SPATIALLY-COUPLED LDPC CODING FOR THE CORRELATED ERASURE CHANNEL

by

Mahmood Reza Alizadeh Ashrafi

B.S., Electrical and Electronics Engineering, University of Tabriz, 2009

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2012

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Assist. Prof. Ali Emre Pusane for his continuous support and caring, his patience, immense knowledge, and brilliant ideas. His guidance was the biggest help in the all time of the research and this thesis would not have been accomplished without his help.

I also wish to thank my jury committee members Prof. Hakan Deliç and Assist. Prof. Güneş Zeynep Karabulut Kurt for their valuable suggestions which have formed the final version of my thesis.

I thank all my fellow labmates in wireless communication laboratory (WCL) for creating a warm and friendly environment, their suggestions and discussions, and for all the fun we have had in the last two years. Also I thank my friend Oğuz Karaduman for the great time we spent together. I also have a special thank to Neda B. Marvasti for her support and great help throughout this research.

Last but not the least, I would like to deeply thank my family: my parents Hamid and Sousan, and my sisters Elnaz and Tannaz for their eternal love, support, and encouragement in my life.

This thesis has been supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under project number 111E276.

ABSTRACT

SPATIALLY-COUPLED LDPC CODING FOR THE CORRELATED ERASURE CHANNEL

In this thesis, spatially-coupled low density parity-check (LDPC) codes, which have been shown to exhibit the threshold saturation phenomenon over the binary erasure channel (BEC), are considered and a low-delay iterative decoding scheme, called window decoding, is explained. Code ensembles with minimal coupling are enumerated and by using density evolution analysis, code ensembles with the best performance over the BEC are determined and some practical design criteria for construction an efficient ensemble are presented. It has been shown that the performance of these codes decreases drastically over correlated erasure channels where burst of high-probable erasures may hit the transmitted codeword. In the literature, this challenge is faced by presenting additional code design rules to build a robust ensemble against bursts. However, this approach results in more constraints on the code, which can be contradictory to some other properties of the code ensemble. In this thesis, we try to confront the transmission over correlated erasure channel by designing a communication system employing a convolutional interleaver. Asymptotic and practical performances of such a system over the correlated erasure channel have been studied and it has been shown that using a convolutional interleaver results in considerable improvement in the performance of the code compared to non-interleaving schemes which are purely based on code design criteria. All in one, major contributions of this thesis are deriving some practical design criteria for constructing a good code ensemble and also introducing the convolutional interleaver based spatially-coupled coding scheme which not only considerably mitigates the effect of burst degradations, but also in contrast to its block counterpart, introduces a negligible amount of delay, which makes this system a very suitable candidate for next-generation wireless communication standards.

ÖZET

İLİNTİLİ SİLME KANALINA UYGUN BİR UZAMSAL BAĞLI LDPC KODLAMA TEKNİĞİ

Bu tezde, ikili silme kanalı (BEC) üzerinde yinelemeli kod çözme eşiği doyumu gösteren uzamsal bağlı düşük-yoğunluklu eşlik-denetim (LDPC) kodları incelenip düşük gecikmeye sahip, pencere-tabanlı bir yinelemeli kod çözücü kullanımı ele alınmıştır. Düşük bağlama miktarına sahip kod aileleri sayılarak numaralandırılmış ve yoğunluk gelişimi tekniği yardımıyla BEC üzerinde en iyi hata başarımına sahip olan kod aileleri belirlenmiştir. Elde edilen sonuçlara göre, yüksek hata başarımına sahip kodların tasarımı için pratik tasarım koşulları önerilmiştir. Uzamsal-bağlı kodların ilintili silme kanalı üzerindeki hata başarımının düşük olduğu gösterilmiş ve literatürde bu problemin çözümü için ortaya atılmış özel kod tasarım koşulları ele alınmıştır. Önerilen bu özel kod tasarımları yüksek başarım sağlamakla birlikte kod tasarımına büyük ölçüde sınırlamalar getirmekte ve kod tasarım işlemini güçleştirmektedir. Bu çalışmada özel kod tasarımları yerine katlamalı serpiştirici kullanan bir iletişim sistemi önerilmiştir. Önerilen bu iletişim sisteminin asimtotik ve sonlu uzunluklu hata başarımı ilintili silme kanalı üzerinde incelenmiş ve katlamalı serpiştirici kullanan bu sistemin literatürdeki çözüme göre çok daha yüksek hata başarımına sahip olduğu gösterilmiştir. Bunun yanısıra, önerilen katlamalı serpiştirici yapısı blok serpiştirici ile karşılaştırılmış ve katlamalı serpiştiricinin iletişim sistemine getirdiği çok düşük gecikme ile gelecek nesil telsiz iletişim standartları için iyi bir aday olduğu gösterilmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . xi
LIST OF SYMBOLS . . . xii
LIST OF ACRONYMS/ABBREVIATIONS . . . xiv
1. INTRODUCTION . . . 1
1.1. Digital Communication System . . . 1
1.2. Communication Channels . . . 3
1.3. Channel Coding . . . 5
1.3.1. Block Codes . . . 5
1.3.2. Convolutional Codes . . . 7
1.4. Channel Decoding . . . 9
1.4.1. Optimal Decoding . . . 9
1.4.2. Iterative Decoding . . . 10
1.5. Thesis Outline . . . 11
2. BASICS OF LDPC CODES . . . 12
2.1. LDPC Block Codes . . . 12
2.2. Iterative Decoding of LDPC Block Codes . . . 14
2.2.1. Density Evolution . . . 17
2.3. LDPC Convolutional Codes . . . 19
3. SPATIALLY-COUPLED LDPC CODES . . . 21
3.1. Protograph-Based SC Codes . . . 22
3.1.1. Counting All Possible Asymptotically $(J, 2J)$-regular Ensembles 25
3.2. Window Decoding . . . 27
3.3. Density Evolution Analysis for SC Codes . . . 29
4. A LOW-DELAY CONVOLUTIONAL-INTERLEAVER-BASED COMMUNICATION SYSTEM . . . 35

4.1. Convolutional Interleaver 35
4.2. Simulation Results 40
5. CONCLUSION 55
REFERENCES 57

LIST OF FIGURES

Figure 1.1. A communication system model. 2

Figure 1.2. Binary erasure channel. 3

Figure 1.3. Single burst channel. Highlighted symbols show erasures. 3

Figure 1.4. Binary symmetric channel. 4

Figure 1.5. A rate $R = 1/2$ convolutional encoder with memory 2. 8

Figure 2.1. Tanner graph for the $\mathbf{H}$ matrix defined in Equation 2.2. 13

Figure 3.1. Copy-and-permute-operation. 22

Figure 3.2. Decomposition of $\mathbf{B}$ into $\mathbf{B}_i$. 25

Figure 3.3. Number of all possible asymptotically $(3, 6)$-regular ensembles increases exponentially with m_s. 26

Figure 3.4. Illustration of WD (A) full scheme, (B) freeze scheme at the first three decoding instants with $W = 4$. 28

Figure 3.5. BER performance all possible SC code ensembles with $m_s = 1$, $M = 20$, $L = 100$, and $W = 6$ over the AWGN channel. 33

Figure 4.1. Block diagram of a low-delay convolutional interleaver based communication system for correlated erasure channels. 36

Figure 4.2. Schematic of the convolutional interleaver π_1 with $\Lambda = 3$ and $\Psi = 2$ at the first time instant. Initial values are set to zero. Output of this time instant will be $v_1, 0, 0$. 37

Figure 4.3. Schematic of the convolutional interleaver π_2 with $\Lambda = 3$ and $\Psi = 1$ at the first time instant. 39

Figure 4.4. Schematic of the convolutional interleaver π_3 with $\Lambda = 2$ and $\Psi = 2$ at the first time instant. 41

Figure 4.5. Density evolution results for non-interleaved data transmission over the correlated erasure channel for code ensembles B and D. Also provided are the window decoding thresholds of the code ensembles when erasures occur randomly. 43

Figure 4.6. Density evolution results for non-interleaved and interleaved data transmission over the correlated erasure channel for code ensembles B and D. Window decoding thresholds of these code ensembles over the BEC are also provided. 44

Figure 4.7. Density evolution results for interleaved data transmission over the correlated erasure channel for code ensemble B under different convolutional interleavers π_1, π_2, and π_3 and also a block interleaver. Window decoding threshold of the code ensemble B over the BEC is also provided. 45

Figure 4.8. Density evolution results for interleaved data, derived from code ensemble B, under three different convolutional interleavers π_1, π_2, and π_3 and also non-interleaved data transmission over the correlated erasure channel for code ensembles D. Window decoding thresholds of these code ensembles over the BEC are also provided. 46

Figure 4.9. BER performance of code ensembles B and D over BEC and SBC. 47

Figure 4.10. BER performance of code ensemble B under convolutional interleavers π_1, π_2, and π_3 over the SBC. 48

Figure 4.11. BER performance of code ensemble D with no interleaving scheme and code ensemble B under convolutional interleavers π_1, π_2, and π_3 over the SBC. 49

Figure 4.12. BER performance of codes ensembles B and D with no interleaving schemes and also code ensemble B under both convolutional and the block interleaver schemes over the SBC. 50

Figure 4.13. BER performance of code ensembles B and D over BEC and SBC (random burst position). 51

Figure 4.14. BER performance of code ensemble B under convolutional interleavers π_1, π_2, and π_3 over the SBC (random burst position). . . . 52

Figure 4.15. BER performance of code ensemble D with no interleaving scheme and code ensemble B under convolutional interleavers π_1, π_2, and π_3 over the SBC (random burst position). 53

Figure 4.16. BER performance of codes ensembles B and D with no interleaving schemes and also code ensemble B under both convolutional and the block interleaver schemes over the SBC (random burst position). 54

LIST OF TABLES

Table 2.1. The received messages from the check nodes and majority vote mechanism. 15

Table 2.2. Threshold values for (J, K)-regular LDPC block codes of code rate $R = 1/2$. 18

Table 3.1. Number of all possible asymptotically $(3, 6)$-regular ensembles for different code memories. 26

Table 3.2. ϵ^*_{blk} and ϵ^*_{wd} for all ensembles with single memory over the BEC. Maximum number of iterations is 10^3, $\delta = 10^{-6}$, $L = 40$, and $W = 4$. 31

Table 3.3. ϵ^*_{wd} of some ensembles with code memory of two which have greater thresholds than threshold of regular-$(3, 6)$ block LDCP code over the BEC for $W = 4$ and $W = 6$. Maximum number of iterations is 10^3, $\delta = 10^{-6}$, and $L = 40$. 32

Table 4.1. Delay profile of the convolutional interleaver π_1. 38

Table 4.2. Delay profile of the convolutional interleaver π_2. 39

Table 4.3. Delay profile of the convolutional interleaver π_3. 41

LIST OF SYMBOLS

b	Number of information bits in a codeword (convolutional codes)
$\mathbf{B}$	Base matrix of spatially-coupled codes
c	Total number of bits in a codeword (convolutional codes)
c_i	Check node i
D	Delay element
$\mathbf{G}$	Generator matrix
$\mathbf{H}$	Parity-check matrix
$\mathbf{H^T}$	Syndrome-former matrix
$\mathbf{I}$	Identity (unit) matrix
J	Degree of variable node
k	Number of information bits in a codeword
K	Degree of check node
l	Number of iterations
L	Repetition factor
m	Number of rows in the parity-check matrix
M	Expansion factor
m_s	Memory of the code
n	Total number of bits in a codeword (number of columns in the parity-check matrix)
N_0	One-sided power spectral density of the noise process
p	Message from variable node to check node
$\mathbf{P}$	Parity matrix
P_b	Bit error probability
P_B	Block error probability
q	Message from check node to variable node
$\mathbf{r}$	Binary received vector
R	Code rate
T	Delay time

T_s	Period of the code
$\mathbf{u}$	Binary information vector
$\hat{\mathbf{u}}$	Estimated binary information vector at the decoder
$\mathbf{v}$	Binary codeword vector
v_j	Variable node j
W	Window size
δ	Target erasure probability
ϵ	Erasure rate of the channel
ϵ_B	Burst erasure rate of the channel
ϵ^*_{blk}	Code threshold under block decoding
ϵ^*_{WD}	Code threshold under window decoding
Λ	Number of delay lines in the convolutional interleaver
π	Convolutional interleaver pattern
σ_n^2	Noise variance
Ψ	Difference in number of delay elements in two consecutive delay lines of the convolutional interleaver

LIST OF ACRONYMS/ABBREVIATIONS

AWGN	Additive White Gaussian Noise
BEC	Binary Erasure Channel
BER	Bit Error Rate
BP	Belief Propagation
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
clk	Clock
DE	Density Evolution
gcd	Greatest Common Divisor
iid	Independent, Identically Distributed
LDPC	Low-Density Parity-Check
LLR	Log-Likelihood Ratio
MAP	Maximum A-Posteriori
ML	Maximum Likelihood
SBC	Single Burst Channel
SC	Spatially Coupled
WD	Window Decoding

1. INTRODUCTION

A world without any type of digital communication device is unimaginable in our era and demand for digital data transmission and storage systems is increasing. An important characteristic of digital communication is to receive or store reliable information which requires a low error rate. To satisfy this low error requirement, application of channel coding techniques is inevitable. For several decades, researches have invested so much effort in finding appropriate channel encoding and decoding schemes which have a desirable performance while having a feasible complexity.

One of the most efficient error-correcting codes to date is the newest member of low-density parity-check (LDPC) codes, called spatially-coupled LDPC codes. Analyzing these codes and designing algorithms to improve their performance are the main purpose of this thesis.

Section 1.1 gives a general overview of digital communications. Section 1.2 describes three commonly-used discrete channels and basic concepts of channel coding are given in Section 1.3. In Section 1.4, basics of optimal and iterative decoding algorithms are discussed. Outline of this thesis is given in Section 1.5.

1.1. Digital Communication System

The aim of a digital communication system is to transmit digital information to the desired destination through a physical channel. A general model for a digital communication system is shown in Figure 1.1. Based on the concept of information theory by means of the mathematical representation in Shannon's phenomenal work [1], two different aspects of a digital communication system were brought to design and application foreground. One of these aspects is source coding which seeks the ultimate limits of data compression. Generally, the information to be transmitted has redundancy. However it is possible to optimally reduce this redundancy and transmit the compressed information to avoid waste of precious resources. The input data can be

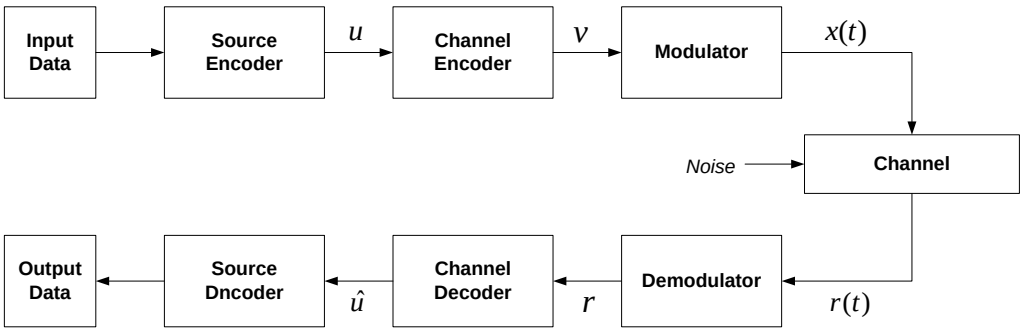

Figure 1.1. A communication system model.

either a continuous waveform or a sequence of discrete symbols. The source encoder converts the input data to a binary information sequence $\mathbf{u}$. Another fundamental result of Shannon's work leads to concept of channel coding, which looks for the ultimate rate of communication. Transmitting data over a physical channel may corrupt the data and therefore, the received data will not be exactly same as the original one. It has been shown that, for wide variety of channels, reliable communication where the original data can be retrieved from the received date, can only take place when the data rate is less than channel capacity. This is achievable by channel coding which adds redundant information to the original data in a controlled manner. The binary information $\mathbf{u} = [u_1, u_2 \ldots, u_k]$ is fed to channel encoding block and the output is the codeword $\mathbf{v} = [v_1, v_2, \ldots, v_n]$. The code rate R is defined as the ration of the number of information symbols k over the number of codeword symbols n as given by

$$R = \frac{k}{n}. \tag{1.1}$$

The aim of modulation is to carry digital information over sinusoidal pulses at higher frequencies. In this thesis, binary phase shift keying (BPSK) modulation has been used. In this modulation, each binary bit $v_i \in \{0, 1\}$ is modulated to $x(t) = A_i \cos \omega t$, where $A_i \in \{-1, 1\}$ and $\omega = 2\pi f$ in which f indicates the carrier frequency. The signal is then transmitted over a channel which can be a wired channel such as copper wire of telephones or optical fibers or a wireless channel such as cellular telephony, microwave, and satellite links. Errors may occur due to noise or channel characteristics. At this

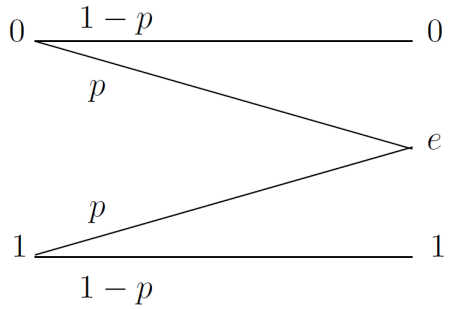

Figure 1.2. Binary erasure channel.

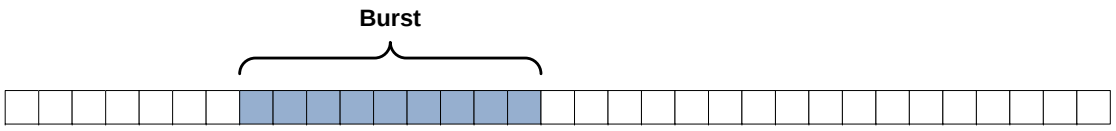

Figure 1.3. Single burst channel. Highlighted symbols show erasures.

point, the received signal $r(x)$ is fed to the communication blocks at the receiver side, which are designed to undo the processes performed at the transmitter side to retrieve the best estimate $\hat{\mathbf{u}}$ of the original data $\mathbf{u}$.

1.2. Communication Channels

In this section, some commonly-used channel models, binary erasure channel (BEC), single burst channel (SBC), binary symmetric channel (BSC), and additive white Gaussian noise (AWGN) channel are described.

Figure 1.2 shows the model for the BEC. A symbol input to this channel is either received correctly with probability $1 - p$ or the output is just an erasure ϵ with probability p. It is clear that BEC does not introduce any errors on the output data. Capacity of the BEC is $1 - p$ bit per use. In the binary SBC, similar to the binary BEC, symbols are either received correctly or erased. However, in SBC, some adjacent symbols are affected by high-probable erasures and these consecutive erasures create a

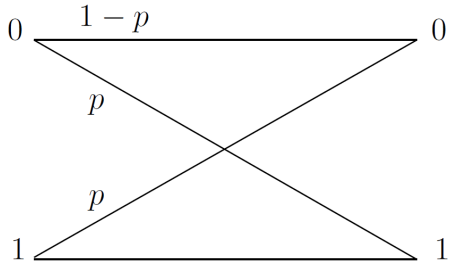

Figure 1.4. Binary symmetric channel.

burst. Figure 1.3 illustrates the SBC.

Figure 1.4 shows the model for the BSC. A symbol input to this channel can be received either correctly with probability $1-p$ or erroneously with probability p referred to as the crossover probability. Capacity of the BSC is calculated as $1-H(p)$ where $H(p)$ is the binary entropy function given as

$$H(p) = -p\log_2 p - (1-p)\log_2(1-p). \tag{1.2}$$

AWGN is modeled for a channel which the noise in it is made up of large number of independent identically distributed (iid) noise sources. In an AWGN channel, white Gaussian noise is added to the input data and output of the channel is real number obtained as

$$r_i = x_i + n_i, \quad i = 1, 2, \ldots, n. \tag{1.3}$$

Noise samples n_i are zero-mean Gaussian random variables with variance $\sigma_n^2 = N_0/2$, where N_0 is the single-sided power spectral density of the noise process. The capacity of discrete AWGN channel can be calculated numerically.

1.3. Channel Coding

As it was mentioned, channel codes (error-correcting codes) add redundant information to the original information symbols to make it possible to detect and correct erroneous symbols on the received symbols from the channel.

An intuitive coding scheme is *repetition coding* which simply repeats every information symbol L times [2]. For instance with $L = 3$, the information symbol sequence $\mathbf{u} = 101100$ is mapped to the codeword $\mathbf{v} = 111000111111000000$. The received symbols can be decoded based on majority count. If the number of a specific symbol v_i in each tuple of length L is greater than $L/2$, then it is decoded as v_i, i.e. the sequence 110 and 010 are decoded as 1 and 0, respectively. The code rate of repetition coding is

$$R = \frac{k}{n} = \frac{k}{kL} = \frac{1}{L}. \tag{1.4}$$

By increasing L, we can achieve a more reliable communication but it comes at the cost of low code rate and enormous waste of resources.

This example illustrates how redundancy enables error detecting and correcting mechanism. However, for practical applications we are intrusted in codes with high efficiency. Error-correcting codes can be categorized into two main classes; block codes and convolutional codes. In the rest of this chapter, a brief review on these classes of codes is presented.

1.3.1. Block Codes

A binary linear block code takes a block of binary information bits $\mathbf{u} = [u_1, u_2, \ldots, u_k]$ and outputs the binary codeword $\mathbf{v} = [v_1, v_2, \ldots, v_n]$. The one-to-one correspondence between $\mathbf{u}$ and $\mathbf{v}$ is set up by *generator* matrix $\mathbf{G}$ through

$$\mathbf{v}_{1\times n} = \mathbf{u}_{1\times k} \times \mathbf{G}_{k\times n}, \tag{1.5}$$

where matrix multiplication is carried out in modulo-2. Generator matrix $\mathbf{G}$ can be formed as

$$\mathbf{G} = [\mathbf{I}_k | \mathbf{P}_{k\times(n-k)}], \tag{1.6}$$

where $\mathbf{I}_k$ is the identity (unit) matrix of size k and $\mathbf{P}$ is the parity matrix. If in a codeword $\mathbf{v}$, information part can obviously be separable from the redundant part (e.g. the first k bits of total n bits of codeword correspond to the information bits), it is called a systematic code. $\mathbf{I}_k$ in the structure of the generator matrix $\mathbf{G}$ given in Equation (1.6) guarantees the code to be systematic.

Example: Let us assume a code with the generator matrix defined as below:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}. \tag{1.7}$$

This coding scheme takes a block of information bits of length 4 and maps them into a codeword of length 7. Thus, its code rate is $R = k/n = 4/7$. If $\mathbf{u} = [1\ 1\ 1\ 0]$, the codeword will be

$$\mathbf{v} = \mathbf{uG} = [1\ 1\ 1\ 0] \times \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = [1\ 1\ 1\ 0\ 1\ 0\ 0]. \tag{1.8}$$

This example shows a systematic code, where the first four bits of the codeword indicate the original information block.

The null space of the generator matrix $\mathbf{G}$ is called the *parity-check* matrix $\mathbf{H}$,

where transposed version $\mathbf{H^T}$, called the *syndrome-former* matrix satisfying

$$\mathbf{G} \times \mathbf{H^T} = \mathbf{0}. \tag{1.9}$$

Therefore,

$$\underbrace{\mathbf{u} \times \mathbf{G}}_{\mathbf{v}} \times \mathbf{H^T} = \mathbf{u} \times \mathbf{0} \;\; \Rightarrow \;\; \mathbf{v} \times \mathbf{H^T} = \mathbf{0}. \tag{1.10}$$

The parity-check matrix $\mathbf{H}$ has size $(n-k) \times n$. A binary parity-check matrix $\mathbf{H}$ can be formed by

$$\mathbf{H} = [\mathbf{P}^T_{k\times(n-k)} | \mathbf{I}_{n-k}]. \tag{1.11}$$

The parity-check matrix $\mathbf{H}$ of the generator matrix $\mathbf{G}$ given in the example is

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{1.12}$$

1.3.2. Convolutional Codes

Convolutional codes are alternative coding schemes to block codes. A binary convolutional encoder with structure based on m delay elements takes b bits of the data sequence and outputs c bits of codeword and this process repeats at each time instant. Codeword sequence depends on the current input bits and stored bits in delay elements, which belong to previous time instants. Input and output sequences of the convolutional encoder at time instant t can be written as $\mathbf{u_t} = [u_t^1, u_t^2, \ldots, u_t^b]$ and $\mathbf{v_t} = [v_t^1, v_t^2, \ldots, v_t^c]$, respectively. Code rate of a convolutional code is defined as $R = b/c$. Figure 1.5 illustrates a convolutional code with $b = 1$, $c = 2$ and $m = 2$. Codeword $\mathbf{v_t} = [v_t^1, v_t^2]$ of the convolutional encoder shown in Figure 1.5 is given as

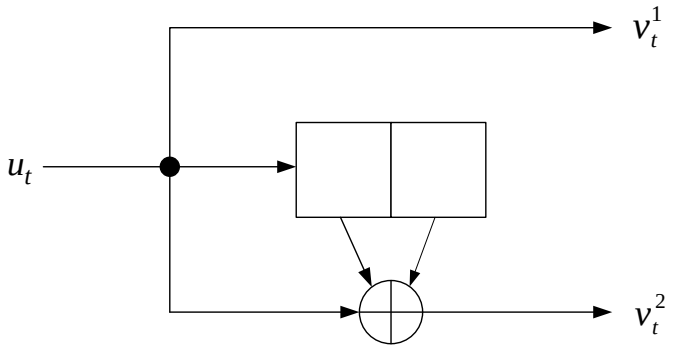

Figure 1.5. A rate $R = 1/2$ convolutional encoder with memory 2.

$$\begin{cases} v_t^1 = u_t, \\ v_t^2 = u_t + u_{t-1} + u_{t-2}. \end{cases} \tag{1.13}$$

Since the codeword given in Equation (1.13) is valid for any time instant t, this convolutional code is said to be time-invariant. The general formulation for convolutional encoder of rate $R = b/c$ can be described using the following notations. The binary information sequence $\mathbf{u}$ is given as

$$\mathbf{u} = [\ldots, \mathbf{u_0}, \mathbf{u_1}, \ldots, \mathbf{u_t}, \ldots], \tag{1.14}$$

where $\mathbf{u_i} = [u_i^1, u_i^2, \ldots, u_i^b]$. This binary information sequence is encoded to

$$\mathbf{v} = [\ldots, \mathbf{v_0}, \mathbf{v_1}, \ldots, \mathbf{v_t}, \ldots], \tag{1.15}$$

where $\mathbf{v_i} = [v_i^1, v_i^2, \ldots, v_i^c]$. The relation between $\mathbf{u}$ and $\mathbf{v}$ for a convolutional encoder without feedback can be written as

$$\mathbf{v} = \mathbf{u_t}\mathbf{G_0}(t) + \mathbf{u_{t-1}}\mathbf{G_1}(t) + \ldots + \mathbf{u_{t-m}}\mathbf{G_m}(t), \tag{1.16}$$

where $\mathbf{G_i}(t)$ is an $b \times c$ matrix. For a time-invariant convolutional code, $\mathbf{G_i}(t)$ can be replaced by $\mathbf{G_i}$. The generator matrix $\mathbf{G}$ of a time-invariant convolutional code has a

repetitive form of

$$\mathbf{G} = \begin{bmatrix} \ddots & \ddots & \ddots & & & & \\ & \mathbf{G_0} & \mathbf{G_1} & \dots & \mathbf{G_m} & & \\ & & \mathbf{G_0} & \mathbf{G_1} & \dots & \mathbf{G_m} & \\ & & & \mathbf{G_0} & \mathbf{G_1} & \dots & \mathbf{G_m} \\ & & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{1.17}$$

Generator matrix $\mathbf{G}$ of the convolutional code shown in Figure 1.5 is

$$\mathbf{G} = \begin{bmatrix} \ddots & \ddots & \ddots & & & & \\ & 11 & 01 & 01 & & & \\ & & 11 & 01 & 01 & & \\ & & & 11 & 01 & 01 & \\ & & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{1.18}$$

1.4. Channel Decoding

The aim of channel decoding is to remove redundancy out of the codeword $\mathbf{r}$ received from the discrete channel to compute an estimate $\hat{\mathbf{u}}$ of the original information sequence $\mathbf{u}$. It is obvious that $\mathbf{r}$ is not necessarily equal to $\mathbf{v}$. Consequently, $\hat{\mathbf{u}} = \mathbf{u}$ does not hold necessarily too. However, an optimal channel decoding scheme minimizes the error probability.

1.4.1. Optimal Decoding

The probability of incorrect decoding of information block is called *block error probability* P_B and is defined as

$$P_B = P(\hat{\mathbf{u}} \neq \mathbf{u}) = \sum_{\mathbf{r}} P(\hat{\mathbf{u}} \neq \mathbf{u}|\mathbf{r})P(\mathbf{r}). \tag{1.19}$$

To minimize P_B, $P(\hat{\mathbf{u}} \neq \mathbf{u}|\mathbf{r})$ should be minimized for each $\mathbf{r}$ (or $P(\hat{\mathbf{u}} = \mathbf{u}|\mathbf{r})$ should be maximized) since $P(\mathbf{r})$ is independent from decoding algorithm. Therefore, P_B is minimized by choosing $\hat{\mathbf{u}}$ such that

$$\hat{\mathbf{u}} = \arg\max_{u} P(\mathbf{u}|\mathbf{r}). \tag{1.20}$$

This decoding algorithm is called *maximum a-posteriori* (MAP). *Maximum likelihood* (ML) decoder minimizes P_B through setting $\hat{\mathbf{u}}$ to

$$\hat{\mathbf{u}} = \arg\max_{u} P(\mathbf{r}|\mathbf{u}). \tag{1.21}$$

There is a relation between ML and MAP decoding algorithms. If all information sequences $\mathbf{u}$ are equally likely, MAP becomes equivalent to ML. It is also possible to derive the same analysis for *bit error probability* P_b, defined as

$$P_b = P(\hat{u}_i \neq u_i). \tag{1.22}$$

In this case, the MAP decoding rule is written as

$$\hat{u}_i = \arg\max_{u \in \{0,1\}} P(u_i = u|\mathbf{r}). \tag{1.23}$$

1.4.2. Iterative Decoding

Optimal decoding becomes computationally infeasible for large information blocks. In such cases, it is preferred to use a suboptimal decoding algorithm to establish a reasonable trade-off between performance and complexity of the decoder. Iterative decoding is an efficient alternative. The basic idea behind it is to break up optimal decoding on a large codeword into several steps and to transfer information between these steps to prevent any information loss. The exchanged information can be either *hard* or *soft* decision values. However, soft information leads to better decoding performance. In

the next chapter, iterative decoding algorithm is discussed in more detail.

1.5. Thesis Outline

This thesis focuses on a special class of LDPC codes called spatially-coupled codes. In addition to calculating several analytical results on the performance of different spatially-coupled codes, we propose a low-delay communication system based on the combination of these codes and a convolutional interleaver which shows a superior performance over the correlated erasure channel. This thesis is organized as follows.

Chapter 2 provides an introduction on LDPC codes. It presents the concept of LDPC block and LDPC convolutional codes and their properties are examined in detail. Then, iterative algorithms such as bit flipping and belief propagation for decoding LDPC codes are explained. Density evolution analysis, which determines the asymptotic performance of a code family is also discussed in this chapter.

Chapter 3 is fully devoted to spatially-coupled codes. Different protograph-based spatially-coupled codes are enumerated and a low-delay iterative decoding algorithm called window decoding, which is suitable for the special structure of spatially-coupled codes, is discussed. The performance of the codes with minimal coupling by density evolution technique is calculated and code ensembles with the best performance are selected. Furthermore, some practical rules and criteria for a good code design are extracted based on those results.

The main contribution of this thesis comes in Chapter 4 in which a low-delay spatially-coupled communication system based on convolutional interleaver is proposed and its performance over the BEC and the correlated erasure channel is evaluated. It is shown that employing the convolutional interleaver increases the performance of the spatially-coupled communication system considerably while introducing a negligible amount of delay. Finally, the last chapter addresses several concluding remarks and some areas and suggestions for possible future works are also included.

2. BASICS OF LDPC CODES

Introduction of Turbo codes in 1993 [3] was an important breakthrough in the history of coding theory, which led people to expand their prospective searching for codes with better performances. LDPC codes were first introduced by Gallager in early 1960 [4]. However, the complexity of proposed decoding algorithm was beyond the capability of computers at that time and therefore, they had been forgotten for several decades until Mackay and Neal rediscovered them in 1996 [5] and since then they have attracted lots of research interest. It has been shown that these codes are able to operate within thousandth of decibels to Shannon limit over the AWGN channel [6].

LDPC codes' decoding is based on an iterative message-passing algorithm. The most common class of message-passing algorithm is called belief-propagation algorithm. In this chapter, a detailed overview on LDPC block codes and LDPC convolutional codes is given and their characteristics and decoding algorithm are discussed.

2.1. LDPC Block Codes

Just like any other type of linear block codes, an LDPC block code of rate $R = k/n$ can be fully defined by the parity-check matrix $\mathbf{H}$ which has n columns and $m \geq n-k$ rows as there may be some linearly-dependent rows. However, this parity-check matrix $\mathbf{H}$ has low density. It means that in a binary LDPC code, number of 1s in the parity-check matrix $\mathbf{H}$ is very small compared to number of 0s. Another condition for sparsity of parity-check matrix $\mathbf{H}$ is that the number of 1s should be very small compared to m and n too.

There are two types of LDPC block codes: regular and irregular codes. In regular codes, number of 1s is constant in each row and column of parity-check matrix $\mathbf{H}$ while in irregular codes, this number can vary for different rows and columns. An LDPC block code is said to be (J, K)-regular if there are exactly K 1s in each row and J 1s in each column of the parity-check matrix $\mathbf{H}$. J and K are called column and row weight,

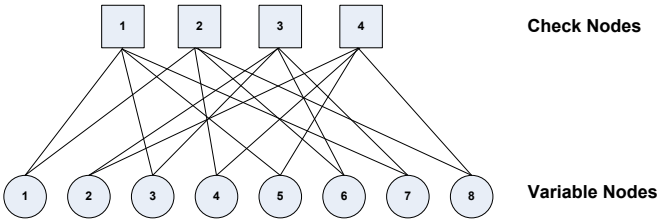

Figure 2.1. Tanner graph for the **H** matrix defined in Equation 2.2.

respectively. For such codes, we can verify that

$$m \times K = n \times J. \tag{2.1}$$

The parity-check matrix **H** of a $(2, 4)$-regular LDPC block code can be given as

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

As it can be seen, there are 2 ones in each column and 4 ones in each row of the parity-check matrix **H**. The rate of this code is $R = 4/8 = 1/2$.

An LDPC block code can be illustrated by means of a graphical representation called the *Tanner* graph which sets a convenient basis for iterative decoding algorithm. A Tanner graph is a bipartite graph with two types of nodes: variable (bit) nodes and check nodes [7]. A variable node corresponds to the components of codewords and check node represents the set of parity-check equations.

There is a one-to-one correspondence between parity-check matrix **H** and the Tanner graph representation. If the variable node v_j $(j = 1, 2, \ldots, n)$ is connected to the check node c_i $(i = 1, 2, \ldots, m)$, then $H_{ij} = 1$, otherwise $H_{ij} = 0$. Figure 2.1 shows the Tanner graph of the parity-check matrix **H** given in Equation (2.2).

As it was mentioned, there is a relation between parity-check matrix **H** and gen-

erator matrix **G** (Equation (1.6) and Equation (1.11)). For encoding process, it is possible to extract generator matrix **G** from desired parity-check matrix **H** by performing Gaussian elimination [8]. However, in this thesis, we focus mainly on the decoding problem.

2.2. Iterative Decoding of LDPC Block Codes

Sparse structure of LDPC codes enables an efficient iterative decoding to be performed on them. Iterative decoding is based on message-passing algorithm. As the name implies, certain messages are exchanged iteratively along the connecting edges between variable nodes and check nodes in the Tanner graph and at each round of iteration, messages are updated. These messages can be either hard-decision information such as binary values of 0 and 1 or soft information such as probabilistic values P_0 and P_1, which indicate probability of the certain bit being 0 and 1, respectively. Log-likelihood ratio (LLR) is also a suitable soft information which is defined as

$$LLR(v_i) = \log \frac{P(v_i = 0)}{P(v_i = 1)}, \quad i = 1, 2, \ldots, n. \tag{2.3}$$

There are different algorithms based on message-passing including bit-flipping algorithm [9], belief-propagation (BP) [10], and min-sum which is the simplified version of BP [11]. Bit-flipping algorithm is based on hard-decision messages which results in less effective performance. Because of the trade-off between simplicity and performance of the algorithm, min-sum algorithm performs weaker than BP. For achieving more efficient results, in this thesis, BP algorithm has been used. We begin with an example of bit-flipping algorithm with hard decision information to illustrate how message-passing algorithm works. Then, BP algorithm is discussed in detail.

Example: Assume an LDPC block code with the parity-check matrix **H** given in Equation (2.2). The codeword $\mathbf{v} = [1\ 0\ 0\ 1\ 0\ 1\ 1\ 1]$ is generated and transmitted over the BSC. Let the received codeword be $\mathbf{r} = [1\ 0\ 0\ 1\ 0\ \underline{0}\ 1\ 1]$ with an error at the sixth position. Iterative message-passing algorithm initializes variable nodes with the correspondent bits of the received codeword r. Then, at the first step, variable node

Table 2.1. The received messages from the check nodes and majority vote mechanism.

v_1(received, c_1, c_2)	$1, 1, 0$	1
v_2(received, c_3, c_4)	$0, 1, 0$	0
v_3(received, c_1, c_3)	$0, 0, 1$	0
v_4(received, c_2, c_4)	$1, 0, 1$	1
v_5(received, c_1, c_4)	$0, 0, 0$	0
v_6(received, c_2, c_3)	$0, 1, 1$	1
v_7(received, c_1, c_3)	$1, 1, 0$	1
v_8(received, c_2, c_4)	$1, 0, 1$	1

v_j sends its value to the connected check nodes. For instance, v_1 sends 1 to c_1 and c_2, v_2 sends 0 to c_3 and c_4, etc. Thus, each check node c_i receives four message from four different connected variable nodes. At the second step, the check node c_i calculates the response for each connected variable node v_j by performing parity-check equations (in modulo-2) assuming that the received bits from the other three variable nodes are correct. More clearly, c_1 receives 1, 0, 0, 1 from v_1, v_3, v_5, v_7, respectively. Therefore, c_1 calculates the responds for these variable nodes as

$$c_1 \to v_1 : \; v_1 + 0 + 0 + 1 = 0 \quad \Rightarrow v_1 = 1. \tag{2.4}$$

$$c_1 \to v_3 : \; 1 + v_3 + 0 + 1 = 0 \quad \Rightarrow v_3 = 0. \tag{2.5}$$

$$c_1 \to v_5 : \; 1 + 0 + v_5 + 1 = 0 \quad \Rightarrow v_5 = 0. \tag{2.6}$$

$$c_1 \to v_7 : \; 1 + 0 + 0 + v_7 = 0 \quad \Rightarrow v_7 = 1. \tag{2.7}$$

These responds are sent to the variable nodes. Now, each variable node v_j has three sources of information, two suggested responses from its connected check nodes and also the original received bit from the channel. Based on the majority vote, the most probable value for the variable node is decided. Table 2.1 shows this procedure. As it can be seen, the sixth bit is flipped back to 1 and the correct codeword is recovered which satisfies parity-check equations. If all parity-check equations are not fulfilled, variable nodes send their updated values back to the check nodes and this iterative

process continues until all parity-check equations are satisfied or the maximum number of iterations is reached.

Topology of the Tanner graph is the key factor for message-passing-based decoding. A Tanner graph with a closed path (cycle) between some of variable nodes leads to failure of message-passing decoding. There several problematic concepts such as stopping set, trapping set, absorbing sets, and etc., which have to be taken into consideration to result a code with high performance.

This example illustrates how bit-flipping algorithm based on hard-decision works. However, this algorithm is not suitable for practical applications. The most common message-passing algorithm is the BP algorithm which is also known as the sum-product algorithm. The basic structure of the BP algorithm is the same as the bit-flipping algorithm, where messages are exchanged between variable and check nodes along the connecting edges of the Tanner graph representation iteratively. However, in BP, soft information (LLR values) are being exchanged. Similarly, there are two types of exchanged messages: a message from check node c_i to variable node v_j (q_{ij}) and the message from variable node v_j to check node c_i (r_{ji}). These messages can be calculated as

$$q_{ij} = \log \frac{1 + \prod_{j' \in N(i)\backslash j} \tanh(r_{j'i})}{1 - \prod_{j' \in N(i)\backslash j} \tanh(r_{j'i})}, \quad j \in N(i). \tag{2.8}$$

$$r_{ji} = LLR_j + \sum_{i' \in N(j)\backslash i} q_{i'j}, \qquad i \in N(j). \tag{2.9}$$

In Equation (2.8) and Equation (2.9), $N(i)$ and $N(j)$ denote the neighbors of the node i and j, respectively. Symbol "\" denotes the exclusion operation. For instance, $N(j)\backslash i$ means all neighbors of node j excluding node i. Initially, variable nodes send their received LLR values from the discrete channel to the connected check nodes, meaning that messages coming from check nodes are assumed to be zero. Then at each round of iteration, every check node collects messages from its connected variable

nodes to calculate the response q_{ij} according to the Equation (2.8). Based on the received messages from different connected check nodes, each variable node calculates another message r_{ji} according to Equation (2.9) and sends it back to the interacted check nodes. This procedure takes place iteratively until the maximum number of iterations is reached. At this point, binary values of variable nodes (which indicate the bits of the codeword) are extracted by performing thresholding on the resulting LLR values.

2.2.1. Density Evolution

We are interested in estimating the asymptotic performance limit (threshold) of the LDPC codes. It is very useful because even before examining the decoding performance of the code, we can foretell how good a code ensemble can perform. In other words, based on the threshold of the code ensemble, we can decide whether a member code is suitable for some certain applications or not.

Density Evolution (DE), introduced in [12] is a powerful analytical technique which calculates the asymptotic performance of a code ensemble by tracking the probability distribution of the exchanged messages between nodes in the message-passing algorithm. Urbanke and Richardson in [12] showed that the performance of a (J, K)-regular LDPC block code for large code lengths converges to the performance of the (J, K)-regular LDPC block code with the best performance, whose Tanner graph is a tree.

In this thesis, we concentrate on DE analysis over the BEC. Binary information received from BEC are either correct or erased. If a bit is received correctly, then LLR= ∞ and if the bit is erased, LLR= 0. The check node c_i sends ∞ to the variable node v_j if and only if all the messages from connected variable nodes excluding v_j itself are ∞. The variable node v_j sends ∞ to the check node c_i if and only if v_j is not erased or at least one of the other interacted check nodes with v_j sends ∞. Let $q_{i\to j}^{(l)}$ be the probability that a message from check node c_i to variable node v_j is an erasure at the l^{th} iteration round. Similarly, $p_{j\to i}^{(l)}$ represents the erasure probability of a message

Table 2.2. Threshold values for (J, K)-regular LDPC block codes of code rate $R = 1/2$.

(J, K)	Threshold
$(2, 4)$	0.1666
$(3, 6)$	0.4294
$(4, 8)$	0.3834
$(5, 10)$	0.3415
$(6, 12)$	0.3074

from variable node v_j to check node c_i at the l^{th} iteration round. The variable-to-check messages are initialized using $p_{j\to i}^{(0)} = \epsilon$, where ϵ represents the channel erasure rate. The probabilities are then iteratively updated using

$$q_{i\to j}^{(l)} = 1 - \prod_{j' \in V(i)\backslash j} (1 - p_{j'\to i}^{(l-1)}). \tag{2.10}$$

$$p_{j\to i}^{(l)} = \epsilon \cdot \prod_{i' \in C(j)\backslash i} (q_{i'\to j}^{(l)}). \tag{2.11}$$

For the (J, K)-regular LDPC code, Equation (2.10) and Equation (2.11) can be simplified as

$$q^{(l)} = 1 - (1 - p^{(l-1)})^{K-1}. \tag{2.12}$$

$$p^{(l)} = \epsilon \cdot (q^{(l)})^{J-1} = \epsilon \cdot [1 - (1 - p^{(l-1)})^{K-1}]^{J-1}. \tag{2.13}$$

Threshold of the (J, K)-regular LDPC block code is a value for ϵ which make $p^{(l)}$ converges to zero. Note that, besides the values of J and K, the only parameter which affects threshold value is the number of iterations l. The larger values for l result in higher thresholds.

Table 2.2 shows threshold values for different (J, K)-regular LDPC block codes of code rate $R = 1/2$ with $l = 1000$.

2.3. LDPC Convolutional Codes

Convolutional counterparts of LDPC block codes are called LDPC convolutional codes which first were introduced in [13]. Similar to LDPC block codes, LDPC convolutional codes can also be described by their sparse parity-check matrices which allow them to be also decoded by iterative message-passing algorithm [14]. These codes have been shown to be very suitable for applications where continuous data transmission with variable frame size is of interest [15].

A rate $R = b/c$ time-varying LCPC convolutional code is defined as the set of semi-infinite binary row vectors $\mathbf{v}_\infty$, satisfying $\mathbf{v}_\infty \cdot {\mathbf{H}_\infty}^T = \mathbf{0}_\infty$ [16], where $\mathbf{H}_\infty$ is the parity-check matrix and $\mathbf{0}_\infty$ is the semi-infinite all-zero row vector. $\mathbf{H}_\infty$ has a form as

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_0(1) & & & & & \\ \mathbf{H}_1(1) & \mathbf{H}_0(2) & & & & \\ \vdots & \mathbf{H}_1(2) & \ddots & & & \\ \mathbf{H}_{m_s}(1) & \vdots & \ddots & \mathbf{H}_0(t) & & \\ & \mathbf{H}_{m_s}(2) & \ddots & \mathbf{H}_1(t) & \ddots & \\ & & & \vdots & \ddots & \\ & & & \mathbf{H}_{m_s}(t) & \ddots & \end{bmatrix}. \tag{2.14}$$

Parameter m_s is called the memory of the code. Each element $\mathbf{H}_i(t), \;\; i = 1, 2, \ldots, m_S$ represents a $(c-b) \times c$ binary matrix which should satisfy three conditions [17] given as

- $\mathbf{H}_i(t) = \mathbf{0}$ for $i < 0$ and $i > m_s$.
- $\exists\, t > 0$ such that $\mathbf{H}_{m_s}(t) > 0$.
- $\mathbf{H}_0(t)$ has full rank.

The first two conditions guarantee that the code has a memory and the last one ensures that the parity-check matrix $\mathbf{H}$ of the code is full rank (useful in encoding process).

Constraint length ν_s is defined as $\nu_s = c \cdot (m_s + 1)$. The parity-check matrix $\mathbf{H}$ of an LDPC convolutional code is said to be periodic with period T_s if

$$\mathbf{H}_i(t) = \mathbf{H}_i(t + T_s), \quad i = 1, 2, \ldots, m_s. \tag{2.15}$$

If the period $T_s = 1$, the code is called time-invariant and its parity-check matrix can be written as [18]

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_0 & & & & & \\ \mathbf{H}_1 & \mathbf{H}_0 & & & & \\ \vdots & \mathbf{H}_1 & \ddots & & & \\ \mathbf{H}_{m_s} & \vdots & \ddots & \mathbf{H}_0 & & \\ & \mathbf{H}_{m_s} & \ddots & \mathbf{H}_1 & \ddots & \\ & & & \vdots & \ddots & \\ & & & \mathbf{H}_{m_s} & \ddots & \end{bmatrix}. \tag{2.16}$$

3. SPATIALLY-COUPLED LDPC CODES

The newest members of LDPC code family that this chapter focuses on are called *spatially-coupled* LDPC codes or simply spatially-coupled (SC) codes. SC codes which are in fact terminated versions of LDPC convolutional codes, have gained considerable research interest during the recent couple of years [19–21]. The parity-check matrix $\mathbf{H}$ of the SC code has the form of

$$\mathbf{H}_{[L]} = \begin{bmatrix} \mathbf{H}_0(1) & & & \\ \mathbf{H}_1(1) & \mathbf{H}_0(2) & & \\ \vdots & \mathbf{H}_1(2) & \ddots & \\ \mathbf{H}_{m_s}(1) & \vdots & \ddots & \mathbf{H}_0(L) \\ & \mathbf{H}_{m_s}(2) & \ddots & \mathbf{H}_1(L) \\ & & & \vdots \\ & & & \mathbf{H}_{m_s}(L) \end{bmatrix}, \tag{3.1}$$

which is a representation of an LDPC convolutional code which has been terminated after L time instants and has a finite length.

As a special case of LDPC convolutional codes, SC codes inherit very good characteristics, including simple code construction property and encoding and decoding simplicity. Furthermore, it has been proven that SC codes exhibit a "*threshold saturation*" phenomenon, which results in maximum-likelihood (ML) performance over BEC [22]; a result may be expanded to other types of channels.

In this chapter, we focus on the protograph-based SC codes. Ensembles with different code memories are enumerated and the structure of the low-delay window decoding for these codes are explained. The performance of the SC code ensembles with minimal coupling is analyzed by DE and demonstrated vis computer simulations.

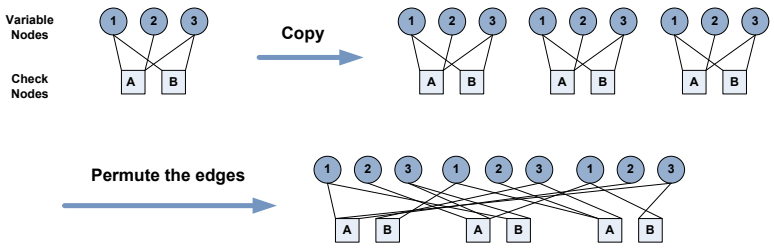

Figure 3.1. Copy-and-permute-operation.

3.1. Protograph-Based SC Codes

A protograph is a relatively small bipartite graph from which larger graphs are derived by the copy-and-permute procedure [23]. First, the protograph is copied M times, and then the edges of the individual copies are permuted to obtain a single, large bipartite graph, referred to as the derived graph. The parameter M is called the expansion factor. If the protograph has N_P variable nodes and M_P check nodes, then the resulting graph consists of $n = N_P \times M$ variable nodes and $m = m_P \times M$ check nodes. Figure 3.1 demonstrates the copy-and-permute operation for a simple example with $N_P = 3$, $M_P = 2$, and $M = 3$.

A protograph can be represented by an $M_P \times N_P$ biadjacency matrix, $\mathbf{B}$, which is called the base matrix of a protograph. Each entry in the base matrix, $\mathbf{B}_{i,j}$, $i = 1, \ldots, M_P$, $j = 1, \ldots, N_P$, is the number of edges between c_i and v_j. This is a non-negative integer and since parallel edges in protographs are allowed, it can be greater than one. Although parallel edges are allowed in the base matrix, they are not allowed in the resulting binary parity-check matrix obtained by the copy-and-permute operation. SC codes can be constructed using protographs with base matrices in the

form

$$\mathbf{B}_{[L]} = \overbrace{\begin{bmatrix} \mathbf{B}_0 & & & \\ \mathbf{B}_1 & \mathbf{B}_0 & & \\ \vdots & \mathbf{B}_1 & \ddots & \mathbf{B}_0 \\ \mathbf{B}_{m_\mathrm{s}} & \vdots & \ddots & \mathbf{B}_1 \\ & \mathbf{B}_{m_\mathrm{s}} & \ddots & \vdots \\ & & & \mathbf{B}_{m_\mathrm{s}} \end{bmatrix}}^{L}, \tag{3.2}$$

where m_s is the code memory, each entry is a $J' \times K'$ submatrix, and for an asymptotically (J, K)-regular SC code family, the parameters J' and K' are obtained using

$$J' = J/\gcd(J, K), \tag{3.3}$$

$$K' = K/\gcd(J, K). \tag{3.4}$$

The corresponding protograph has $N_P = LK'$ variable nodes and $M_P = (L + m_s)J'$ check nodes. Thus, the rate of the associated SC code can be defined as

$$R_L = 1 - \frac{L + m_\mathrm{s}}{L}\frac{J'}{K'} = 1 - (1 + \frac{m_\mathrm{s}}{L})(1 - R), \tag{3.5}$$

where $R = 1 - (J'/K')$ is the rate of unterminated LDPC convolutional code [24].

An example for the base matrix with $\mathbf{B}_0 = [2\ \ 2]$ and $\mathbf{B}_1 = [1\ \ 1]$ can be given as

$$\mathbf{B}_{[L]} = \begin{bmatrix} 2 & 2 & & & & & & & & \\ 1 & 1 & 2 & 2 & & & & & & \\ & & 1 & 1 & 2 & 2 & & & & \\ & & & & & \ddots & & & & \\ & & & & & & \ddots & & & \\ & & & & & & 1 & 1 & 2 & 2 \\ & & & & & & & & 1 & 1 \end{bmatrix}. \tag{3.6}$$

Another example with $\mathbf{B}_0 = [3\ \ 3]$ and $\mathbf{B}_1 = [0\ \ 0]$ is

$$\mathbf{B}_{[L]} = \begin{bmatrix} 3 & 3 & & & & & & & & \\ 0 & 0 & 3 & 3 & & & & & & \\ & & 0 & 0 & 3 & 3 & & & & \\ & & & & & \ddots & & & & \\ & & & & & & \ddots & & & \\ & & & & & & 0 & 0 & 3 & 3 \\ & & & & & & & & 0 & 0 \end{bmatrix}. \tag{3.7}$$

Equation (3.6) and Equation (3.7) show the base matrices with $m_s = 1$ while Equation (3.8) shows a base matrix $\mathbf{B}$ with $m_s = 2$ where $\mathbf{B}_0 = [1\ \ 0]$, $\mathbf{B}_1 = [2\ \ 1]$ and $\mathbf{B}_2 = [0\ \ 2]$.

$$\begin{bmatrix} 1 & 0 & & & & & & & & \\ 2 & 1 & 1 & 0 & & & & & & \\ 0 & 2 & 2 & 1 & 1 & 0 & & & & \\ & & 0 & 2 & 2 & 1 & 1 & 0 & & \\ & & & & & & \ddots & & & \\ & & & & & & & \ddots & & \\ & & & & 0 & 2 & 2 & 1 & 1 & 0 \\ & & & & & & 0 & 2 & 2 & 1 \\ & & & & & & & & 0 & 2 \end{bmatrix} \tag{3.8}$$

For a base matrix in this form, the copy-and-permute operation is equivalent to replacing each entry $\mathbf{B}_{i,j}$, $i = 1, \ldots, M_P$, $j = 1, \ldots, N_P$, in the base matrix with a sum of $\mathbf{B}_{i,j}$ permutation matrices of size $M \times M$. As a result, in Equation (3.6), Equation (3.7), and Equation (3.8), each of the entries 1, 2, 3, and 0 is replaced by an $M \times M$ matrix such that each column and row contains only 1, 2, 3, and 0 ones, respectively. The resulting matrix after this procedure represents the parity-check matrix $\mathbf{H}$ of the code which has a size of

$$(L + m_{\mathrm{s}})M \times 2LM. \tag{3.9}$$

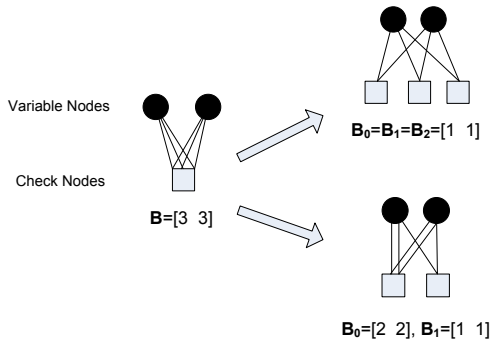

Figure 3.2. Decomposition of $\mathbf{B}$ into $\mathbf{B}_i$.

Note that, due to the special structure presented in Equation (3.6), Equation (3.7), and Equation (3.8), the first and last m_s rows of the base matrix $\mathbf{B}$, or equivalently, the first and last Mm_s check nodes of the constructed parity-check matrix $\mathbf{H}$ of the SC code have lower weights than the rest of the graph. Therefore, we are not able to call these codes (J, K)-regular anymore, even though a big portion of the parity-check matrix satisfies the regularity condition. Instead, SC codes with parity-check matrices that differ slightly from a regular parity-check matrix are referred to as *asymptotically* (J, K)-regular, meaning that if we neglect the first and last portions of irregularities in $\mathbf{H}$ (corresponding to the case with $L \to \infty$), the remaining matrix can be called (J, K)-regular. This slight irregularity of SC codes resulting from the termination of the underlying LDPC convolutional code has been shown to be the reason for SC codes having better BP decoding thresholds compared to corresponding block code ensembles [22].

3.1.1. Counting All Possible Asymptotically $(J, 2J)$-regular Ensembles

The base matrix of an asymptotically $(J, 2J)$-regular code can be written as

$$\mathbf{B} = [b_1 \;\; b_2] = \sum_{i=0}^{m_\mathrm{s}} \mathbf{B}_i, \tag{3.10}$$

where $\mathbf{B}_i = [b_{i1} \;\; b_{i2}]$, $i = 0, 1, \ldots, m_\mathrm{s}$, and $b_j = \sum_{i=0}^{m_\mathrm{s}} b_{ij}$, with $j = 1, 2$. An illustration of such a decomposition is given in Figure 3.2.

Table 3.1. Number of all possible asymptotically $(3, 6)$-regular ensembles for different code memories.

m_{s}	1	2	3	4	5	6	7
N	16	100	400	1225	3136	7056	14400

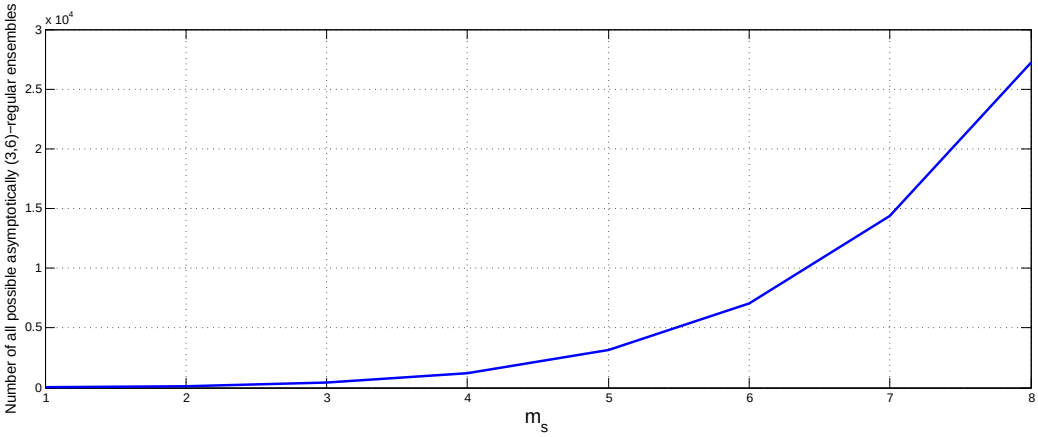

Figure 3.3. Number of all possible asymptotically $(3, 6)$-regular ensembles increases exponentially with m_s.

In order to count the number of all possible decompositions of $\mathbf{B}$ into $\mathbf{B}_i$, we refer to the decompositions of b_1 and b_2 as two independent events. The problem can be considered as counting the number of different ways to distribute b_1 balls in $m_{\mathrm{s}}+1$ bins and b_2 balls in $m_{\mathrm{s}}+1$ bins. Using combinational arguments we obtain this number as

$$N = \binom{b_1 + m_{\mathrm{s}}}{b_1}\binom{b_2 + m_{\mathrm{s}}}{b_2} = \binom{b_1 + m_{\mathrm{s}}}{m_{\mathrm{s}}}\binom{b_2 + m_{\mathrm{s}}}{m_{\mathrm{s}}}. \tag{3.11}$$

Table 3.1 shows the number of all possible asymptotically $(3, 6)$-regular ensembles for different values of the code memory m_{s} and $\mathbf{B} = [3\ \ 3]$.

As it can be seen in Figure 3.3 the number of possible code ensembles is increasing very fast by m_{s} but we are more interested in ensembles with code memory of 1 or 2, since larger values for the code memory m_{s} result in larger decoding complexity and delay.

3.2. Window Decoding

The special structure of the parity-check matrix $\mathbf{H}$ of an SC code allows the use of a window decoding (WD) scheme that breaks down the BP decoding into a series of decoding steps. A window decoder with a window size of W performs BP decoding over a portion of the codeword where the window consists of $W \times M$ rows and all connected columns. The symbols to be decoded within a window are referred to as the targeted symbols. After successful decoding of the targeted symbols is achieved or when the maximum number of iterations has been reached, the window slides over another portion to the right and again performs BP to decode the targeted symbols in the new position [24]. Figure 3.4 illustrates the concept of WD under *full* and *freeze* schemes for a window size of 4 at the first three decoding instants for an SC code with $m_{\mathrm{s}} = 1$ and $L = 9$. In this figure, each box indexed with 1 represents a permutation matrix of size $M \times M$ and every box indexed with 2 represents the sum of two permutation matrices of size $M \times M$ such that there are two ones in each row and column of the resulting matrix. As can be seen from the figure, these two decoding schemes are slightly differ from each other. For the full scheme, there is overlap (in terms of columns) between the current and the next window position. For instance at the first decoding instant, rows $1-4$ and all connected columns $(1-8)$ are processed. At the second decoding instant, rows $2-5$ and columns $1-10$ are processed. The first two non-zero elements of each row are processed twice in this scheme. This can be considered as the priori information injected to the decoder for the next window position. However in *freeze* scheme, initially rows $1-4$ and all connected columns $(1-8)$ are processed and for the second decoding instant, rows $2-5$ and columns $3-10$ are decoded. Window slides to the next position and decoding takes place with no overlap between the current and the next window position.

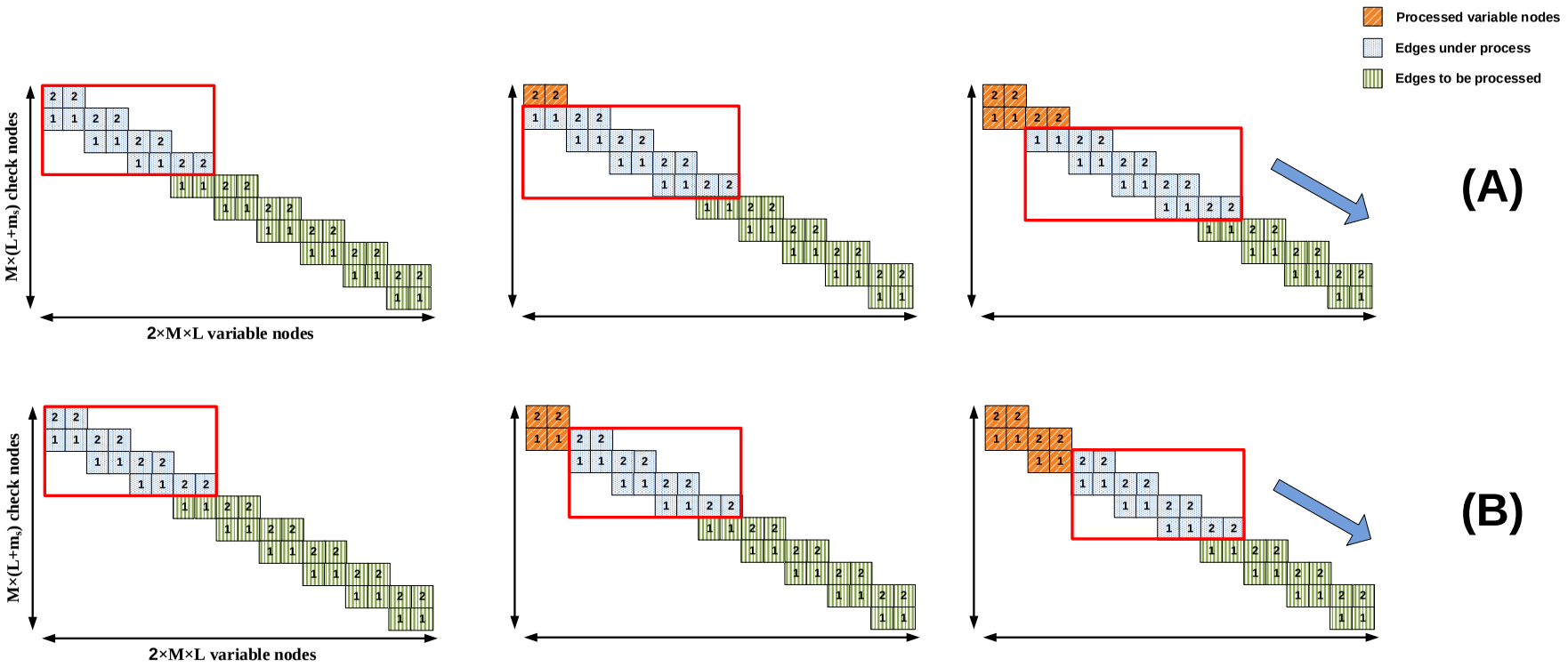

Figure 3.4. Illustration of WD (A) full scheme, (B) freeze scheme at the first three decoding instants with $W = 4$.

The most desirable advantage of WD of SC codes is reduced latency of this decoding scheme. In this method, decoding is done within a window which contains only a portion of the codeword, thus the delay will be lower since decoder does not have to wait for the whole codeword to reach the receiver. The window decoder starts decoding as soon as it receives the codeword portion corresponding to the first window position. However, the decoding performance of a code ensemble under WD is lower than decoding performance of the same ensemble under block decoding since WD engages a portion of the codeword while block decoding takes the whole codeword under process.

Since low delay is a very desirable feature for many wireless communication applications, these codes can be very suitable candidates in communication scenarios where low latency is the primary design criterion. SC codes also allow decoding with variable frame lengths and variable maximum number of iterations for specific frames, which is also a very desirable feature for wireless applications. We can exploit this flexibility to design window decoders adjusted to use on wireless channels. If the fading is strong, the number of iterations is set to maximum and if the fading effect is weaker, the number of iterations is decreased to speed up the decoding process and reduce the delay. Furthermore, it has been studied in [25] that the asymptotic performance of WD of protograph-based SC codes is superior to the performance of block decoding of regular LDPC codes over the BEC and AWGN channels; a result that could possibly be extended to the fading channels.

3.3. Density Evolution Analysis for SC Codes

To determine the asymptotic performance of SC codes, density evolution analysis can be performed on the protograph of SC codes [19]. For the BEC, DE analysis to calculate the threshold is done again according to Equation (2.10) and Equation (2.11). However, there is a small difference for WD threshold calculation. As it was mentioned, threshold value makes the $p^{(l)}$ (variable-to-check-node message at l^{th} iteration) converge to zero. In WD, DE is calculated in a smaller portion of the protograph (in a window of size W) rather than the whole graph. Thus, messages exchanged may not converge

to zero even for large number of iteration. In this case, the target erasure probability δ is defined such that the window threshold makes $p^{(l)}$ converges to δ.

In this section, we evaluate the performance of asymptotically regular-$(3, 6)$ SC ensembles with minimal coupling ($m_s = 1$ and $m_s = 2$) using DE analysis over the BEC. In order to do this, all possible members of these ensembles constructed from the base matrix $\mathbf{B} = [3 \;\; 3]$ are enumerated and WD threshold (ϵ^*_{wd}) is calculated for each one of them. For the $m_s = 1$ family, block decoding thresholds (ϵ^*_{blk}) are also provided. Results are given in Tables 3.2 and Table 3.3. In Table 3.3, only the ensembles with ϵ^*_{wd} greater than the threshold of a regular-$(3, 6)$ block LDPC code are listed, since using an SC code with worse performance than that of the underlying LDPC block code would not be preferable. In the tables, each entry $\mathbf{B} = [a_1 a_2 \; ... \; a_{i-1} a_i]$, $i = 1, 2, \ldots, 2(m_s+1)$, represents an ensemble with $\mathbf{B_0} = [a_1 \;\; a_2], \mathbf{B_1} = [a_3 \;\; a_4], \ldots, \mathbf{B_{m_s}} = [a_{i-1} \;\; a_i]$. As can be seen in Table 3.2, block thresholds are greater than window thresholds, which is intuitive, since block decoding engages the whole graph while WD uses only some portion of the graph corresponding to the window. The only exception is the family of ensembles

$$a_1, a_2, a_3, \ldots a_i \in \{0, 3\}\,. \tag{3.12}$$

These ensembles are indeed small $(3, 6)$-regular block codes that are put diagonally into the base matrix without any spatial coupling. Thus, they show the same behavior under both block decoding and WD regardless of the window size ($\epsilon^*_{blk} = \epsilon^*_{wd} = 0.4294$).
Another point that has been observed is the equal WD threshold values of the *mirror* ensembles. We define mirror ensembles as

$$mirror[a_1 a_2 \; ... \; a_{i-1} a_i] = [a_2 a_1 \;\; \ldots \;\; a_i a_{i-1}]. \tag{3.13}$$

As it was mentioned, the most vital parts of an SC ensemble under the WD scheme are the irregular rows at the beginning and end of the base matrix. For mirror ensembles, degree of those irregular check nodes are the same and therefore, they carry the same

Table 3.2. ϵ^*_{blk} and ϵ^*_{wd} for all ensembles with single memory over the BEC. Maximum number of iterations is 10^3, $\delta = 10^{-6}$, $L = 40$, and $W = 4$.

Ensemble	ϵ^*_{blk}	ϵ^*_{wd}
[33 00]	0.4294	0.4294
[32 01]	0.4770	0.4710
[31 02]	0.4770	0.4685
[30 03]	0.4294	0.4294
[23 10]	0.4770	0.4710
[22 11]	0.4830	0.4843
[21 12]	0.4837	0.4307
[20 13]	0.4770	0.4704
[13 20]	0.4770	0.4685
[12 21]	0.4837	0.4307
[11 22]	0.4830	0.3599
[10 23]	0.4770	0.4079
[03 30]	0.4294	0.4294
[02 31]	0.4770	0.4704
[01 32]	0.4770	0.4079
[00 33]	0.4294	0.4294

Table 3.3. ϵ^*_{wd} of some ensembles with code memory of two which have greater thresholds than threshold of regular-$(3, 6)$ block LDCP code over the BEC for $W = 4$ and $W = 6$. Maximum number of iterations is 10^3, $\delta = 10^{-6}$, and $L = 40$.

Ensemble	$\epsilon^*_{wd=4}$	Ensemble	$\epsilon^*_{wd=6}$	Ensemble	$\epsilon^*_{wd=6}$
[12 21 00]	0.4307	[12 10 11]	0.4315	[00 01 32]	0.4717
[02 00 31]	0.4447	[11 12 10]	0.4377	[00 12 21]	0.4725
[22 00 11]	0.4495	[01 12 20]	0.4419	[02 11 20]	0.4725
[02 20 11]	0.4585	[02 10 21]	0.4468	[01 22 10]	0.4738
[00 02 31]	0.4668	[10 13 10]	0.4571	[13 10 10]	0.4751
[02 01 30]	0.4668	[12 11 10]	0.4592	[11 22 00]	0.4751
[02 30 01]	0.4668	[02 00 31]	0.4668	[22 00 11]	0.4801
[23 00 10]	0.4668	[02 30 01]	0.4704	[02 20 11]	0.4835
[13 20 00]	0.4685	[23 00 10]	0.4704	[12 21 00]	0.4843
[00 23 10]	0.4704	[00 02 31]	0.4710	[00 13 20]	0.4849
[02 31 00]	0.4704	[02 01 30]	0.4710	[03 10 20]	0.4849
[03 20 10]	0.4704	[12 20 01]	0.4711	[00 22 11]	0.4853
[23 10 00]	0.4710	[00 23 10]	0.4712	[02 21 10]	0.4853
[22 01 10]	0.4766	[02 31 00]	0.4712	[22 11 00]	0.4855
[02 21 10]	0.4801	[03 20 10]	0.4712	[01 32 00]	0.4855
[00 22 11]	0.4801	[23 10 00]	0.4712	[22 01 10]	0.4856
[22 11 00]	0.4843	[10 20 03]	0.4717	[13 20 00]	0.4874

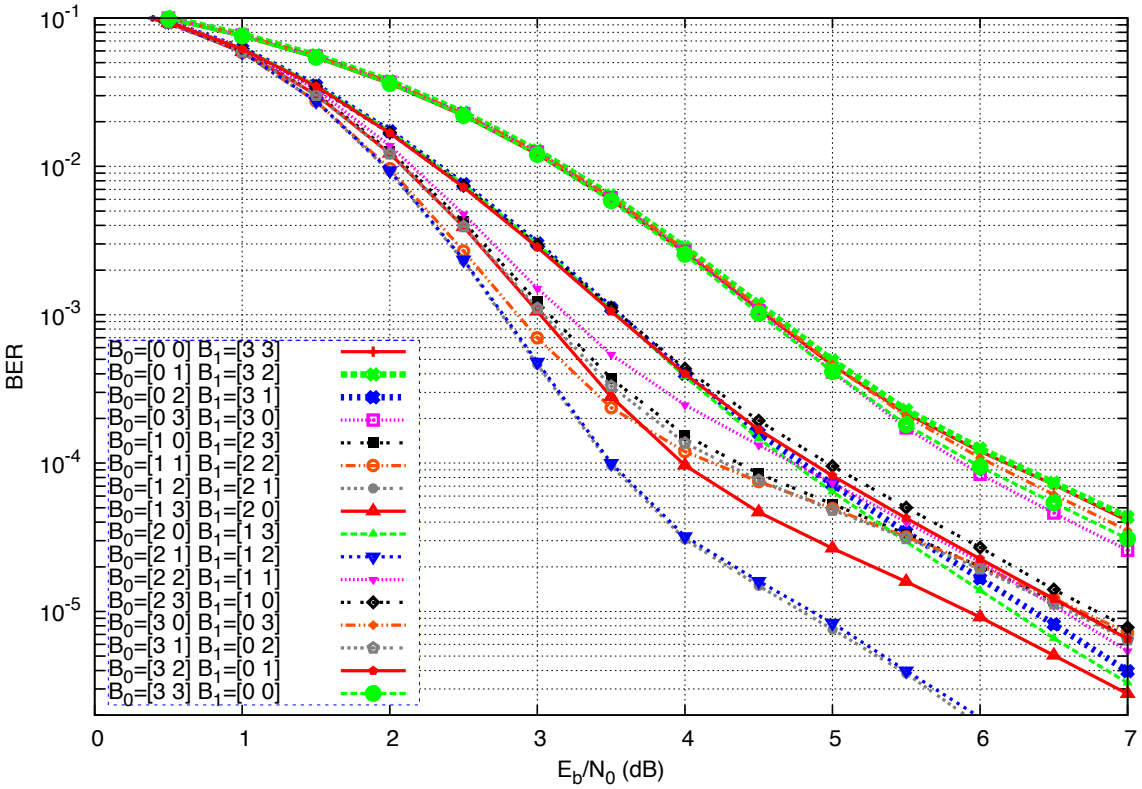

Figure 3.5. BER performance all possible SC code ensembles with $m_s = 1$, $M = 20$, $L = 100$, and $W = 6$ over the AWGN channel.

threshold values. Note that in Table 3.3, only one of the mirror ensembles is listed.

Although there are 100 asymptotically $(3, 6)$-regular SC ensembles with $m_s = 2$ (Table 3.1), only 32 of them (including mirror ensembles) have greater ϵ^*_{wd} than threshold of a regular-$(3, 6)$ block LDPC code for $W = 4$. For $W = 6$ case, there are 64 ensembles with higher ϵ^*_{wd} values. This verifies the effect of window size on the iterative decoding performance, nevertheless increasing W will result in more delay.

Bit error rate (BER) performance of codes constructed from all possible members of the SC ensembles with $m_s = 1$ (code length of 4000) over AWGN channel is presented in Figure 3.5 in which similar performance of the mirror code ensembles and also ensembles defined in Equation (3.12) can be observes. Note that, there may be some inconsistency with the corresponding DE results. However, we emphasize again that

the DE results indicate the asymptotic performance of code ensemble which may differ for the finite-length codes.

Some rules for designing SC ensembles were presented in the literature before [24]. However, these rules just emphasize the need for having large weights on the left side of a window to make sure that reliable messages are obtained early in the decoding process and passed on to the following window positions. In this work, we attempt to introduce additional practical rules based on actual DE analysis results to obtain low-delay and low-complexity SC communication systems. Satisfying all of these rules may not be feasible but they can be used as rules of thumb by a code designer.

- Ensembles with $m_s = 1$, while having less complexity and decoding delay, show better performance in general. However using ensembles with $m_s = 2$ increases the degree-of-freedom of the code construction process, allowing for additional code construction rules, such as those for modifying the cycles, trapping set, and absorbing set spectra, to be satisfied more freely.
- If an ensemble description has one of summations $(a_1 + a_2)$ or $(a_{i-1} + a_i)$ equal to zero, there is an unnecessary rate loss. In this case, it is better to use an ensemble with smaller code memory.
- It is helpful to choose a_1 or $a_2 \geq 2$. The best case is $a_1 = a_2 = 2$, if possible. This follow from the rules given in [24] and is verified here.
- For $m_s = 2$ ensembles, if $\max(a_3, a_4) \geq 2$, choosing $\max(a_1, a_2) > \max(a_5, a_6)$ improves the performance.
- For $m_s = 2$ ensembles, if $\max(a_3, a_4) < 2$, choosing a_i such that $(a_1+a_2)-(a_5+a_6)$ is minimized improves the performance.

Note that there may be some exceptions in the tables, but satisfying these rules in general, yields an SC ensemble with good characteristics. According to these criteria, ensemble with $\mathbf{B_0} = [2 \quad 2]$ and $\mathbf{B_1} = [1 \quad 1]$ has the best performance among all ensembles with $m_s = 1$. For $m_s = 2$, ensemble with $\mathbf{B_0} = [2 \quad 2]$, $\mathbf{B_0} = [0 \quad 1]$ and $\mathbf{B_2} = [1 \quad 0]$ (and its mirror) outperforms the other ensembles.

4. A LOW-DELAY CONVOLUTIONAL-INTERLEAVER-BASED COMMUNICATION SYSTEM

As discussed in the previous chapter, the ribbon-shaped structure of SC codes brings along several advantages in terms of code construction, encoding and decoding complexity, and decoding delay. However, it is the same structure that hurts the performance of SC codes over channels with memory, e.g., correlated erasure channels. In many practical applications, randomly constructed LDPC block codes allow the system designer to avoid using block interleavers, since the random connections of an LDPC graph guarantees that even consecutive erasures are spread over the entire codeword length. However, since the ribbon-shaped structure has a limited constraint length in the SC code parity-check matrix, bursts of erasures severely degrade the error performance.

In the literature, this problem has been tackled in the way of designing a special code structure and choosing the code parameters in a way to disperse the effects of the erasure bursts [26]. However, this solution imposes several constraints in the code construction step that could otherwise be relaxed to optimize iterative decoding thresholds, code properties, etc.

In this chapter, we introduce an SC-based communication system with convolutional interleaver which mitigates fatal effects of correlated erasure channel while still maintaining low delay. The performance of this this scheme is then evaluated by density evolution analysis and also several computer simulations.

4.1. Convolutional Interleaver

In this section, we propose to separate the code design and erasure burst dispersion tasks and design a spatially-coupled communication system that has high error

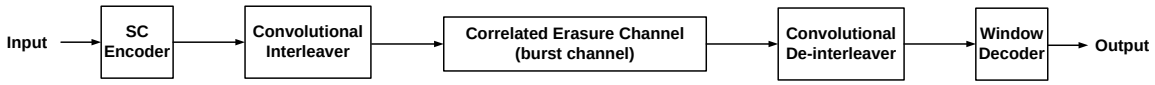

Figure 4.1. Block diagram of a low-delay convolutional interleaver based communication system for correlated erasure channels.

performance and design flexibility. The proposed system employs a convolutional interleaver, which is used to disperse the erasure burst over a window that is wider than W so that the window decoder can observe the consecutive erasures of the channel as random. The corresponding block diagram is presented in Figure 4.1.

The information sequence is first input to an SC encoder and the corresponding codeword is interleaved using a convolutional interleaver, whose structure will be described shortly. The output of the interleaver is then fed to the channel and is degraded by a burst of erasures starting at a random position. At the receiver side, the received sequence is deinterleaved and decoded using the window decoder. It is clear that the interleaving and deinterleaving steps cannot be achieved without causing some delay, but we will show next that this delay can be kept to a minimum while achieving a great improvement in error performance.

A convolutional interleaver is composed of Λ delay lines where the λ^{th} delay line consists of $1 + (\lambda - 1) * \Psi$ delay elements, $\lambda = 1, 2, \ldots, \Lambda$ [27]. The block diagram of the convolutional interleaver with $\Lambda = 3$ and $\Psi = 2$ is given in Figure 4.2. The first delay element in each delay line is initialized with data, whereas the remaining ones are assumed to be initialized with zeros.

There are two practical issues faced when a convolutional interleaver is used to inerleave a finite-length sequence. The first one is the zeros present within the delay elements when the interleaver is initialized. The second issue relates to the end of the data sequence. Since there are unequal number of delay elements on each delay line, some portion of data still remain in the delay elements, even after the last code symbol has been input to the interleaver. In order to flush this portion out of the

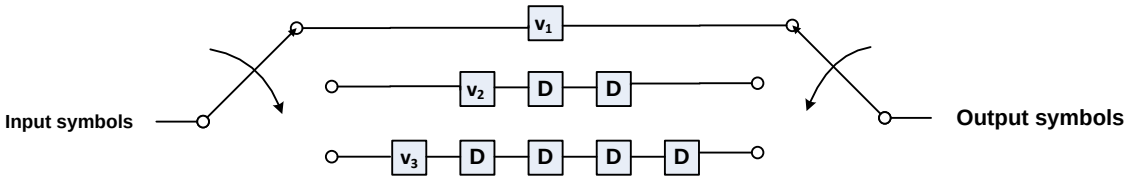

Figure 4.2. Schematic of the convolutional interleaver π_1 with $\Lambda = 3$ and $\Psi = 2$ at the first time instant. Initial values are set to zero. Output of this time instant will be $v_1, 0, 0$.

interleaver, several dummy symbols (zeros) must be padded at the end of each input symbol stream. As a result, there will be several deterministic zero symbols among the output symbols that need to be filtered out.

As an example, assume that a codeword $\mathbf{v} = (v_1, v_2, \ldots, v_{10})$ of length of 10 is the input to the convolutional interleaver π_1. The output of the interleaver is then determined as

$$\pi_1(\mathbf{v}) = (v_1, 0, 0, v_4, 0, 0, v_7, v_2, 0, v_{10}, v_5, 0, 0, v_8, v_3, 0, 0, v_6, 0, 0, v_9, 0, 0, \ldots). \tag{4.1}$$

In order not to waste transmission resources by transmitting non-information-carrying symbols, zeros are punctured before transmitting the data over the channel. Therefore, the actual data to be transmitted becomes

$$\mathbf{v}' = \pi_1(\mathbf{v}) = (v'_1, v'_2, v'_3, \ldots, v'_{10}) = (v_1, v_4, v_7, v_2, v_{10}, v_5, v_8, v_3, v_6, v_9). \tag{4.2}$$

We then observe that the deinterleaver to be used to recover $\mathbf{v}$ from $\mathbf{v}'$ is

$$\mathbf{v} = \pi_1^{-1}(\mathbf{v}') = (v'_1, v'_4, v'_7, v'_2, v'_{10}, v'_5, v'_8, v'_3, v'_6, v'_9)$$

i.e. $\pi_1 = \pi_1^{-1}$ and the same interleaver can be used as a deinterleaver to recover the original codeword. This gives the system designer an advantage in the form of reduced memory requirements to store the interleaver and deinterleaver and also allows

Table 4.1. Delay profile of the convolutional interleaver π_1.

clk	$\mathbf{v}$	$\mathbf{v}'$	T
1	v_1	v_1	0
2	v_2	v_4	2
5	v_3	v_7	2
8	v_4	v_2	0
9	v_5	v_{10}	1
10	v_6	v_5	0
10	v_7	v_8	0
10	v_8	v_3	0
10	v_9	v_6	0
10	v_{10}	v_9	0

the reuse of the same hardware block to interleave and deinterlave data for two-way communications.

Table 4.1 gives the delay analysis of this interleaver for the first 10 symbols of the codeword. $\mathbf{v}$ and $\mathbf{v}'$ denote original and interleaved symbol sequences, respectively. clk shows the current time instant and T is the delay introduced in current time instant.

At the first time instant ($clk = 1$), the interleaver assigns the first input symbol to the first output symbol without any delay. At the second time instant ($clk = 2$), the fourth input symbol is assigned to the second output symbol, but since the second and third symbols of the input have not arrived yet, the interleaver waits for two more instants in order to access the fourth input symbol, which causes $T = 2$ units of delay and make the clock $clk = 4$. At the next time instant ($clk = 5$), the interleaver waits for two more instants to output the seventh input symbol. Once $clk = 10$ is reached, there is no additional delay.

In this simple example, the total delay (T_{total}) equals to 5 time instants and is actually upper bounded by the number of zero elements in the delay elements at the

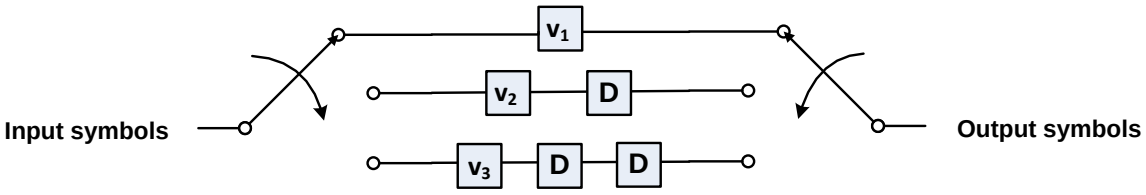

Figure 4.3. Schematic of the convolutional interleaver π_2 with $\Lambda = 3$ and $\Psi = 1$ at the first time instant.

Table 4.2. Delay profile of the convolutional interleaver π_2.

clk	$\mathbf{v}$	$\mathbf{v}'$	T
1	v_1	v_1	0
2	v_2	v_4	2
5	v_3	v_2	0
6	v_4	v_7	1
8	v_5	v_5	0
9	v_6	v_3	0
10	v_7	v_{10}	0
10	v_8	v_8	0
10	v_9	v_6	0
10	v_{10}	v_9	0

time of initialization, which is 6. For a larger code of length 80 ($L = 40$, $M = 1$, and $K' = 2$) the total delay is indeed verified to be equal to 6. This helps us in demonstrating the delay introduced by the interleaver (and deinterlaver) is almost negligible when compared to the block length of the code.

For a more comprehensive comparison, we also study the properties of two more convolutional interleavers with less delay elements in their structure. Interleaving patterns of convolutional interleaver π_2 given in Figure 4.3 with input sequence $\mathbf{v} =$

$(v_1, v_2, \ldots, v_{10})$ is

$$\mathbf{v}' = \pi_2(\mathbf{v}) = (v'_1, v'_2, v'_3, \ldots, v'_{10}) = (v_1, v_4, v_2, v_7, v_5, v_3, v_{10}, v_8, v_6, v_9). \tag{4.3}$$

Delay profile of this interleaver is given in Table 4.2. As can be seen, total delay of interleaving process $T_{total} = 3$ has been reduced since there are fewer delay elements in the structure of this convolutional interleaver. The same analysis is repeated for the convolutional interleaver π_3 illustrated in Figure 4.4. The output of this interleaver for input sequence $\mathbf{v} = (v_1, v_2, \ldots, v_{10})$ is

$$\mathbf{v}' = \pi_3(\mathbf{v}) = (v'_1, v'_2, v'_3, \ldots, v'_{10}) = (v_1, v_3, v_5, v_2, v_7, v_4, v_9, v_6, v_8, v_{10}). \tag{4.4}$$

Correspondent delay profile is given in Table 4.3 and total delay of this interleaver equals to $T_{total} = 2$.

Note that in all three convolutional interleavers that discussed above, de-interleaving pattern can be simply determined as $\pi_i = \pi_i^{-1},\ i = 1, 2, 3$. These interleavers have smaller delay values. However, intuitively one can guess that their performance will not as good as the first interleaver since they cannot spread the elements of the input sequence as widely as the first one and therefore, the consecutive elements hit by a burst may still be close to each other even after interleaving which can result in degradation of the performance.

In the next section, we try to demonstrate the trade-off between interleaving delay and performance of the system by means of computer simulations.

4.2. Simulation Results

The SC code ensemble with $\mathbf{B}_0 = [2\ \ 2]$, $\mathbf{B}_1 = [0\ \ 1]$ and $\mathbf{B}_2 = [1\ \ 0]$ (ensemble D) is proposed in [24] as a robust ensemble for the correlated erasure channel. This result is derived from some code selection criteria and without using any interleaver scheme.

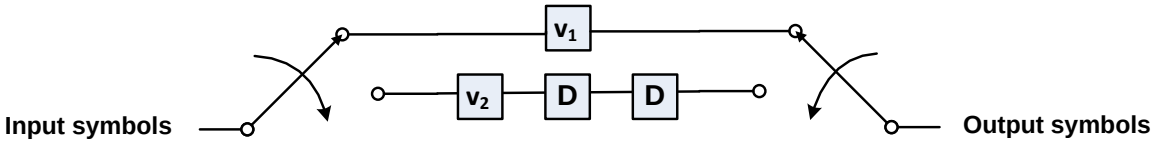

Figure 4.4. Schematic of the convolutional interleaver π_3 with $\Lambda = 2$ and $\Psi = 2$ at the first time instant.

Table 4.3. Delay profile of the convolutional interleaver π_3.

clk	$\mathbf{v}$	$\mathbf{v}'$	T
1	v_1	v_1	0
2	v_2	v_3	1
4	v_3	v_5	1
6	v_4	v_2	0
7	v_5	v_7	0
8	v_6	v_4	0
9	v_7	v_9	0
10	v_8	v_6	0
10	v_9	v_8	0
10	v_{10}	v_{10}	0

We first evaluate the asymptotic performance of the ensemble D and also the SC code ensemble with $\mathbf{B}_0 = [2\ \ 2]$ and $\mathbf{B}_1 = [1\ \ 1]$ (ensemble B) under window decoding (full schedule) by the density evolution technique over the BEC. In our analysis, L is set to 40 and therefore, the base matrix is of size 41×80 for ensemble B and 42×80 for ensemble D as $m_s = 2$ for ensemble D. The maximum number of iterations is 1000, the window size is chosen to be $W = 4$ and the target erasure probability is $\delta = 10^{-6}$. We show that our convolutional interleave based system with ensemble B outperforms the proposed ensemble D in [24] over correlated erasure channel.

The window decoding threshold over the standard BEC are calculated as $\epsilon^*_{wd} = 0.4843$ and $\epsilon^*_{wd} = 0.4766$ for ensembles B and D, respectively. However, we are interested in calculating the window decoding threshold of these code ensembles over the correlated erasure channel. We assume the SBC where few high rate erasures occurs at consecutive intervals and the rest of the codeword is degraded by the channel's standard erasure rate. We consider erasure bursts of length four with a burst erasure rate of $\epsilon_B = 0.6$. Note that the burst length of four is defined on the protograph. When the expansion factor is chosen to be nontrivial ($M > 1$) for a practical code, this analysis corresponds to having a single burst of $4M$ high probability erasures.

We sweep this single burst along the codewords of length $N_p = 80$, i.e., 4-tuples $(v'_i, v'_{i+1}, v'_{i+2}, v'_{i+3})$, $i = 1, \ldots, 77$ are erased with probability $\epsilon_B = 0.6$ and the remaining symbols are erased with probability ϵ. Using the density evolution technique, we look for the largest erasure rate ϵ that allows for successful decoding. The results are presented in Figure 4.5, where the iterative decoding threshold of the corresponding code ensembles are demonstrated to be severely reduced due to the presence of consecutive high probability erasures. The top plots in this figure show the iterative decoding thresholds of the code ensembles when no bursts are present. It can be seen that whereas ensemble B can overcome channel erasures with probability $\epsilon^*_{wd} = 0.4843$ without any erasure bursts, the threshold values fall in the region $0.2182 \leq \epsilon \leq 0.2790$. This is a large loss of performance due to the correlated nature of erasures. For ensemble D, threshold values fall in the region $0.2972 \leq \epsilon \leq 0.3107$ which is better than the ensemble B. It verifies the effect of code design criteria on mitigating degradation

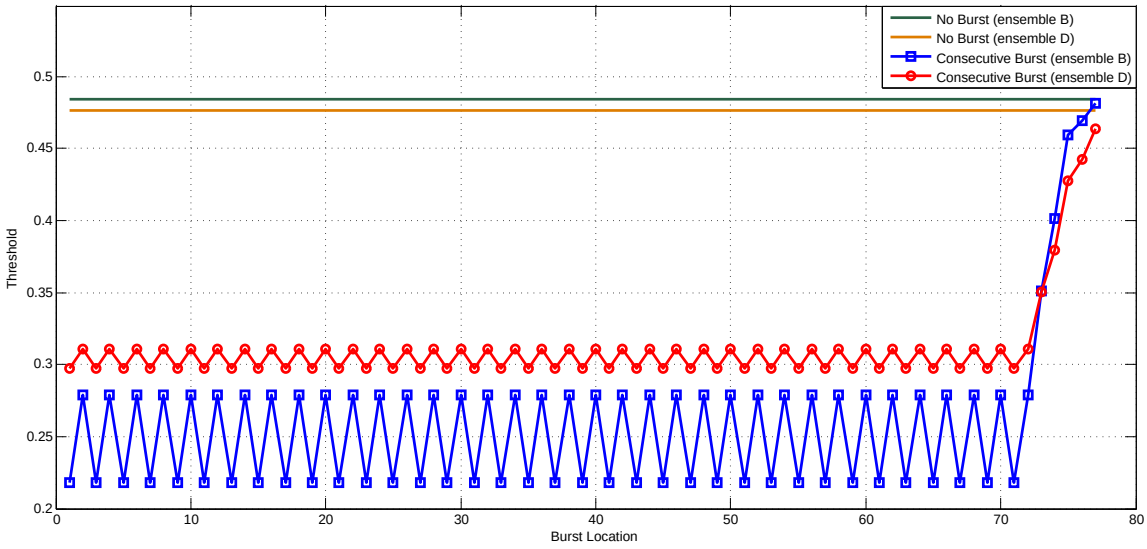

Figure 4.5. Density evolution results for non-interleaved data transmission over the correlated erasure channel for code ensembles B and D. Also provided are the window decoding thresholds of the code ensembles when erasures occur randomly.

caused by burst of erasures.

As can be seen from the figure, iterative decoding thresholds are dependent on whether a burst starts at an odd or even position of the codeword. A burst that starts at an odd position (e.g. first burst that happens on symbols $1-4$), affects three check nodes connected to these variable nodes. Two of these check nodes are partially affected and the third one is completely erased (assuming that the erasures within a burst are highly probable). This will decrease the threshold since there is a loss of connectivity in the graph. This makes threshold to be is at its minimum. For the even start positions (e.g. second burst that happens on symbols $2-5$), four check nodes are affected, but none of the check nodes are completely erased and this causes a slightly higher iterative decoding threshold compared to that of an odd start position. This behavior describes the alternating nature of the plot. Note that the burst covering the last symbols of a codeword does not severely affect the targeted symbols and that is why pseudo-high thresholds are observed on those burst locations. Furthermore, since it is more likely that a burst of erasures hits the middle regions of a codeword (for large L values), their threshold improvement is more concerning.

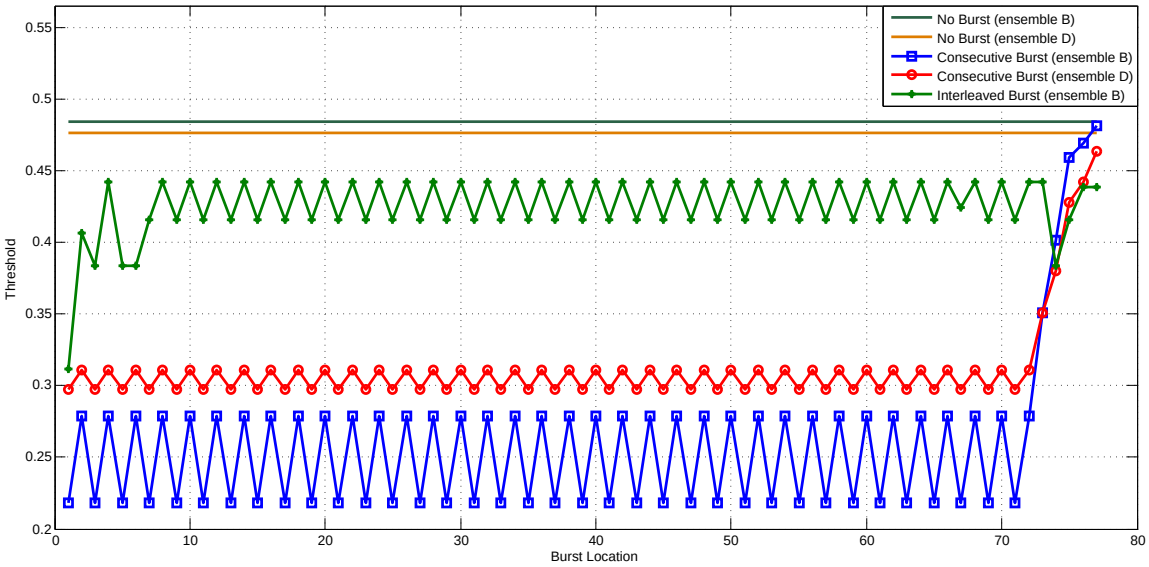

Figure 4.6. Density evolution results for non-interleaved and interleaved data transmission over the correlated erasure channel for code ensembles B and D. Window decoding thresholds of these code ensembles over the BEC are also provided.

Figure 4.6 shows the iterative decoding thresholds of the ensemble B when the convolutional interlaver π_1 is employed. Note that for interleaved burst, each burst location indicates the interleaved values of the consecutive burst location. For instance, burst location 1 (which indicates a burst at symbols 1, 2, 3, and 4) means that high probable erasures ϵ_B happens at locations 1, 4, 7, and 2.

It is clear from the obtained iterative decoding thresholds that by employing a convolutional interleaver, the error performance of the communication system increases considerably. Although this gain comes at cost of introducing an interleaving delay, this delay is almost negligible when the codeword is long enough (corresponding to large L values). We also present the iterative decoding thresholds of the ensemble B under convolutional interleavers π_2 and π_3 in Figure 4.7 to demonstrate the effect of decreased delay on the performance. For an exhaustive comparison of the performance losses by using convolutional interleavers, threshold of the ensemble B when block interleaver is used, is also provided. It can be seen that the performance of interleaver π_1 is better than the other two since it consists of more delay elements. Therefore, it

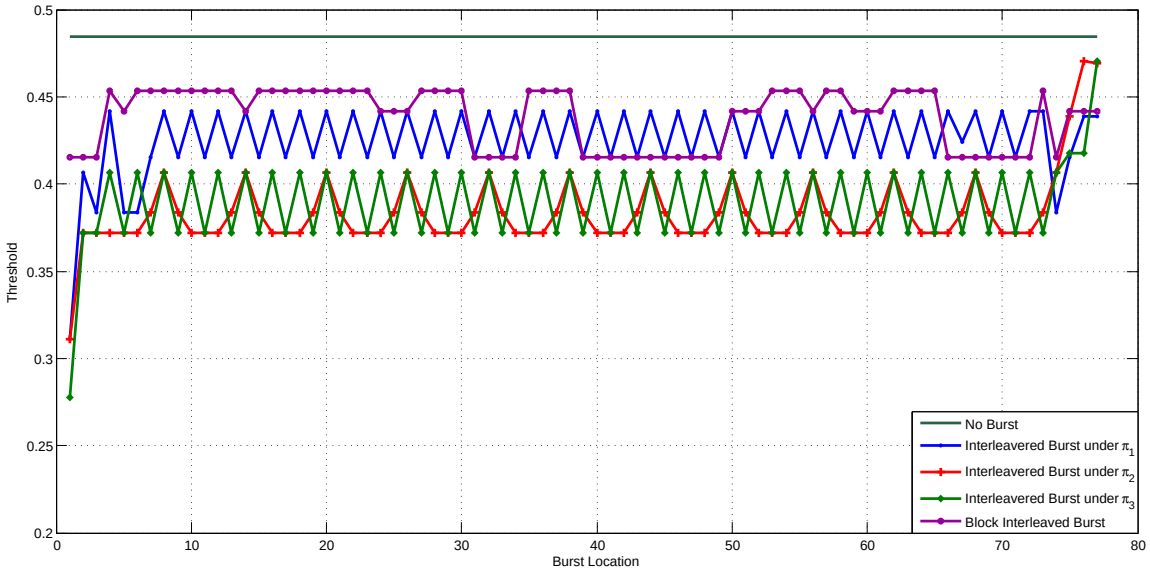

Figure 4.7. Density evolution results for interleaved data transmission over the correlated erasure channel for code ensemble B under different convolutional interleavers π_1, π_2, and π_3 and also a block interleaver. Window decoding threshold of the code ensemble B over the BEC is also provided.

interleaves the elements of input sequence more separately. This causes a consecutive burst to be replaced by several separate single erasures located farther from each other and this enables window decoder to decode these affected symbols more efficiently.

Another important fact that should be considered is the performance of the block interleaver. Clearly, employing a block interleaver results in higher threshold values since it is expected that a block interleaver assigns adjacent input symbols to distant locations in the output sequence compared to the convolutional interleavers. However, Figure 4.7 shows that there is not a serious loss between threshold of code ensemble B with block interleaver and its threshold under convolutional interleaver π_1.

It is crucial if we notice that, while the total delay of the block interleaver is $T_{total} \leq 80$ units, convolutional interleaver π_1 introduces the total delay of $T_{total} = 6$ units only. This means that by sacrificing small portion of the performance, we can gain almost 13 times of improvement in delay reduction of the interleaving system by using

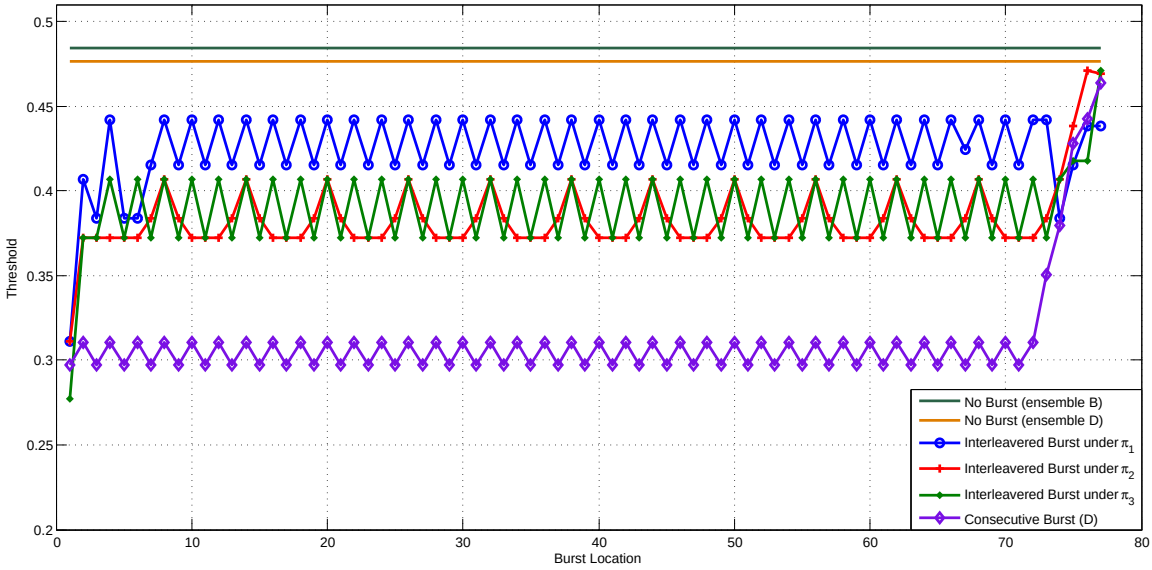

Figure 4.8. Density evolution results for interleaved data, derived from code ensemble B, under three different convolutional interleavers π_1, π_2, and π_3 and also non-interleaved data transmission over the correlated erasure channel for code ensembles D. Window decoding thresholds of these code ensembles over the BEC are also provided.

convolutional interleaver π_1. Even for some certain applications in which less delay is of the first priority, the use of convolutional interleavers π_2 and π_3 is also reasonably justified since they have almost 26 and 40 times less interleaving delays compared to the block interleaver, respectively. On the other hand, convolutional interleavers π_2 and π_3 assign input symbols to output sequence with less distance which causes decreasing in the decoding performance of WD. However as it is illustrated in Figure 4.8, coding performance of the system even under convolutional interleavers π_2 and π_3 is higher than the non-interleaving approach.

The results from the DE analysis show the asymptotic behavior of the system and can be used as a benchmark for analysis of practical applications which we are more interested in. To do this, we also evaluate the BER performance of the SC codes derived from ensembles B and D over different erasure rate (ϵ) of the channel in which $L = 40$, and $M = 100$. Thus according to Equation (3.9), the length of the codeword

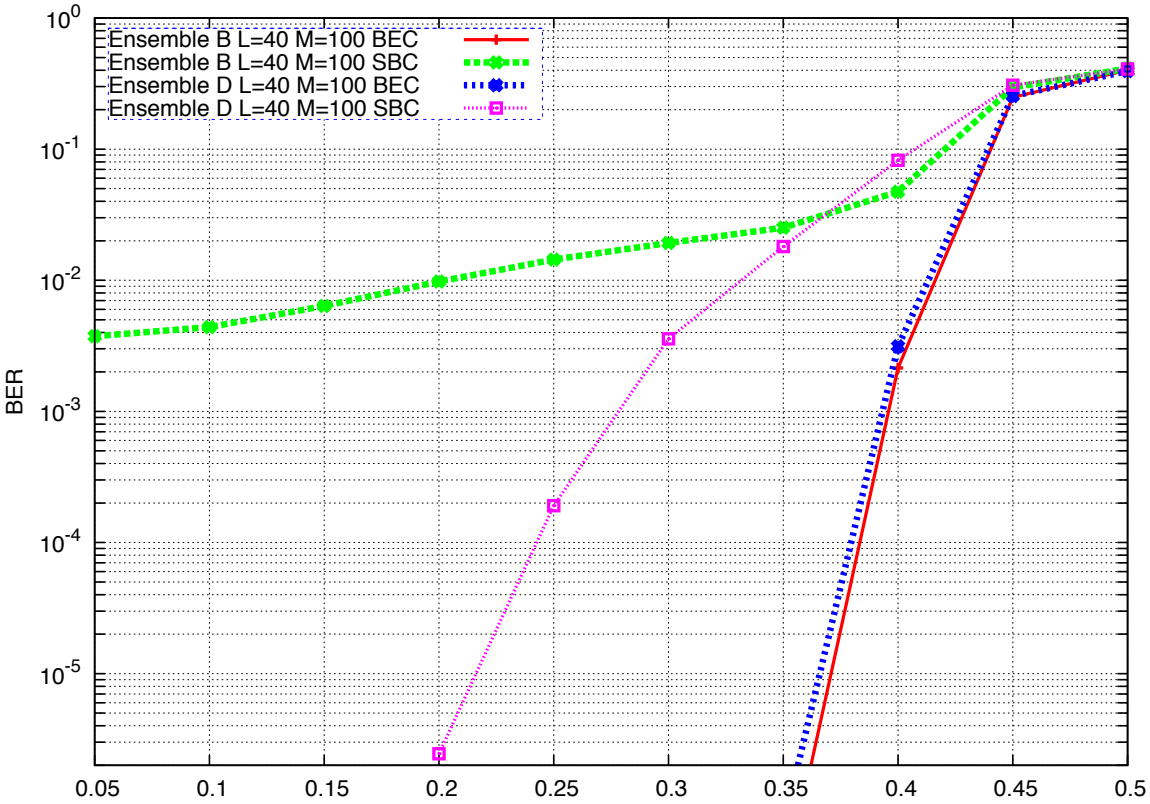

Figure 4.9. BER performance of code ensembles B and D over BEC and SBC.

is 8000. In these simulations, burst of high-probable erasures ($\epsilon_B = 0.6$) with length of $4M = 400$ bits hits the constant position in the middle of the codeword. The results of simulations are presented in Figures 4.9-12.

Figure 4.9 shows the BER performance of the codes B and D over the BEC and the SBC without any interleaving scheme. As can be seen, BER performance of code B is slightly higher than code D over the BEC which is consistent with the DE results for ensembles B and D. However as estimated by the DE analysis, BER performance of code D outperforms code B over the SBC.

Figure 4.10 shows the BER performance of the code B over the SBC, when three different convolutional interleavers π_1, π_2, and π_3 are employed. It is clear that BER performance under convolutional interleavers π_1 is better which confirms the trade-off between the performance and the delay of the system. BER performance of the code

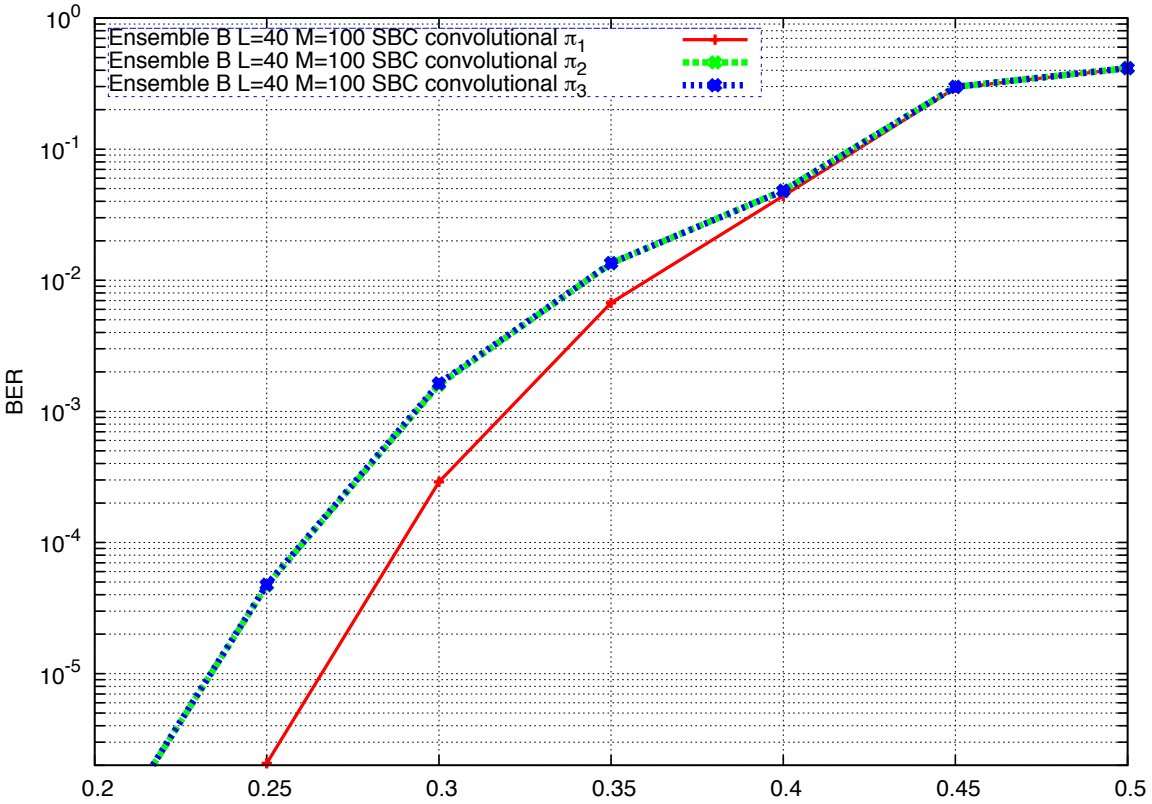

Figure 4.10. BER performance of code ensemble B under convolutional interleavers π_1, π_2, and π_3 over the SBC.

under convolutional interleavers π_2 and π_3 is almost the same.

In Figure 4.11, the performance gain of the convolutional interleaver scheme compared to non-interleaving scheme which recommends using code ensemble D based on code design criteria is presented. Note that even convolutional interleavers π_2 and π_3 which have fewer delay elements (consequently, relatively weaker performance), performs better than the code that is specifically designed to be implemented over erasure channels with memory. This observation is important since we can use lower-memory codes with no code design constraints to achieve a better performance. This means that relying only on design criteria which produce some good codes mitigating the effects of burst of erasures in the correlated channels is not enough and use of the interleaver schemes to improve the performance is inevitable.

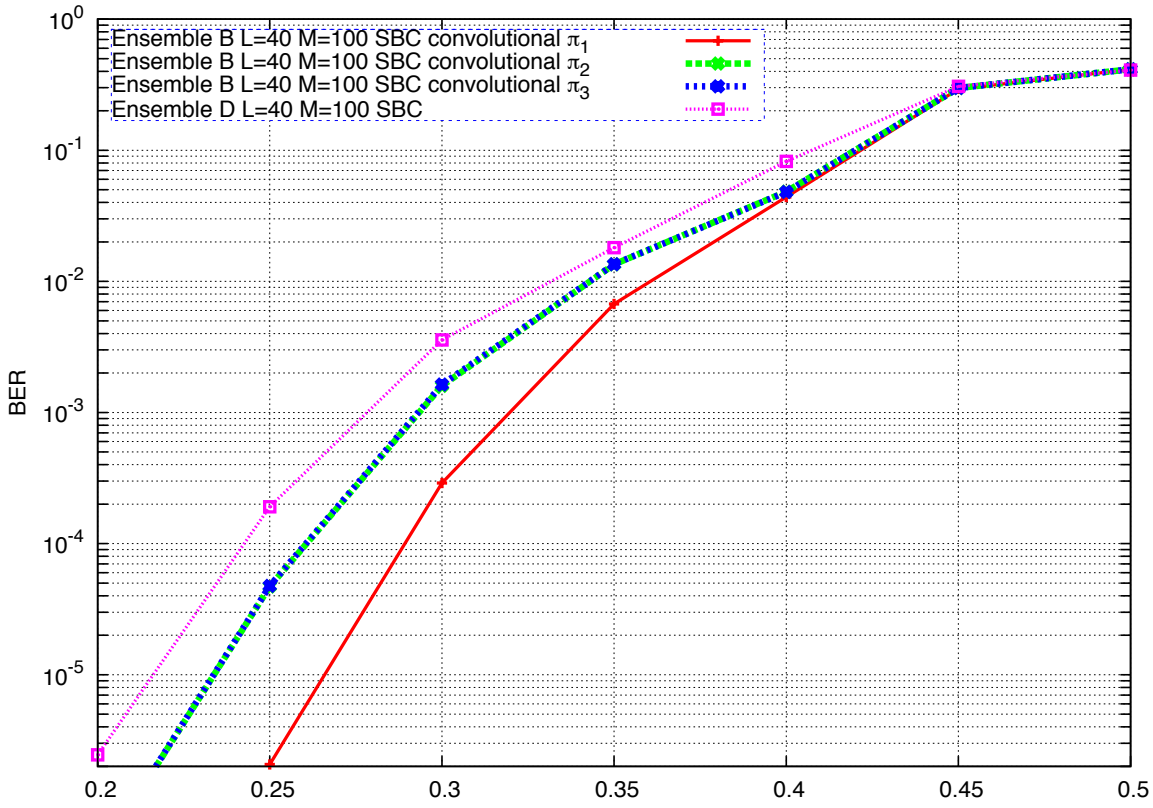

Figure 4.11. BER performance of code ensemble D with no interleaving scheme and code ensemble B under convolutional interleavers π_1, π_2, and π_3 over the SBC.

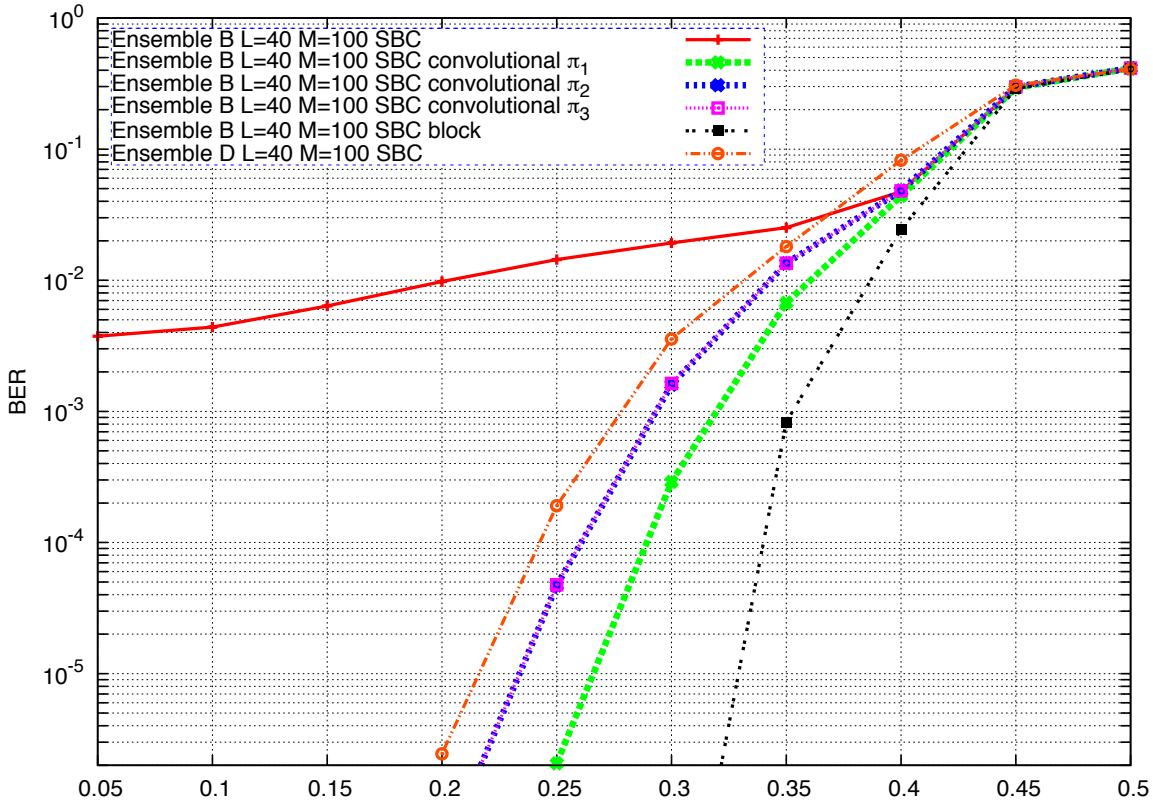

Figure 4.12. BER performance of codes ensembles B and D with no interleaving schemes and also code ensemble B under both convolutional and the block interleaver schemes over the SBC.

Finally, a comprehensive comparison between performance of codes B and D is illustrated in Figure 4.12. It shows the performance of code B under three different convolutional interleavers and a block interleaver and also performance of code D over the SBC. This figure actually indicates the performance loss under convolutional interleavers compared to block interleaver. As can be seen, BER performance under block interleaver is the highest. Nevertheless, this superiority comes at the cost of very large delay. Depending on specific application, trade-off between performance loss and reduced delay should be taken into account. However based on the results, it seems that employing a low-delay convolutional interleaver gives a efficient trade-off between performance and delay.

For the case when the burst hits a random position of the codeword, different

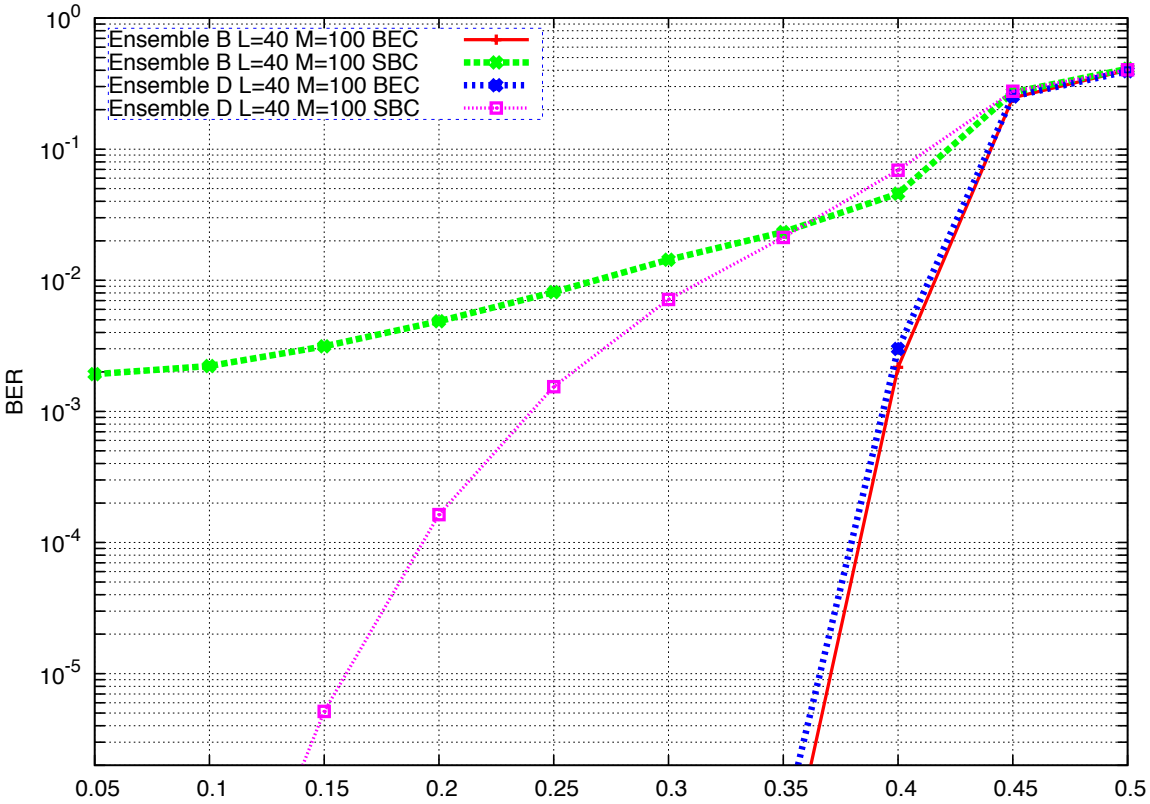

Figure 4.13. BER performance of code ensembles B and D over BEC and SBC (random burst position).

simulations have been performed in the same order. The results are presented in the Figures 4.13-16. It can be seen that these results are consistent with the previous simulations in which burst start at a constant position in the middle of the codeword. However, as can be observed, BER performance of the codes is generally weaker in the latter simulations. It follows from the fact that start position of the burst sequence is random for each block and it may hit the beginning of the codeword (corresponds to first few window positions) which severely damages the window decoding performance even after interleaving process.

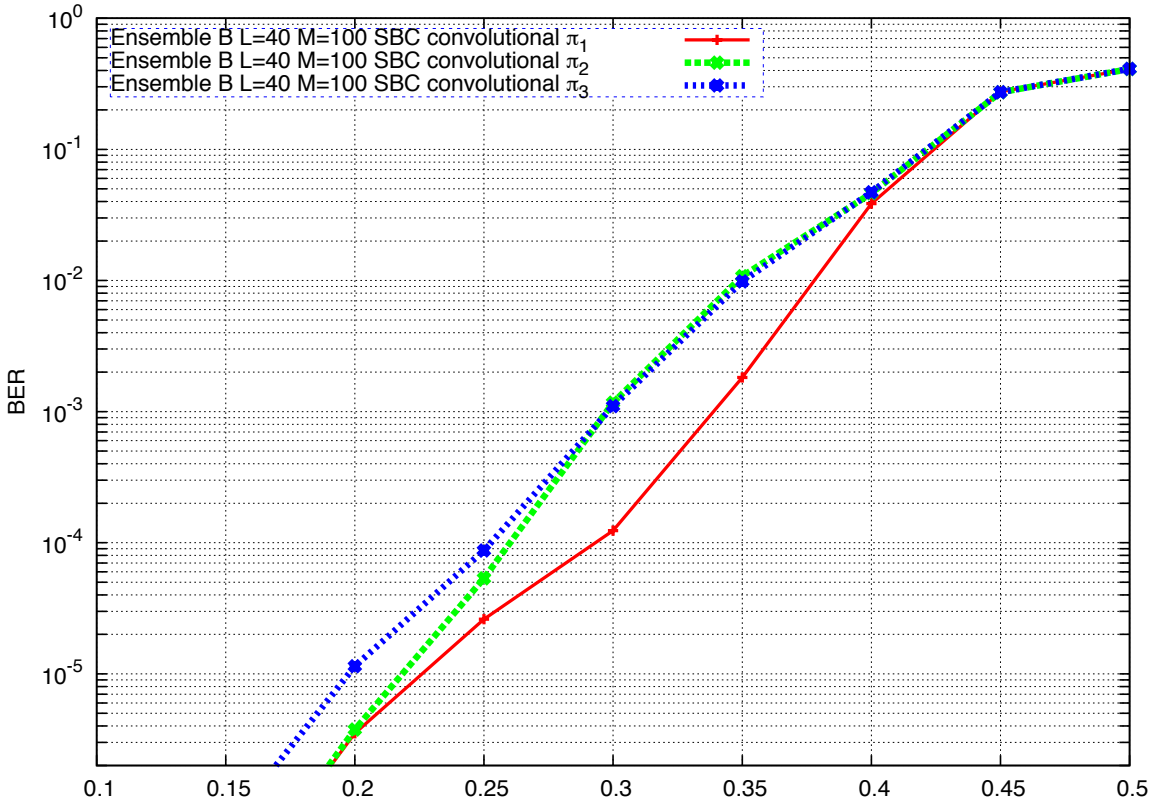

Figure 4.14. BER performance of code ensemble B under convolutional interleavers π_1, π_2, and π_3 over the SBC (random burst position).

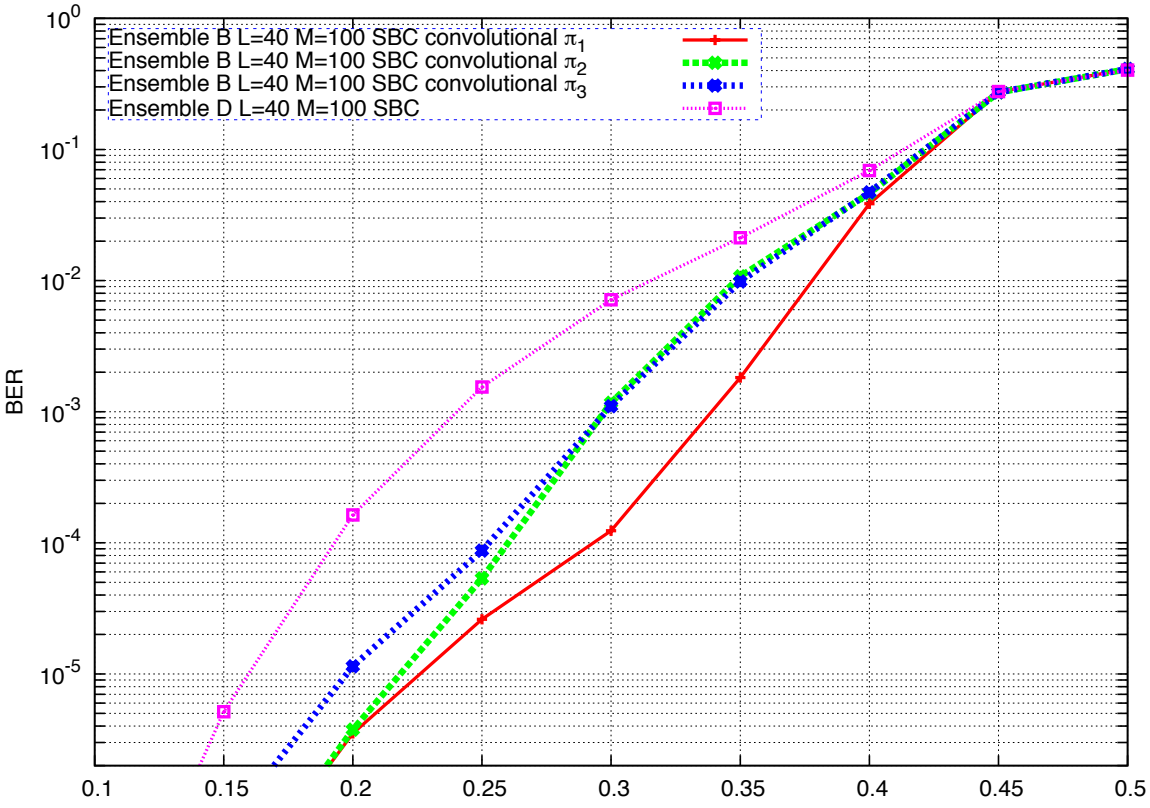

Figure 4.15. BER performance of code ensemble D with no interleaving scheme and code ensemble B under convolutional interleavers π_1, π_2, and π_3 over the SBC (random burst position).

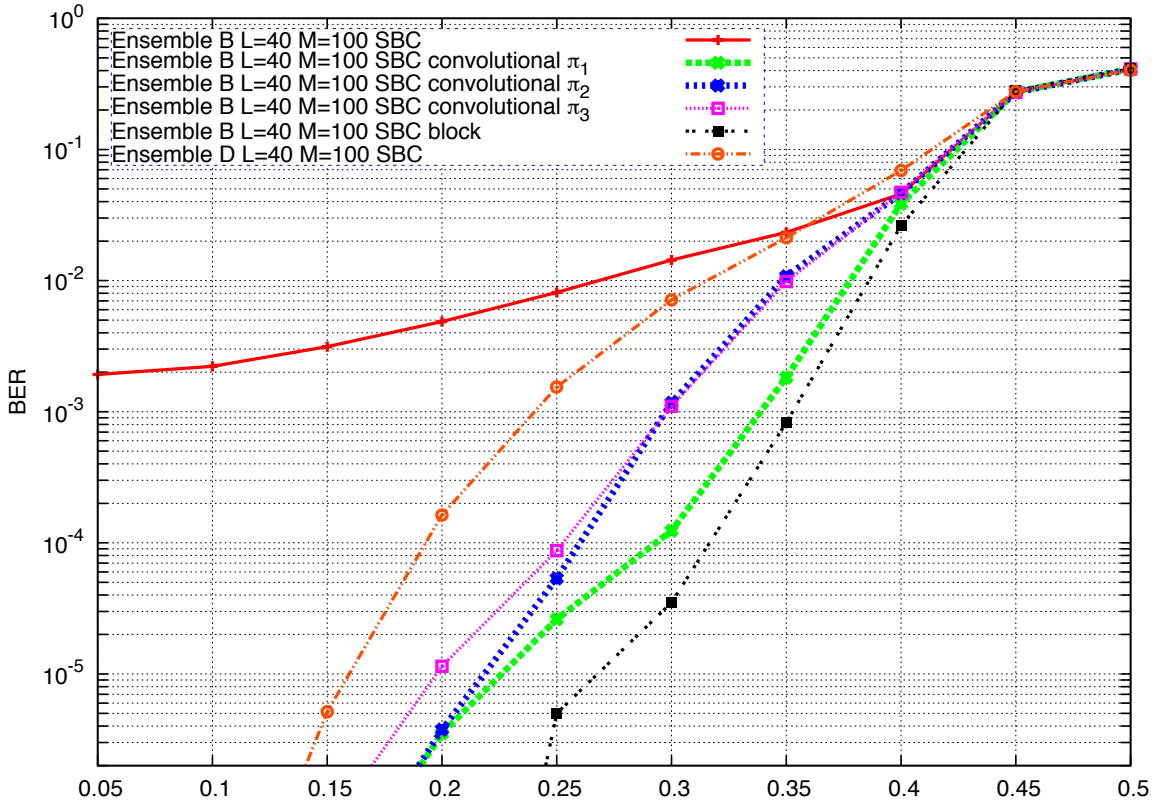

Figure 4.16. BER performance of codes ensembles B and D with no interleaving schemes and also code ensemble B under both convolutional and the block interleaver schemes over the SBC (random burst position).

5. CONCLUSION

This thesis outlines the spatially-coupled LDPC codes which have recently gained major research interest. In addition to their relatively simple structure and low-delay window decoding scheme, they have been shown to exhibit threshold saturation over BEC. However, their performance degrades drastically over erasure channels with memory, e.g. SBC. We address this problem and a system which considerably improves the performance while keeping the delay low is proposed.

We start with searching for the best SC code ensembles with minimal coupling by DE analysis. Some code design rules and criteria are established based on the results for practical applications over the BEC, which may also be consistent in other types of communication channels.

It has been observed that the performance of SC codes decreases extremely over correlated erasure channel where some adjacent symbols in the codeword can be effected by high-probable erasures. The reason is laid in the shape of the SC ensembles, which has a narrow constraint length. In the literature, mitigating the effects of a burst on the performance of SC codes is tried via some code design rules and criteria. However, these extra constraints limits the ability of considering other problematic issues in the structure of code.

In this thesis, we confront the effect of burst channel by enabling the SC communication system with convolutional interleaver. We do not take block interleaver into consideration as a solution since it introduces very large amount of delay which may ruin the low-delay characteristic of SC codes. We examine asymptotic performance of the system under three different convolutional interleavers by DE analysis. The same analysis is repeated for finite-length codeword. Based on the results it is observed that the more delay of convolutional interleaver is, the more performance of the code increases. Furthermore, it is also inferred that the coding performance under the weakest convolutional interleaver scheme with almost negligible delay, still outper-

forms the approach purely based on code design criteria. Coding performance under block interleaver is also evaluated but as it was mentioned, performance gain of block interleaver is not worth to produce tens of times of more delay.

All in one, as the main contribution in this thesis, it has been shown that employing a convolutional interleaver while transmitting SC coded data over correlated erasure channel is an efficient approach which both mitigates data degradation and keeps the delay low. However, there are several open problems related to this set up which require appropriate analysis. Design and utilization of more efficient interleavers can further improve the coding performance. Performance evaluation of different SC code ensemble other than ensemble B under the proposed system can be a potential research direction for future works. Another candidate for ongoing research is employing the proposed system over other types of practical communication channel, e.g. fading channel.

REFERENCES

1. Shannon, C. E., "A Mathematical Theory of Communication", *Bell Systems Technical Journal*, Vol. 27, pp. 379–423, 1948.

2. Lee, W., "The Advantages of Using Repetition Code in Mobile Radio Communications", *IEEE Vehicular Technology Conference*, Vol. 36, pp. 157 – 161, 1986.

3. Berrou, C., A. Glavieux and P. Thitimajshima, "Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes", *Proceedings of the IEEE International Conference on Communications.*, pp. 1064–1070, 1993.

4. Gallager, R. G., "Low-Density Parity-Check Codes", *IRE Transactions on Informormation Theory*, Vol. IT-8, pp. 21–28, 1962.

5. MacKay, D. J. C. and R. M. Neal, "Near Shannon Limit Performance of Low-Density Parity-Check Codes", *Electronics Letters*, Vol. 32, No. 18, pp. 1645–1646, 1996.

6. Chung, S. Y., G. D. Forney, Jr., T. J. Richardson and R. L. Urbanke, "On the Design of Low-Density Parity-Check Codes within 0.0045 dB of the Shannon Limit", *IEEE Communications Letters*, Vol. 5, pp. 58–60, 2001.

7. Tanner, R. M., "A Recursive Approach to Low Complexity Codes", *IEEE Transactions on Information Theory*, Vol. IT-27, pp. 533–547, 1981.

8. Richardson, T. J. and R. L. Urbanke, "Efficient Encoding of Low-Density Parity-Check Codes", *IEEE Transactions on Information Theory*, Vol. IT-47, No. 2, pp. 638–656, 2001.

9. Miladinovic, N. and M. Fossorier, "Improved Bit-Flipping Decoding of Low-Density Parity-Check Codes", *IEEE Transactions on Information Theory*, Vol. 51, No. 4,

pp. 1594 – 1606, 2005.

10. Gallager, R. G., *Low-Density Parity-Check Codes*, M.I.T. Press, Cambridge, MA, 1963.

11. Vontobel, P. O. and R. Koetter, "On the Relationship Between Linear Programming Decoding and Min-Sum Algorithm Decoding", *Proceedings of International Symposium on Information Theory and its Applications*, pp. 991–996, 2006.

12. Richardson, T. J. and R. L. Urbanke, "The Capacity of Low-Density Parity-Check Codes under Message-Passing Decoding", *IEEE Transactions on Information Theory*, Vol. IT-47, pp. 599–618, 2001.

13. Jiménez-Feltström, A. and K. Sh. Zigangirov, "Time-Varying Periodic Convolutional Codes with Low-Density Parity-Check Matrix", *IEEE Transactions on Information Theory*, Vol. IT-45, pp. 2181–2191, 1999.

14. Johannesson, R. and K. Sh. Zigangirov, *Fundamentals of Convolutional Coding*, IEEE Press, Piscataway, NJ, 1999.

15. Bates, S., L. Gunthorpe, A. E. Pusane, Z. Chen, K. Sh. Zigangirov and D. J. Costello, Jr., "Decoders for Low-Density Parity-Check Convolutional Codes with Large Memory", *Proceedings of the IEEE Symposium on Circuits and Systems*, 2006.

16. Mitchell, D. G. M., A. E. Pusane, N. Goertz and D. J. Costello, Jr., "Free Distance Bounds for Protograph-Based Regular LDPC Convolutional Codes", *Proceedings of International Symposium on Turbo Codes and Related Topics*, 2008.

17. Pusane, A. E., A. Jiménez-Feltström, A. Sridharan, M. Lentmaier, K. Sh. Zigangirov and D. J. Costello, Jr., "Implementation Aspects of LDPC Convolutional Codes", *IEEE Transactions on Communications*, Vol. 56, No. 7, pp. 1060–1069, 2008.

18. Pusane, A. E., *Analysis of LDPC Convolutional Codes Derived from LDPC Block Codes*, Ph.D. Thesis, University of Notre Dame, Notre Dame, Indiana, USA, 2008.

19. Lentmaier, M. and G. P. Fettweis, "On the Thresholds of Generalized LDPC Convolutional Codes Based on Protographs", *Proceedings of the IEEE International Symposium on Information Theory*, pp. 709–713, 2010.

20. Uchikawa, H., K. Kasai and K. Sakaniwa, "Spatially Coupled LDPC Codes for Decode-and-Forward in Erasure Relay Channel", *Proceedings of the IEEE International Symposium on Information Theory*, pp. 1474 –1478, 2011.

21. Rathi, V., R. Urbanke, M. Andersson and M. Skoglund, "Rate-Equivocation Optimal Spatially Coupled LDPC Codes for the BEC Wiretap Channel", *Proceedings of the IEEE International Symposium on Information Theory*, pp. 2393 –2397, 2011.

22. Kudekar, S., T. Richardson and R. Urbanke, "Threshold Saturation via Spatial Coupling: Why Convolutional LDPC Ensembles Perform so Well over the BEC", *IEEE Transactions on Information Theory*, Vol. 57, No. 2, pp. 803 –834, 2011.

23. Thorpe, J., "Low-Density Parity-Check (LDPC) Codes Constructed from Protographs", *JPL INP Progress Report*, Vol. 42-154, 2003.

24. Iyengar, A. R., M. Papaleo, P. H. Siegel, J. K. Wolf, A. Vanelli-Coralli and G. E. Corazza, "Windowed Decoding of Protograph-Based LDPC Convolutional Codes over Erasure Channels", *IEEE Transactions on Information Theory*, Vol. 58, No. 4, pp. 2303–2320, 2012.

25. Sridharan, A., M. Lentmaier, K. Sh. Zigangirov and D. J. Costello, Jr., "Terminated LDPC Convolutional Codes with Thresholds Close to Capacity", *Proceedings of the IEEE International Symposium on Information Theory*, pp. 1372–1376, 2005.

26. Iyengar, A. R., M. Papaleo, G. Liva, P. H. Siegel, J. K. Wolf and G. E. Corazza, "Protograph-Based LDPC Convolutional Codes for Correlated Erasure Channels", *Proceedings of the IEEE International Conference on Communications*, pp. 1–6, 2010.

27. Vafi, S. and T. Wysocki, "Weight Distribution of Turbo Codes with Convolutional Interleavers", *IET Communications*, Vol. 1, No. 1, pp. 71 –78, 2007.