* region. From now on and for the rest of this section, the focus will be on fountain code design for the purpose of achieving any arbitrary point of the *Slepian-Wolf* region.

7.6 Approaching the Slepian-Wolf limit using syndromes

The *Slepian-Wolf* problem by arbitrary rate allocation between the sources was first addressed and outlined in the work of Pradhan and Ramchandran as Distributed Source Coding Using Syndromes (DISCUS)[11]. The main idea of the work is to model the correlation between sources with a channel code and then partition that single channel code. This has been inspired by the fact that the *Slepian-Wolf* Coding (SWC) problem is indeed a channel coding problem. This approach has been exemplified by partitioning the G matrix of a $(7, 4, 3)$ binary Hamming code and allocating each partition to encode one of the uniformly distributed sources X and Y. Assume X and Y are two memoryless 7-bit binary random variables with their correlation modeled by $d_H(X, Y) \leq 1$. $d_H(X, Y)$ denotes the Hamming distance between X and Y. Entropies and joint entropies are expressed by $H(X) = H(Y) = 7$ bits and $H(X, Y) = H(Y) + H(X|Y) = 10$ bits respectively. Let x and y denote the codewords allocated to the sources X and Y. Let sources X and Y be encoded at rates R_1 and R_2 such that $R_1 \geq 3$, $R_2 \geq 3$ and $R_1 + R_2 \geq 10$. Now, according to *Slepian* and *Wolf*, it is possible to devise a jointly decoding method at the decoder. At first let's assume we want to approach a corner point on *Slepian-Wolf* region. In this case, Y sends over its complete information using 7 bits whereas X only transmits its syndrome given by $s = Hx$, which contains only 3 bits. Since $d_H(X, Y) \leq 1$, one can easily retrieve x once given y and s. In fact the correlation between the two sources could be modelled by BSC.

Another scenario is to achieve a non-corner point like $R_1 = R_2 = 5$ bits. For this case, the generator matrix of the systematic $(7, 4)$ Hamming code $\mathcal{C}$, defined by (7.120) is broken up

$$G_{k\times n} = [I_4 \;\; P] = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \qquad (7.120)$$

into two generator matrices G_1 and G_2. Each of these matrices is comprised of two rows of G and thereby construct two subcodes C_1 and C_2. 5-bit Syndromes s_1 and s_2 are then transmitted to the decoder as $s_1^T = H_1x^T$ and $s_2^T = H_2y^T$. Appending the syndromes s_1 and s_2 with zeros gives t_1 and t_2. $x \oplus t_1$ and $y \oplus t_2$ are clearly the codewords of C_1 and C_2 respectively. The joint decoder receives s_1 and s_2 and finds the closest codeword to $t = t_1 \oplus t_2$ in $\mathcal{C}$ which is $c = x \oplus t_1 \oplus y \oplus t_2$. Recovering the systematic parts of x and y from c and letting them be denoted by $\hat{a_1}$ and $\hat{a_2}$, one can easily reconstruct the sources as $\hat{x} = \hat{a_1}G_1 \oplus t_1$ and $\hat{y} = \hat{a_2}G_2 \oplus t_2$. Since the Hamming distance between x and y is 1 by maximum and the applied $(7, 4, 3)$ Hamming code has minimum distance of 3, x and y can certainly be recovered error-free.

7.7 Slepian-Wolf problem and fountain codes

Thanking to the insight taken from [11], later works were done to get as close as polssible to the *Slepian-Wolf* region with arbitrary rate allocation. The most important works were carried out by Qian Xu *et al.* using rateless codes and particularly raptor codes. Raptor code is known as the best approximation to a digital fountain [18]. In this work, Irregular repeat accumulative (IRA) has been used as the precoding of the proposed raptor approach.

7.7.1 The coding and decoding scheme

Here, the applied scheme is elaborated. Sources X and Y have been seperately encoded as illustrated in Figure 7.9. Source X is assumed to have been coded with two sets of bits Z_{tx} and S_{lx}. The subindices t and l indicate the number of collected bits of each kind at the decoder side, and index x represents belonging to source X.
Source Y has similarly been encoded with the sets of bits Z_{ty} and R_{my}. As the common term Z_t may suggest, the same output connection is applied to this part of the two graphs. The sets S_l and R_m are connected differently, although still having the same node degree distribution. The idea is firstly

to decode intermediate nodes of $X \oplus Y$ using the the set of $X \oplus Y$ nodes and payload set $Z_{tx} \oplus Z_{ty}$. Then, $R_{mx} \oplus R_{my}$ is constructed with having $X \oplus Y$ at hand. R_{mx} is then recovered through XOR ing $R_{mx} \oplus R_{my}$ and R_{my}. At the final stage, we use all the three sets of Z_{tx}, S_{lx} and R_{mx} for erasure decoding of X intermediate nodes. Having X decoded, we would easily manage to retrieve Y through $X \oplus (X \oplus Y)$.

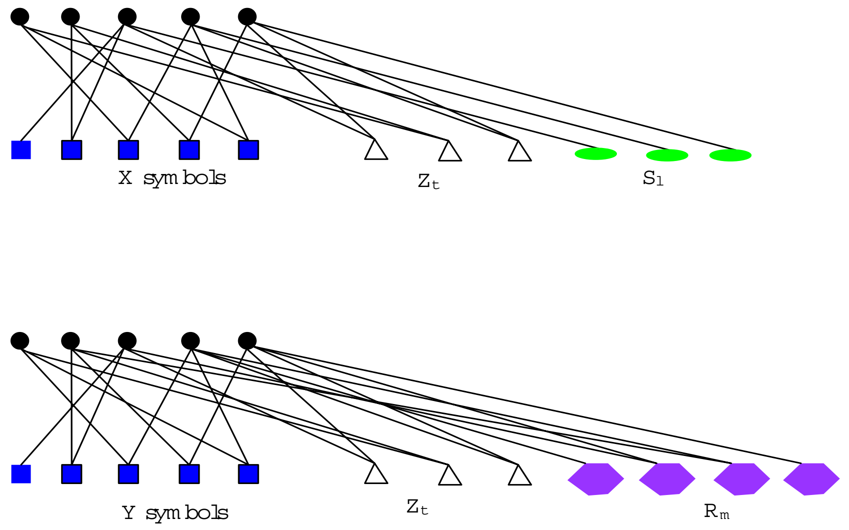

Figure 7.9: *Sources X and Y have been seperately encoded*

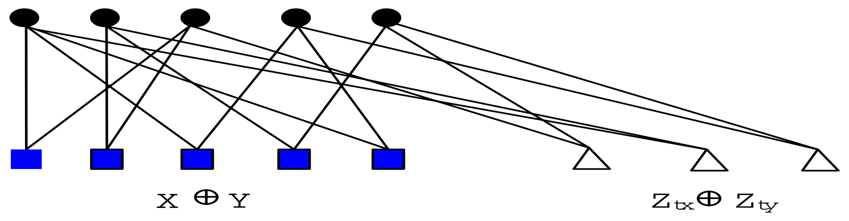

Figure 7.10: *Decoding of $X \oplus Y$ with $Z_{tx} \oplus Z_{ty}$ has been depicted in the figure.*

Assume that the correlation between X and Y is modelled by $BSC(\epsilon)$, X and Y word length to be k. $t = kH(Y|X)$ bits would then suffice to decode $X \oplus Y$. l and m could be adjusted to achieve any point of the *Slepian-Wolf* line and not only the corner points. Eventhough we expect to be confined with some node degree distribution constraints that would deter us from achieving all the desired points.

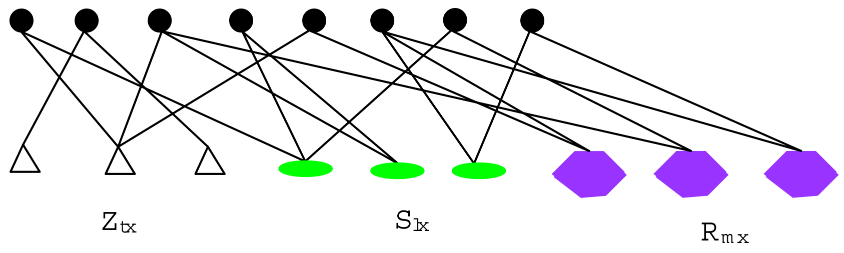

Figure 7.11: *Erasure decoding of X using the payload sets Z_{tx}, S_{lx} and R_{mx}*

7.7.2 Code design

Now, we proceed with design details of the proposed scheme. As mentioned earlier, there are $t + l$ bits used to encode X and $t + m$ bits used to encode Y. In order to decode $X \oplus Y$ and X successfully and achieve the desired *Slepian-Wolf* region, the following constraints are required to hold

$$\begin{aligned} t &= kH(X|Y) \\ t + l + m &= k \\ 2t + l + m &= kH(X, Y). \end{aligned} \tag{7.121}$$

It is easy to conclude that $l + m = k\,(1 - H(X|Y))$. In case X and Y are equiprobable binary sources, it makes sense to let l and k be equal and thereby to achieve the middle point of the *Slepian-Wolf* broken line. To this end, l is assigned by $l = m = \frac{k(1-H(X|Y))}{2}$, where k is the source word length. *DE* is applied on the scheme in the Section 7.1 to derive the edge degree distributions $\omega_d^{(1)}$ and $\omega_d^{(2)}$ by solving the optimization problem of (7.111). Soft constraint of (7.111) was discretized into steps of 0.01 for $\mu \in [0, 20]$. The hard constraint is $\alpha_1 \sum_d \frac{\omega_d^{(1)}}{d} = 1$, where $\omega_d^{(1)}$ and $\omega_d^{(2)}$ correspond to the edge degree distributions for output node sets $X \oplus Y$ and payload bits z_t respectively. It was observed that the decoding performance was degraded for high average node degree of the payload bits. For this specific design problem, we are concerned with the case in which the correlation between the sources X and Y is modelled by *BSC(ϵ)* where $\epsilon = 0.0417$. Hence, the observation *LLR* for the first set output nodes($X \oplus Y$ bits) is calculated as $\log\left(\frac{Pr(X=0)}{Pr(X=1)}\right) = 3.13$. *LLR* of the payload bits would be $\pm\infty$ theoretically, although assigned with ± 20 in the simulations. Let $\Omega^{(1)}(x)$ denote the input bits degree distribution and $\Omega^{(2)}(x)$ be the payload bits degree distribution. It is useful to recall the necessity of erasure decoding of X after retrieving

$X \oplus Y$. $\Omega(x)$ is used to represent the overall output node degree distribution in the Figure 7.11. $\Omega^{(3)}(x)$ is calculated as the node degree distribution of the output node sets S_l and R_m which are depicted in the Figure 7.9,

$$\begin{aligned} &\frac{t}{k}\Omega^{(2)}(x) + \frac{k-t}{k}\Omega^{(3)}(x) = \Omega(x) \\ &\Omega^{(3)}(x) = \frac{\Omega(x) - \frac{t}{k}\Omega^{(2)}(x)}{1-\frac{t}{k}}\cdot \end{aligned} \tag{7.122}$$

$\Omega(x)$ is the output node degree distribution required for erasure decoding and can be assigned with robust Soliton distribution or be extracted from the table in [5]. The latter equality in (7.122) adds a new constraint to what we had beforehand,

$$\begin{aligned} &\Omega(x) > \frac{t}{k}\Omega^{(2)}(x), \\ &\Omega_d > H(Y|X)\Omega_d^{(2)}(x), \\ &\Omega_d > H(Y|X)\beta_2\frac{\omega_d^{(2)}}{d}, \\ &\omega_d^{(2)} < d\frac{\Omega_d}{\alpha_2}\cdot \end{aligned} \tag{7.123}$$

The latter relation has been concluded from the fact that

$$\tfrac{\alpha_2}{\beta_2} = \tfrac{t}{k} = H(Y|X)\cdot$$

Although the ideal is to have an overhead free decoding, one is always required to add some overhead in order to come up with successful decoding with *SPA*. In fact the latter inequality of (7.123) added to set of constraints detailed in Section 7.3 restricts us to only be able to work with a limited subset of average node degrees α_1 and α_2.

7.7.3 Implementation

In order to simulate the proposed system, *SPA* is applied on the scheme illustrated in the Figure 7.10. α_1 and α_2 are assigned with 2.8 and 2.25 respectively, whereas the source length is set to 4000. It is noteworthy that *Doping* is also used to assist the decoding process. *Doping* means to reveal the intermediate bit which has the least reliability at each step. $\Omega_d^{(1)}$ and $\Omega_d^{(2)}$

are derived with semi-Gaussian *DE* as

$$\Omega^{(1)}(x) = 0.06x + 0.3572x^2 + 0.4937x^3 + 0.0744x^6 + 0.0148x^7,$$
$$\Omega^{(2)}(x) = 0.0903x^3 + 0.0057x^4 + 0.3302x^5 + 0.1872x^8 + 0.1489x^9 + 0.2224x^{19}. \tag{7.124}$$

$\Omega^{(3)}(x)$ is consequently calculated from (7.122) , where

$$\begin{aligned}\Omega(x) = 0.00797x + 0.4936x^2 + 0.1662x^3 + 0.0726x^4 + 0.0826x^5 + \\ 0.0561x^8 + 0.0372x^9 + 0.0556x^{19} + 0.0250x^{65} + 0.0031x^{66}\end{aligned} \tag{7.125}$$

has been taken from [5], hence

$$\begin{aligned}\Omega^{(3)}(x) = 0.0106x + 0.6531x^2 + 0.1915x^3 + 0.095x^4 + \\ 0.0123x^8 + 0.0334x^{65} + 0.0042x^{66}.\end{aligned} \tag{7.126}$$

The error probability of 0.006 is achieved with $t = 1300$ and 50 iterations which is unacceptable and in contrast with *DE* results. This can be reasoned in different ways of which the most probable might be the inefficiency of the applied *DE*. For instance, the messages coming out of input intermediate nodes may be far away from Gaussian.

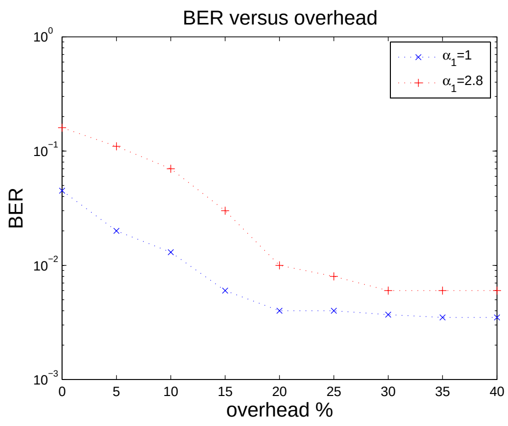

Figure 7.12: *BER with respect to overhead for the cases $\alpha_1 = 1$ and $\alpha_1 = 2.8$ is depicted with blue dashed and red dashed lines respectively.*

Due to the error floor problem of *LT* codes which is illustrated in the Figure 7.12, no remarkable improvement was observed when increasing t even up to 1600. Theoretical value of t is $nH(Y|X) = 1000$, where n is the source message length and $H(Y|X) = 0.25$ while dealing with *BSC(ϵ)*, $\epsilon = 0.0417$ as before. Unexpectedly the best result is achieved when α_1 is assumed to be 1. With covered graph assumption, this is equivalent to the complete matching between input $X \oplus Y$ bits and intermediate bits. What meant by covered graph is that the attaching subgraph of these two sets should have no dangling intermediate node. The best result is achieved with $\alpha_1 = 1$ and $\alpha_2 = 2.3$ yielding the *BER* of 0.004 while 1200 payload bits have been collected.

$$\begin{aligned}\Omega^{(1)}(x) &= x \\ \Omega^{(2)}(x) &= 0.1333x^2 + 0.2675x^5 + 0.2242x^8 + \\ &\quad 0.1489x^9 + 0.2224x^{19} + 0.0036x^{65},\end{aligned} \tag{7.127}$$

means that the left hand set of output nodes have only come to the use of assigning the intermediate nodes with obsevation *LLR*. This, along with the suffering from the error floor problem, arises the motivation of taking on the raptor code approach instead of *LT* codes that was used earlier.

7.8 Raptor code approach for the Slepian-Wolf problem

As discussed in the previous section, our *LT* code design failed to get close to capacity while decoding $X \oplus Y$. Hence, in order to lower the Bit Error Rate (BER) down to zero, the idea of using raptor codes is devised instead. The idea translates to adding some redundant bits to the end of the source words to construct *LDPC* codewords and consequently adding some *LDPC* checknodes to the payload bits. This precoding would effectively assist to deal with the remaining *BER* and error floor problem of *LT* code approach. In other words, this precoding follows as adding m redundant bits to the source words $\mathbf{x}$ and $\mathbf{y}$. Thereby, m *LDPC* check bits are added to the set of checknodes or output nodes. For the sake of simplicity in implementation, a **high rate** regular *LDPC* is taken on as the precoder. To cancel out a remaining *BER* of $\mathscr{E}_r$, the corresponding capacity is calculated, $Cap(BSC(\mathscr{E}_r))$, for which we are seeking the precoder. As already known, $Cap(BSC(\mathscr{E}_r)) = 1 - h(\mathscr{E}_r)$ and rate of a *LDPC* taken from (d_v, d_c) ensemble with d_v denoting variable node degree and d_c indicating the checknode degree is calculated by $R = 1 - \frac{d_v}{d_c}$. Therefore $R < C$, where C is capacity, implies that the inequality $1 - \frac{d_v}{d_c} > Cap(BSC(\epsilon))$ needs to hold. The latter inequality yields $\frac{d_v}{d_c} = \frac{m}{n}$, where m denotes the number of *LDPC* checknodes and n indicates the number of *LDPC* variable nodes or *LDPC* codeword length. For the purpose of canceling out a *BER* of $\mathscr{E}_r$, we need to make sure that $\frac{m}{n} > h(\mathscr{E}_r)$. For instance, to address our previous section problem or remaining error of 0.004,

$$\frac{m}{n} > h(0.004) = 0.0376, \tag{7.128}$$

which practically translates to adding 151 redundant bits to the set of variable nodes (intermediate bits) and 151 *LDPC* check nodes to the set of output check nodes. Our *LDPC* pre-code is chosen from a regular $(3, 51)$ *LDPC* ensemble that yields a precoder of rate 0.9412. This pre-code compensates for the remaining error of 0.004 observed in the previous section. This introduces 250 *bits* of redundant and checknode bits. Hypothetically, a *LDPC* code of rate 0.9624 suffices for that purpose. Nonetheless, since we don't expect a regular *LDPC* to be capacity achieving on the **BSC**, it makes sense to take on lower rate codes to achieve the capacity and lower the error down to zero. Fortunately, there is no need to take on an ensemble with the rate of much lower than capacity to come up with the zero *BER*. Number of collected output symbols (payload bits) to achieve the zero *BER* was estimated to be 1060. What meant by estimation is that $t = 1060$ seemed adequate to lower the *BER* down to zero on more than 90% of the graphs randomly taken from the designed ensemble. We continue with decoding $\mathbf{x}$ as explained in the previous section. Having $\Omega^{(2)}(x)$ as in (7.127), we manage to derive $\Omega^{(3)}(x)$ using (7.122),

$$\begin{aligned}\Omega^{(3)}(x) = 0.0106x + 0.6137x^2 + 0.2216x^3 + 0.0968x^4 + \\ 0.0210x^5 + 0.0001x^8 + 0.0321x^{65} + 0.0041x^{66}.\end{aligned} \tag{7.129}$$

Then having $t = 1060$, the second $\mathbf{y}$ encoding bits set length m is assigned with 1500. We then look for the appropriate second $\mathbf{x}$ encoding bits set length l that yields the error free decoding of $\mathbf{x}$. Basically, since $H(X) = 1$, at least k encoding bits are required for the successful decoding of $\mathbf{x}$. For this case, $k = 4000$ and (7.121) imply that for the overhead free decoding of $\mathbf{x}$, at least $l = 1440$ *bits* are needed. Therefore we vary l and run belief propagation on the scheme depicted in the Figure 7.11. The only difference is that the upper row circles are replaced with $\mathbf{x}$ bits rather than intermediate bits, to achieve the lowest possible *BER*. With $l = 1650$, *BER* of 0.007 was achieved, thereby persuading us to use raptor code again. Theoretically, according to the discussion brought up above, a pre-coder of rate 0.94 would have sufficed to cancel out the error. Although we finally took the ensemble $(4, 44)$ of rate 0.9 that helped us get the *BER* down to zero using $l = 1570$ bits. This translates to being about 3.8 % away from the *Slepian-Wolf* limit $H(X, Y) = 1.25$.

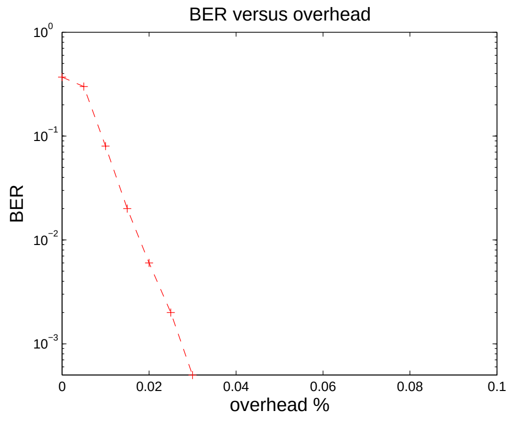

Figure 7.13: *BER with respect to overhead while decoding $\boldsymbol{x}$ with raptor code.*

Another test is performed with $m = 1700$ bits and the same *LDPC* ensemble as the pre-coder. This leads us to come up with $l = 1450$ bits to lower the *BER* down to zero, thereby to be 5.4% away from $H(X, Y)$. Being away from the *Slepian-Wolf* limit could be addressed by the source message length. Moreover, the source message length of infinity could get us closer to achieve the limit. Defining $R_x = \frac{t+l}{k}$ and $R_y = \frac{t+m}{k}$, the two obtained points are shown in the Figure 7.14.

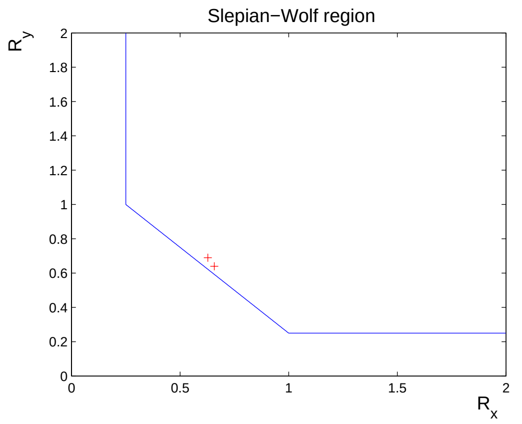

Figure 7.14: *Slepian-Wolf region for the sources X and Y along with the achieved points depicted with red plus points are illustrated in the Figure.*

Design details of the joint source-channel code for the *Slepian-Wolf* problem is not included, although the points depicted in the Figure 7.14 are expected to get further to the limit $\frac{H(X,Y)}{C}$ when σ increases. σ denotes the channel parameter.

8 Fountain Codes On Markov Channels

The problem of designing and applying *LDPC* codes for correlated sources on Markov channels have been mostly addressed by Andrew W. Eckford *et al.* in [9], [10]. We will exploit the notation of [9] for this section. Moreover, *DE* on Markov channels have been limited to *GE* channels. In this section and from now on, the concern is only with *GE* channels which is a particular realization of hidden Markov channels. The Sum-Product-Algorithm could be applied on these codes as before. Nevertheless besides messages passed between symbol variable nodes and factor nodes, 4 other messages are added. These messages are to represent the backward and forward messages between channel factor nodes and from channel factor nodes to symbol variable nodes and the otherway round respectively. Here, these messages are expressed as they have been derived in [9]. As already known from LDPC codes, the codeword $\mathbf{x}$ is valid, if it satisfies all the parity check nodes. Let h_k or $h_k(\boldsymbol{x}_k)$ denote a factor node and the subscript k imply the restrictions on $\mathbf{x}$ codeword. In other words, k indicates the specific positions of $\boldsymbol{x}$ that participates in the parity check equation. Letting m be the number of factor nodes,

$$h(\boldsymbol{x}) = \prod_{k=1}^{m} h_k(\boldsymbol{x}_k).$$

The probability of a specific codeword $\boldsymbol{x}$ to be received is expressed as

$$p(\boldsymbol{x}) = h(\boldsymbol{x})/|\mathcal{C}|,$$

which is derived by the assumption that all codewords in $\mathcal{C}$ are equally likely to be transmitted. From Markov chain, the probability of the state sequence $\boldsymbol{s}$ can be easily calculated as follow,

$$p(s) = p(s_1)\prod_{j=1}^{n-1} p(s_{j+1}|s_j).$$

Now, the joint **pmf** for the codeword $\boldsymbol{x}$, channel output $\boldsymbol{y}$ and state sequence $\mathbf{s}$ is given as,

$$\begin{aligned} p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{s}) &= p(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{x})p(\boldsymbol{s})p(\boldsymbol{x}) \\ &= \frac{1}{|\mathcal{C}|}\left(\prod_{i=1}^{n} p(y_i|x_i,s_i)\right) \\ &\quad \cdot \left(p(s_1)\prod_{j=1}^{n-1} p(s_{j+1}|s_j)\right)\left(\prod_{k=1}^{m} h_k(\boldsymbol{x}_k)\right). \end{aligned} \tag{8.130}$$

The first equality of (8.130) has been concluded from the fact that Markov channel state transition is independent from the transmitted codeword, hence the joint **pdf** of them would break up to the multiplication of **pdf**s. Now, having the basic relations, the sum-product messages can be derived. As noted earlier, a Markov subgraph needs to be attached to the formerly devised *LDPC* subgraph. The purpose of adding this subgraph is to send the channel messages to *LDPC* subgraph and receiving the extrinsic messages from *LDPC* subgraph. It also comes to the use of exchanging two internal messages between *GE* subgraph channel factor nodes and state variable nodes as shown in Figure 8.16. These messages are known as *forward* and *backward* messages. Forward message $\mathbf{A}$ gives an estimation of the future state mode. This estimation is based on the extrinsic message $\mathbf{D}$ received from the current symbol variable node and the forward message $\mathbf{A}^-$ received from the previous state channel factor node. Backward message $\mathbf{B}$ gives an estimation of the current state mode based on the extrinsic information received from the current symbol variable node $\mathbf{D}$ and the backward message $\mathbf{B}^-$ received from the future state. Extrinsic message $\mathbf{D}$ gives the reliability of the symbol variable X_i excluding all the information received from the *GE* subgraph. Whereas the channel message $\mathbf{C}$ gives the reliability of the symbol variable X_i only relying on the information received from the Markov subgraph.

Now, we step into the venture of deriving these explained messages. Let $\mathbf{P}$ denote the state transition probability matrix,

$$\mathbf{P} = \begin{bmatrix} 1-b & b \\ g & 1-g \end{bmatrix},$$

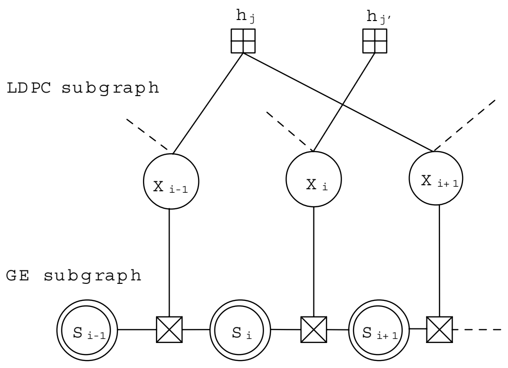

Figure 8.15: *GE-LDPC decoder factor graph*

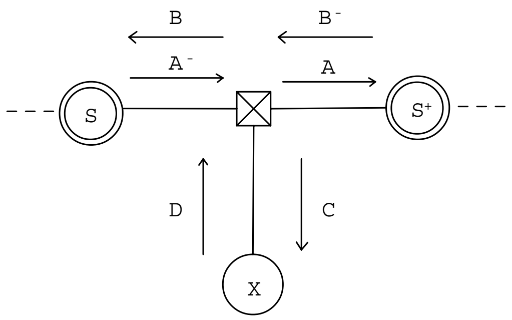

Figure 8.16: *Channel factor graph, backward-forward, extrinsic and channel messages are illustrated*

and σ be the mapping function $\sigma : \{0,1\} \to \{1,-1\}$, $\gamma : \mathbb{R} \times \{0,1\} \to [0,1]$ be defined as

$$\gamma(\lambda, y) = \frac{1}{2}\left[1 + \sigma(y)\tanh\left(\frac{\lambda}{2}\right)\right], \tag{8.131}$$

where λ is the log-likelihood ratio $\log(p(0)/p(1))$, and $\gamma(\lambda, y)$ can be taken as $p(y)$. Letting

$$\begin{aligned} N &= \begin{bmatrix} \eta_G & 0 \\ 0 & \eta_B \end{bmatrix}, \\ I &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \end{aligned} \tag{8.132}$$

and then defining $\mathbf{E}$ as

$$\mathbf{E}(\lambda, y) = \mathbf{N}(1 - \gamma(\lambda, y)) + (\mathbf{I} - \mathbf{N})\gamma(\lambda, y),$$

allows us to express the messages **A**,**B**,**C** and **D** in the compact form. *SPA* which results in the following equalities is avoided, although one can easily expand out these relations and come up with the more clear terms of *SPA*.

$$\begin{aligned} A &= \frac{P^T E(D,Y)A^-}{u_2^T P^T E(D,Y)A^-}, \\ B &= \frac{E(D,Y)PB^-}{u_2^T E(D,Y)PB^-}, \\ C &= \sigma(Y)\log\frac{(\mathbf{A}^-)^T(\mathbf{I}-\mathbf{N})\mathbf{P}^T\mathbf{B}^-}{(\mathbf{A}^-)^T\mathbf{N}\mathbf{P}^T\mathbf{B}^-}, \\ D &= \sum_{j=1}^{d} X_j\,. \end{aligned} \tag{8.133}$$

The latter equation is derived almost the same way as the messages leaving symbol variable nodes to symbol factor nodes are achieved. The only difference is that all the messages entering the variable node from the neighboring symbol factor nodes get involved and the observation *LLR* is excluded. The message passing schedule has been outlined in [9] as having n iterations in *LDPC* subgraph for each iteration in GE subgraph. Nevertheless in this work, n has been assigned with 1 for the sake of simplicity.

8.1 DE on LDPC and GE-LDPC subgraphs

DE could be done in different ways. It is possible to either carry out a full Gaussian DE as devised in [17] or apply the method described in [10] by Eckford *et al.*. The former method presumes all messages to be Gaussian. Here, Chung's density evolution on *LDPC* codes which is appliable on *BEC*s and and *BIAWGNC*s is briefly explained. We would then proceed with Eckford's method for correlated sources with *HMM* which with some manipulations will be later applied on Fountain codes too.

8.1.1 Full Gaussian DE on LDPC codes

Let $\lambda(x)$ and $\rho(x)$ be the edge degree distribution of the variable and check nodes respectively. $\lambda(x) = \sum_{i=2}^{d_l} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{i=2}^{d_r} \rho_i x^{i-1}$. As known, rate of the code would be given by $r = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}$. Φ function is defined as

$$\Phi(x) = \begin{cases} \frac{1}{\sqrt{4\pi x}} \int_R \tanh(\frac{u}{2}) exp\left(-\frac{(u-x)^2}{4x} du\right). & if\, x > 0, \\ 0 & if\, x = 0. \end{cases} \tag{8.134}$$

Chung's density evolution is then expressed as

$$\bar{u}_l = \sum_j \rho_j \Phi^{-1} \left\{ \left[\sum_i \lambda_i \Phi(\bar{u}_0 + (i-1)\bar{u}_{l-1}) \right]^{j-1} \right\}, \tag{8.135}$$

where $\bar{u}_l$ indicates mean of the messages leaving factor nodes after lth iteration. $\bar{T}_l$ is defined as

$$\bar{T}_l = \sum_i \lambda_i \Phi(\bar{u}_0 + (i-1)\bar{u}_l), \tag{8.136}$$

which helps us to simplify $\bar{u}_l$ as

$$\bar{u}_l = \sum_j \rho_j \Phi^{-1} \left(\bar{T}_{l-1}^{j-1} \right), \tag{8.137}$$

where $\bar{T}_l = E\left(\tanh(\frac{v_l}{2})\right)$ and v_l is the message sent by variable nodes. Iterative decoding then translates to staircase motion between the two curves (8.137) and (8.136). Successful decoding is feasible as long as the tunnel between the two curves is open. Tunnel between the two curves gets narrower as the *BIAWGN* parameter σ increases [14]. This concept leads to definition of the design optimization problem,

$$\sum_{j=2}^{d_v} \lambda_j = 1,\, \lambda_j \geq 0,$$
$$\sum_i \lambda_i \Phi(\bar{u}_0 + (i-1)\bar{u}_l) > \bar{T} \text{ for } (\bar{T}, \bar{u}) \text{ pairs that satisfy} \begin{cases} \bar{u}_l = \sum_j \rho_j \Phi^{-1}\left(\bar{T}_{l-1}^{j-1}\right) \\ \Phi(\bar{u}_0) < \bar{T} < 1 \end{cases}$$
$$\lambda_2 < \frac{1}{\rho'(1)\exp(\frac{\bar{u}_0}{4})}. \tag{8.138}$$

With the above brief explanation of Gaussian *DE* on *LDPC* codes, *DE* on *GE-LDPC* subgraphs which is of more concern and interest will be discussed in the next section.

8.1.2 DE on GE-LDPC codes

As noted earlier, the main part of this work has been done by Eckford *et al.* in [9] and [10]. As can be seen in the Figure 8.15, in *GE-LDPC* subgraph, channel factor nodes are attached to symbol variable nodes. This is slightly different from our *GE-LT* subgraph which will be of our more future interest in this work. In fact in *GE-LT* subgraph, channel factor nodes are not connected to symbol variable nodes but to parity-check nodes of the *LT* subgraph. Here, we note the *DE* procedure on *GE-LDPC* subgraph which is then applicable to *GE-LT* subgraph with some slight manipulations. Of all the 4 added messages described in Section 8, only the extrinsic message D is subject to modification, which will be explained later in the correspondent section. Taking P and Q to be the messages leaving symbol variable nodes and parity-check nodes respectively, the regular *LDPC SPA* updating rule for P and Q is outlined as before,

$$\begin{aligned} f_{P,j+1} &= \mathscr{F}^{-1}\left[\mathscr{F}(f_{P,0})(\mathscr{F}(f_{Q,j}))^{d_v-1}\right] \\ \tanh\left(\frac{Q}{2}\right) &= \prod_{i=1}^{d_c-1}\tanh\left(\frac{P_i}{2}\right). \end{aligned} \tag{8.139}$$

$\mathscr{F}$ in (8.139) represents the Fourier transform whereas index j indicates the iteration number. The former equation of (8.139) indicates the Fourier transform of the second equality in (5.79),

$$f_{D,j} = \mathscr{F}^{-1}\left[(\mathscr{F}(f_{Q,j}))^{d_v}\right] \tag{8.140}$$

says that D at every iteration can be obtained through summation of all the incoming edges to that specific symbol variable node. It is noteworthy that the messages **A** and **B** are two component vectors. For instance, the first component of **B** shows the probability of currently being in the **Good** state and the second component shows the probability of laying in the **Bad** state. In the following relations, $f_{A_1,j+1}$ represents the **Pdf** of the Good state. With doing some manipulations on **A** in (8.133), it is easy to see that

$$A_1 = \frac{N_1 + N_2 A_1^-}{D_1 + D_2 A_1^-}, \tag{8.141}$$

where

$$\begin{aligned}N_1 =& g(\eta_B + (1-2\eta_B)\gamma(D,Y))\\ N_2 =& (1-b)(\eta_G + (1-2\eta_G)\gamma(D,Y))\\ &- g(\eta_B + (1-2\eta_B)\gamma(D,Y))\\ D_1 =& \eta_B + (1-2\eta_B)\gamma(D,Y)\\ D_2 =& \eta_G + (1-2\eta_G)\gamma(D,Y)\\ &- \eta_B - (1-2\eta_B)\gamma(D,Y).\end{aligned} \tag{8.142}$$

B can also be similarly calculated. conditioning on the state s, we come up with

$$\begin{aligned}f(a_1^-|s) &= \frac{p(s|a_1^-)f(a_1^-)}{p(s)},\\ f(b_1^-|s) &= \frac{p(s|b_1^-)f(b_1^-)}{p(s)},\end{aligned} \tag{8.143}$$

where $f(a_1^-) = f_{A_1,j}$ and $p(s|a_1^-) = a_1^-$ in case $s = G$ and $p(s|a_1^-) = 1-a_1^-$ for $s = B$. For further notes on how the equalities in (8.143) have been derived, the interested reader is referred to [9]. Finally, obtaining $f(a_1|s,y,d)$ (which is more thoroughly expressed in [9]) as

$$\begin{aligned}f(a_1|s,y,d) = f&\left(\frac{D_1A_1 - N_1}{N_2 - D_2A_1}\middle| s\right)\\ &\cdot\left|\frac{D_1(N_2 - D_2A_1) + D_2(D_1A_1 - N_1)}{(N_2 - D_2A_1)^2}\right|,\end{aligned} \tag{8.144}$$

and marginalizing yields

$$f(a_1) = \sum_s \sum_y p(y|s)p(s). \int_d f(a_1|s,y,d)f(d)dd. \tag{8.145}$$

$f(b_1)$ is then obtained in the same manner. Following the above procedure, the channel message $\mathcal{C}$ is then derived and extracted as

$$f(C) = \sum_s \sum_y p(y|s)p(s). \int_{b_1} f(c|s,y,b_1)f(b_1|s)db_1, \tag{8.146}$$

which plays the role of P_0 in (8.139) at every iteration. Density of these messages is initialized as

$$\begin{aligned} & f_{A_1,0}(a_1) = \delta(a_1 - g/(b+g)) \\ & f_{B_1,0}(b_1) = \delta(b_1 - 1/2) \\ & f_{C,0}(c) = \bar{\eta}\delta(c + \log(1-\bar{\eta})/\bar{\eta}) \\ & + (1-\bar{\eta})\delta(c - \log(1-\bar{\eta})/\bar{\eta}) \\ & f_{P,0} = f_{C,0}, \end{aligned} \tag{8.147}$$

where $\bar{\eta}$ is the average flip probability and $\delta(\cdot)$ is the Dirac delta function. $f_{P,j+1}$ and $f_{D,j+1}$ are rephrased as

$$\begin{aligned} f_{P,j+1} &= \mathscr{F}^{-1}\left[\mathscr{F}(f_{C,j})\sum_{i=1}^{v_{max}}\lambda_i(\mathscr{F}(f_{Q,j}))^{i-1}\right], \\ f_{D,j+1} &= \mathscr{F}^{-1}\left[\sum_{i=1}^{v_{max}}\hat{\lambda}_i(\mathscr{F}(f_{Q,j}))^{i}\right], \end{aligned} \tag{8.148}$$

where λ_i is the probability that a randomly taken edge is connected to a symbol variable node of degree i. Similarly, $\hat{\lambda}_i$ is the probability that a randomly taken symbol variable node is of degree i.

8.1.3 SPA on GE-LDPC subgraphs

As noted earlier, the differenece between *GE-LDPC* and *GE-LT* subgraphs is that in *GE-LT* subgraphs, channel factor nodes are connected to symbol variable nodes. In *GE-LDPC* subgraph, as seen earlier, the channel factor nodes are connected to symbol variable nodes. *LT* check node of degree d would take on the updating rule of

$$D = 2\text{arctanh}(\prod_{j=1}^{d}\tanh(\frac{X_j}{2})), \tag{8.149}$$

as the extrinsic message sent from *LT* subgraph to *GE* subgraph. As it can be obviously seen, any channel message or observation *LLR* has been excluded from (8.149). The update equations under *SPA* are described as follows:

$$
\begin{aligned}
&\tanh\left(\frac{m_{o,i}^{(l)}}{2}\right) = \tanh\left(\frac{\xi}{2}\right) . \prod_{i' \neq i} \tanh\left(\frac{m_{i',o}^{(l)}}{2}\right) \\
&m_{i,o}^{(l+1)} = \sum_{o' \neq o} m_{o',i}^{(l)} \\
&\alpha(s_{i+1}) = \sum_{s_i \in S} p(s_{i+1}|s_i)\alpha(s_i) \sum_{x_i \in \{0,1\}} p(x_i|\chi)p(y_i|s_i, x_i) \\
&\beta(s_i) = \sum_{s_{i+1} \in S} p(s_{i+1}|s_i)\beta(s_{i+1}) \sum_{x_{i+1} \in \{0,1\}} p(x_{i+1}|\chi)p(y_{i+1}|s_{i+1}, x_{i+1}) \\
&\xi = \log \frac{p(x_i = 0|\alpha, \beta)}{p(x_i = 1|\alpha, \beta)}, \text{where} \\
&\\
&p(x_i = 0|\alpha, \beta) = \sum_{s_i \in S} \sum_{s_{i+1} \in S} p(y_i|s_i, x_i = 0)p(s_{i+1}|s_i)\alpha(s_i)\beta(s_{i+1}) \text{ and} \\
&p(x_i = 0|\alpha, \beta) = 1 - p(x_i = 1|\alpha, \beta) \\
&D = 2\text{arctanh}\left(\prod_{j=1}^{d} \tanh\left(\frac{m_{i',o}^{(l)}}{2}\right)\right), \\
&p(x|\chi) \text{ being the probabilistic form of } \chi : \\
&p(x|\chi) = \begin{cases} \frac{1}{2} + \frac{1}{2}\tanh\frac{\chi}{2}, & x = 0; \\ \frac{1}{2} - \frac{1}{2}\tanh\frac{\chi}{2}, & x = 1; \end{cases} \\
&\text{and} \\
&\alpha : S \to [0,1] \text{ and } \beta \to [0,1] \cdot
\end{aligned}
\tag{8.150}
$$

8.2 Applying the BSC design on the GE channels

In the Section 6.1, a particular (b, g, P_B, P_G) *GE* channel was exemplified. Some brief explanation was also given on how to calculate the capacity of such a channel according to [13]. Although we did not come up with any linear programming solution like in Section 7.4 for the *GE-LT* subgraphs, it is still an interesting approach to design a scheme for *BSC* of the same capacity and do some comparison. The comparison is to find out the amount of extra overhead needed comparing to *BSC* to lower the decoding error down to the error floor achieved when the sources were correlated through the noted *BSC*. The other motivating reason for such an approach is that *SPA* has not been tested on *GE-LT* graph illustrated in the Figure 8.17 before. Moreover, the *GE* channel discussed at the end of the Section 6.1 is taken on as the correlation channel between sources X and Y. Then having $\mathbf{x}$ intactly recovered as the side information, we would step into decoding $\mathbf{y}$. As noted earlier, from the capacity point of view the $(0.01, 0.01, 0.22, 0.01)$ *GE* channel is equivalent to $BSC(0.094)$. To get a better insight of this equivalence, it is noteworthy that the average inversion probability of the considered *GE* is 0.1105. Output node degree distributions $\Omega^{(1)}(x)$ and $\Omega^{(2)}(x)$ are obtained

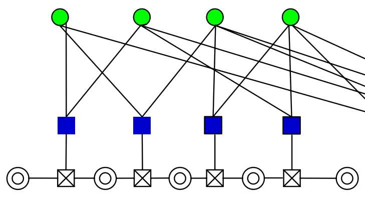

Figure 8.17: *GE-LT graph. Input nodes are depicted with squares whereas intermediate nodes and payload bits are illustrated with circles and triangles respectively. lower row squares and circles show channel factor nodes and state variable nodes.*

by linear programming and *DE* described in Section 7.3. Again we look for the suitable average node degrees α_1 and α_2 that provide us with better results on *BSC(ϵ)*, $\epsilon = 0.094$. In other words we seek a design on the scheme depicted in Figure 7.6 that works fine with the shortest payload overhead. $\alpha_1 = 3$ and $\alpha_2 = 4.05$ are set, as long as such an assignment of α_2 yields the same β_2 as in Section 7.7.3,

$$\begin{aligned}\Omega^{(1)}(x) &= 0.45x^2 + 0.5379x^3 + 0.0055x^5 + 0.0066x^{70} \\ \Omega^{(2)}(x) &= 0.0615x + 0.4462x^9 + 0.4922x^{10}.\end{aligned} \tag{8.151}$$

8.2.1 Simulation

Simulation was done on *BSC(0.094)* with the distributions in (8.151) and 8000 $\mathbf{y}$ input symbols to be decoded when we are given with $\mathbf{x}$ as the side information. A simple information analysis reveals that

$$\begin{aligned} n(1-h(\epsilon)) + m &= n \\ m = nh(\epsilon) &\simeq 3600, \end{aligned} \tag{8.152}$$

where n and m indicate source length and payload length respectively. Finally, 4200 bits were collected to lower the error down to the floor of 0.003 that implies the overhead of 0.167%. Then, the same distributions as in (8.151) is applied on the scheme of the Figure 8.17 and parameters of $(0.01, 0.01, 0.22, 0.01)$ to compare the *BER* versus overhead with that of *BSC(0.094)* and the scheme of the Figure 7.6.

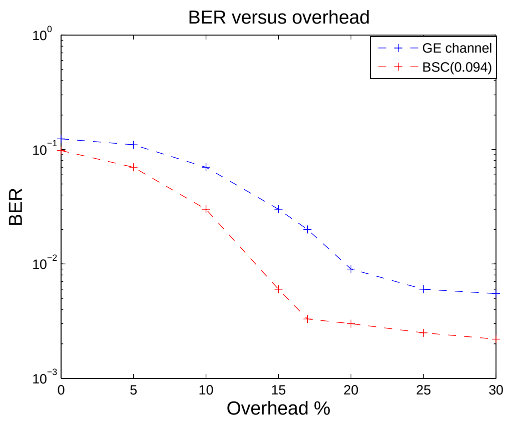

Figure 8.18: *Comparison of BER with respect to overhead for the cases BSC(0.094) and GE(0.01,0.01,0.22,0.01) is illustrated in the figure.*

Again, the raptor code approach could have probably lowered the error down to zero. This shows that not only decoding on the proposed Markov channel requires more overhead to reach its floor, but the floor is also higher comparing to that of the *BSC* for which the code has been designed. This arises the necessity of a Fountain code design on Markov channels. A raptor code design could again be the solution to this problem although it has not been addressed in this work.

9 Conclusion

This report was mostly devoted to design of classical LT and raptor codes for addressing the *Slepian-Wolf* problem. Design was done on sources which are correlated by a Binary Symmetric Channel(BSC). We did not manage to propose a novel fountain coding scheme for sources correlated by Gilbert-Elliot(GE) channel although it was of much interest. Instead, the scheme for memoryless sources was tested on the GE case. The concept of the work for source coding in 7th chapter is expandable to channel and joint source-channel coding too, as seen in the report. In other words, the second source symbols can be thought of being the first source symbols which are sent over a channel. Hence, the design is easily applicable on BSCs. Future work can focus on channel code design for Markov channels.

Bibliography

[1] Thomas M. Cover, Joy A. Thomas,“Elements Of Information Theory”, *Wiley Series in Telecommunications, First Edition, 1991*, **ISBN** 0-471-06259-6

[2] W.Cary Huffman, Vera Pless,“Fundamentals of Error-Correcting Codes”, *Cambridge, First Edition, 2003*, **ISBN** 0-521-78280-5

[3] WIKIPEDIA, *The Free Encyclopedia*

[4] Michael Luby, “LT codes” *, The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002

[5] A.Shokrollahi, “Raptor codes” *IEEE Transactions on Information Theory*, vol. 52, pp. 2551-2567, 2006.

[6] T.J. Richardson and R.L. Urbanke, “The capacity of low-density parity-check codes under message-passing decoding”, *IEEE Trans. Inform. Theory,* vol.47, pp.599-618, Feb.2001.

[7] Bertrand Ndzana Ndzana, Amin Shokrollahi, “Fountain codes for the Slepian-Wolf Problem”

[8] Andrew W. Eckford and Wei Yu, “Density evolution for the simultaneous decoding of LDPC-Based Slepian-Wolf source codes”, *ISIT. 2005. Proceedings. International Symposium on Information Theory,* Issue 11

, 4-9 Sept. 2005, pp.1401 - 1405

[9] Andrew W.Eckford, Frank R. Kschischang and Subbarayan Pasupathy, “Analysis of Low-Density Parity-Check codes for the Gilbert-Elliot Channel” *IEEE Transactions on Information Theory, Volume 51, Issue 11, Nov. 2005*, pp.3872 - 3889

[10] Andrew W. Eckford, Frank R. Kschischang and Subbarayan Pasupathy, “On designing good LDPC codes for Markov channels” *IEEE Transactions on Information Theory,Volume 53, Issue 1, Jan. 2007*, pp.5 - 21

[11] , S.S.Pradhan, K.Ramchandran, “Distributed source coding using syndromes (DISCUS): design and construction”, *IEEE Transactions on Information Theory, Volume 49, Issue 3, Mar 2003* pp.626 - 643

[12] J.Garcia-Frias, “Decoding of low-density parity-check codes over finite-state binary Markov channels” *IEEE Transactions on Communications,Volume 52, Issue 11, Nov. 2004* pp. 1840 - 1843

[13] M.Mushkin and I. Bar-David, “Capacity and coding for the Gilbert-Elliot channels”, *IEEE Transactions on Information Theory, Volume 35, Issue 6, Nov 1989*, pp.1277 - 1290

[14] D.Divsalar, S.Dolinar, and F.Pollara. “Iterative turbo decoder analysis based on density evolution” *IEEE J. Select. Areas Commun.*, 19:891-907, May 2001.

[15] Omid Etesami, Amin Shokrollahi, “Raptor codes on binary memoryless symmetric channels”, *IEEE Transactions on Information Theory*, vol. 52, num. 5, 2006, p. 2033-2051

[16] Thomas J.Richardson, M.Amin Shokrollahi, Rüdiger L. Urbanke, “Design of capacity-approaching irregular low-density parity check codes”, *IEEE Transactions on Information Theory, Volume 47, Issue 2, Feb 2001* pp.619 - 637

[17] S.-Y. Chung, T.Richardson, and R. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation" *IEEE Transactions on Information Theory, Volume 47, Issue 2, Feb 2001* pp.657 - 670

[18] Qian Xu,V.Stankovic, Zixiang Xiong, "Distributed joint source-channel coding of video using raptor codes", *Data Compression Conference, 2005. Proceedings. Volume , Issue , 29-31 March 2005*, Page(s): 491

[19] M.Ardakani and F.R. Kschischang, "A more accurate one-dimensional analysis and design of LDPC codes", *IEEE Transactions on Communications, Volume 52, Issue 12, Dec. 2004*, pp. 2106 - 2114