Sparse Codes on Graphs with Convolutional Code Constraints

Umar Farooq, Muhammad

2022

Sparse Codes on Graphs with Convolutional Code Constraints

Doctoral Dissertation

Muhammad Umar Farooq

LUND UNIVERSITY

Department of Electrical and Information Technology
Lund University, Lund, Sweden
October, 2022

Department of Electrical and Information Technology
Lund University
Box 118, SE-221 00 LUND
SWEDEN

This thesis is set in Computer Modern 10pt
with the LaTeX Documentation System

Series of licentiate and doctoral theses
ISSN 1654-790X150
ISBN: 978-91-8039-449-9 (print)
ISBN: 978-91-8039-450-5 (pdf)

Printed in Sweden by *Tryckeriet i E-huset*, Lund.
Oct 2022.

In the loving memory of my father

Riaz ud Din

Abstract

Modern coding theory is based on the foundation of the sparse codes on graphs, such as the low-density parity-check (LDPC) codes, and the turbo-like codes (TCs) with component convolutional codes. The success of the LDPC codes and the TCs lies in their ability to perform low-complexity iterative message passing decoding procedures. The iterative message passing decoders that exchange messages probabilities, or beliefs, within the code graph are known as the belief propagation (BP) decoders. The BP decoders are sub-optimal, whereas maximum-a-posteriori (MAP) decoders for these codes are computationally infeasible. These codes can be optimized for their BP decoding performance, which improves their error rate performance in the waterfall region at the cost of a performance loss in the error-floor region. On the contrary, optimizing these codes for the MAP performance results in an improved performance in the error-floor region at the expense of a degraded performance in the waterfall region.

In practice, the BP decoding performance of the LDPC codes and the TCs, in the asymptotic block length regime, is determined by computing their BP decoding thresholds from the *density evolution* (DE), or the *extrinsic information transfer* (EXIT) chart techniques. The MAP decoding thresholds can be obtained with an application of the *area theorem* to the BP decoder performance. The graphs of the LDPC codes and the TCs are optimized for the BP, and the MAP decoding performance by using their decoding thresholds. For very large block lengths, spatially coupled (SC) versions of LDPC codes, and the TCs—which are optimized for MAP decoding performance—were shown to achieve excellent BP decoding performance in both the waterfall and the error-floor region, thanks to the *threshold saturation*. However, the BP decoding performance of these spatially coupled codes suffer from a high error-floor at a moderate to short code block length.

BP and MAP decoding thresholds of TCs on a binary erasure channel (BEC) have previously been investigated via the DE analysis. The capacity achieving SC-TCs were determined, where the underlying TCs were optimized for the BP and MAP performance. The TCs ensembles—parallel concatenated codes, serially concatenated codes, braided convolutional codes, and hybrid concatenated codes—with varying component convolutional codes strengths were considered in these investigations.

This thesis focuses on investigating the BP decoding performance of SC-TCs—which were earlier investigated for the BEC—on an additive white Gaussian noise (AWGN) channel. Furthermore, the problem of high error-floor of SC-TCs for short to moderate block lengths is investigated and addressed with a design of an optimized convlutional permutor in constructing the SC-TCs. Fi-

nally, the connection of the TCs and the LDPC codes is explored by introducing a family of convolutional codes (CC) based generalized LDPC codes (GLDPCs). These research areas are summarized under the following three topics.

In the first topic, we have computed the iterative decoding thresholds of SC-TCs on the AWGN channel via the Monte Carlo density evolution (MC-DE) methods. The MC-DE methods are time consuming, which has motivated us to introduce an efficient alternative that predicts the AWGN thresholds of SC-TCs with the knowledge of their BEC thresholds. The results show that the estimated thresholds via the MC-DE method and the predicted thresholds are very close for the capacity achieving randomly punctured SC-TCs. For the high rate uncoupled TCs, which are obtained by randomly puncturing their mother code, the predicted thresholds are improved by incorporating the estimated AWGN threshold of the mother code ensemble into the threshold prediction method.

In the second topic, we have introduced the design of a single block-wise periodic time-varying convolutional permutor to construct the SC-TCs. The convolutional permutor is designed by applying the unwrapping procedure to an optimized block permutor, which optimize the bit error rate (BER) performance of a TC in an error-floor region. We showed that a convolutional permutor obtained via the unwrapping procedure inherits the properties of its parent permutor. Due to this reason, the BER performance of block-wise periodically time-varying convolutional permutor based SC-SCCs does not suffer from a high error-floor problem at short block lengths, which was demonstrated through the simulation results.

In the third topic, we have introduced the families of regular and irregular CC-GLDPCs. The CC-GLDPCs enabled us to connect TCs and LDPC codes in terms of their graph structures. The BEC thresholds and the minimum distance properties of the regular CC-GLDPCs were compared to the regular LDPC codes. Furthermore, we performed an exhaustive grid search using the BEC thresholds of the class of CC-GLDCPs, and determined the design configurations of optimized CC-GLDPCs. The results suggest that, for regular graphs, it is possible to find a sparser CC-GLDPCs than the LDPC codes at the expense of a slightly negligible loss in the performance. Furthermore, the BP optimized CC-GLDPC is observed to have a better BP and MAP thresholds than the turbo codes on the BEC.

Popular Science Summary

We are living in an information age where digital information is readily available to anyone with a mobile phone or computer with an internet connection. This has enabled us to video call each other, watch a favorite TV-show, play a video game online with friends, shop online, etc, at any time we want to. Slowly all functions of the society are moving to the digital network. For example, mobiles, smart homes, offices, cars, etc, are becoming part of the networked society.

The information in the digital network is kept somewhere on storage systems for an on-demand access later. Due to technological advancements, the storage capacity is increasing yearly, and so does the speed at which information can be transmitted from one place to another. In the network, such advancements lead to an increased amount of information processing, and consequently an increased amount of information transmissions. The network uses radio waves, wires, optical fibres or hard-disks to transmit information from one end to another. All of these different physical media introduce errors to the transmitted information. Such erroneous transmissions compromise the integrity of the digital network, and therefore it is critical for it to reliably transmit information between two ends at all situations.

To achieve reliable communication, every digital communication device in the network uses a channel code to correct the errors that happen during the information transmission. The fundamental limit of reliable communication, theoretically achievable through channel coding, was defined by Shannon in his landmark paper in 1948. Since 1950, many classes of channel codes have been introduced. Among these are the turbo and LDPC codes that perform close to the fundamental limits with moderate operational complexity. These codes have become part of satellite, Wi-Fi, LTE and 5G communication networks.

The strength of turbo and LDPC codes stems in their ability to perform low-complexity iterative message passing decoding, that corrects the errors in the received data in an iterative fashion. Iterative message-passing decoders are sub-optimal, but they have the potential to perform very close to the optimal decoders, which are infeasible due to their high operational complexity. The gains of long codes with sub-optimal decoding, however, come with some side-effects.

A generalized class of turbo codes, known as the turbo-like codes (TCs), has the potential to compensate for these side-effects leading to a degraded sub-optimal decoding performance. On a simple erasure channel, it has been shown that spatially coupled (SC)-TCs, that are constructed by combining several TCs in a structured way, are capable of achieving an optimal performance with no side-effects using sub-optimal decoders. But what about SC-TCs on more

complicated channels such as the Gaussian channel? The research focus of this thesis can be summarized as *performance analysis of SC-TCs on a Gaussian channel*, and *a unified view of TCs and LDPC codes*.

Both TCs and LDPC codes belong to the class of sparse codes with graph structures. The code graph of LDPC codes consists of a large number of weak local decoders, whereas the code graph of TCs consists of a few more powerful local decoders. These local decoders engage with each other in an iterative manner to correct the errors. Except the local decoders, both TCs and LDPCs share a common structure. However, due to the way they were represented, both of these codes were viewed as different classes of codes. A unified view of TCs and LDPCs based on graphical models enables us to perform one-one comparison of both classes of codes, and develop the same techniques in their design and analysis.

Contribution Statement

This doctoral thesis summarizes my academic work as a Ph.D. student, and it contains two parts. The first part gives an overview of the research field in which I have been working during my Ph.D. studies and a brief summary of my contributions in the research field. The second part is composed of five following papers listed in chronological order:

- **M. U. Farooq**, S. Moloudi and M. Lentmaier, "Thresholds of Braided Convolutional Codes on the AWGN Channel", *2018 IEEE International Symposium on Information Theory (ISIT)*, Vail, CO, USA, 2018.
- **M. U. Farooq**, S. Moloudi and a. M. Lentmaier, "Generalized LDPC Codes with Convolutional Code Constraints", *2020 IEEE International Symposium on Information Theory (ISIT)*, LA, USA, 2020.
- **M. U. Farooq**, A. G. i. Amat and M. Lentmaier, M. "Threshold Computation for Spatially Coupled Turbo-Like Codes on the AWGN Channel", *Entropy* 2021, 23, 240.
- **M. U. Farooq**, A. G. i. Amat and M. Lentmaier, "Spatially-Coupled Serially Concatenated Codes with Periodic Convolutional Permutors," *2021 11th International Symposium on Topics in Coding (ISTC)*, Montreal, Canada, 2021.
- **M. U. Farooq**, A. G. i. Amat, and M. Lentmaier, "Improving the Thresholds of Generalized LDPC Codes with Convolutional Code Constraints", *To be submitted to IEEE Communication Letters*, 2022.

For all papers in this list, I have been the first author under the supervision of my supervisors. As a first author, I have carried out the main part of work which involved following major activities: problem formulation, analysis, numerical computations, and writing. During the Ph.D. studies, I have also contributed to the following publication that is not included in this thesis.

- M. Mahdavi, **M. Umar Farooq**, L. Liu, O. Edfors, V. Öwall and M. Lentmaier, "The Effect of Coupling Memory and Block Length on Spatially Coupled Serially Concatenated Codes", *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1-7.

The work done during this PhD has been supported by the Swedish Research Counsel.

Acknowledgments

A PhD journey is a tough one, and if I have reached somewhere in it, then it is because of the encouragement from many people. I like to express a special thanks to all of them.

First and foremost, I am extremely grateful to Michael Lentmaier, my supervisor, for believing in me throughout the PhD studies. Thank you Michael, for giving me this unique learning opportunity, for walking me through this exciting field of research, and for helping and supporting me, whenever I needed it.

Thank you Alexander Graell i Amat for supporting me in the research activities with valuable suggestions and feedbacks.

Thank you Ove Edfors for your support during my Ph.D studies.

I am very grateful to my colleagues and friends from the department: Juan Vidal, Christopher, Jonathan Noxet, Andreia, Hu sha, Jing, Guoda, Saeedeh, Makambi, Sara, Sidra, Christian Nilsson, Anders, Joao, Muris, Fredrik Rusek, Fredrik Tufvesson, Michiel, Jesus, Erik Bengtsson, Dino, Aleksei, Neharika, Hedieh, Ashkan, Juan Sanchez, Russell, Junshi, Xuesong, Vincent, Wei, Arturo, Lucas, Masoud and everyone, thank you for sharing good times. Your friendship and company is one of the best part of my stay at the EIT during these years.

Finally, I would like to express my deepest gratitude to my family. Thank you for your support, encouragement, love, and care. This thesis is dedicated to you: Abu, Ame, Hassan, Khadija, Nayab, Hanya, and Ibrahim.

Muhammad Umar Farooq

List of Acronyms

AWGN Additive White Gaussian Noise

AAP A-posteriori Probability

BCC Braided Convolutional Code

BER Bit Error Rate

BCC Braided Convolutional Code

BP Belief Propagation

BEC Binary Erasure Channel

CC Convolutional Code

CN Check/Constraint Node

dB Decibel

DE Density Evolution

EBP Extended belief propagation

EXIT EXtrinsic Information Transfer

GLDPC Generalized LDPC

HCC Hybrid Concatenated Code

IO-WEF Input-Output Weight Enumerator Function

IP-WEF Input-Parity Weight Enumerator Function

LDPC Low-Density Parity-Check

LLR Log-Likelihood Ratio

MAP Maximum A-posteriori

ML Maximum Likelihood

PCC Parallel Concatenated Code

pdf Probability Density Function

RA Repeat-Accumulate Code

SE Single-Edge

SCC Serially Concatenated Code

SC Spatially Coupled

SNR Signal-to-Noise Ratio

SISO Soft Input Soft Output

TC Turbo-like Code

VN Variable Node

WEF Weight Enumerator Function

Contents

Part I

Overview of Research Field

Chapter 1

Introduction

Channel coding [1], which is also known as error control coding, is an essential block in the design of digital communication networks. A communication network consists of transmitters, receivers and channels that connect these transmitters and receivers. A channel may introduce errors during message transmissions, which affects the reliability of the received messages. Channel codes are used by the networks to detect and correct the errors that occur during message transmissions.

Error control coding adds redundancy to the information bit sequence. Due to this redundancy, the errors that occur in the transmissions can be detected and possibly corrected. The task of the channel code designer is to suggest a rule that maps a set of information sequences to the codeword sequences, which contain redundancy. In addition to the definition of a rule—or a code—a suitable method, which recovers the information sequence from the received codeword sequence with a highest probability, is required as well. A channel *encoder* implements the rule to map the information and codeword sequences, which are transmitted by a communication system over a channel. The errors in the noisy received message are corrected at the receiver with the help of a suitable *decoding algorithm*, which is implemented by a *decoder*. A repetition code is an example of a simple channel code, which creates multiple copies of the data. For instance, if a letter 'A' is to be transmitted, then the encoder of a repetition code of length four generates the codeword 'AAAA', which is then sent to the receiver. Suppose the noisy message at the receiver appears to be 'AxAy'. Then a decoder with a simple decoding algorithm at the receiver could decide that the message 'A' was transmitted based on the observation that 'A' has occurred twice in the received message, whereas the letters 'x' and 'y' have occurred only once.

Shannon, in his landmark paper [2] on 1948, defined the fundamental limits on the minimum amount of redundancy required for reliable communication achievable through codes with large block lengths. However, the design of codes that could perform close to these limits with low to moderate decoding complexity was left open. Since 1950s, many strong channel codes have been introduced to the field, which are classified into the families of block, or convolutional codes [3]. The strength of a code is determined by its error correction and detection capability, which is associated with the minimum distance of a code. Among these strong codes are turbo codes [4], and LDPC codes [5], [6] that

have modernized the field.

Turbo and LDPC codes belong to the class of codes defined by graphs [7]. These graphs consist of variable nodes (VNs), associated with the code symbols, and constraint nodes (CNs), which generate redundancy. The VNs and CNs in a graph are connected via a set of edges. The degrees of CNs and VNs are defined by the number of edges these nodes are connected to. In a regular graph, each node type has the same number of degrees, whereas in an irregular graph, the degrees of each node type may differ. Such a graphical representation facilitates the implementation of low-complexity iterative massage passing decoding procedures. The belief propagation (BP) decoding algorithm is commonly used to perform an iterative message passing decoding procedure, in which the beliefs, or the probabilities, of the code symbols are exchanged between the VNs and CNs of a code graph in an iterative fashion. Iterative BP decoders are sub-optimal due to the presence of *cycles* in the graph. For a large enough cycle lengths, these BP decoders may perform very close to the optimal decoders, which are infeasible due to their high operational complexity.

In the design phase of codes on graphs, an adequate code performance is of interest in the following regions: the water-fall region, defined by a low signal-to-noise ratio (SNR), and the error-floor region, defined by a high SNR. For a BP decoding procedure under an asymptotic code block length, the strength of these codes in the water-fall region is determined by computing their thresholds via density evolution (DE) or extrinsic information chart (EXIT) [8] techniques. A code threshold partitions the channel space into two regions: in one region, reliable communication is possible, whereas in the other region, reliable communication is not be possible. The strength of these codes in the error-floor region is determined through their weight enumerator functions (WEF), which provides insights into minimum distance properties of these codes with varying block lengths. Minimum gap between the code thresholds and the fundamental Shannon limits, as well as increasing minimum distance with an increasing block length, are often the desirable features during the code design phase.

Both LDPC codes and turbo codes perform close to the fundamental limits for very large block lengths. However, on medium to short block lengths, the decoding performance of their BP decoders deteriorates. Even the maximum-a-posterior (MAP) decoders of both families of codes exhibit error-floor at short block lengths, in which the *bit error rate* (BER) performance of these codes improve slowly with increasing signal-to-noise ratio (SNR). Optimizing the code performance in one region may lead to worst performance in the other. For example, regular LDPCs of low VNs degree have strong BP thresholds, but relatively weaker MAP thresholds and weaker minimum distance properties. Increasing the VNs degree degrades the BP decoding performance, but it improves the MAP decoding performance and improves the minimum distance properties. This situation leads a code designer to navigate a complicated space of design trade-offs.

Turbo codes are parallel concatenated codes (PCC) with two component convolutional codes (CCs), where the component CCs are separated by a permutor. The design of a permutor with large enough *spread* is essential in having strong minimum distance properties. A generalized class of turbo codes, known as the turbo-like codes (TCs), were investigated by Moloudi *et al.* [9, 10]. Compared to PCCs, TCs ensembles, including serially concatenated codes (SCCs), braided convolutional codes (BCCs) and hybrid concatenated codes (HCCs),

were observed to have stronger MAP performance, stronger minimum distance properties, and poor BP decoding performance. The TCs with strong MAP performance, and strong minimum distance properties are attractive for *the spatial coupling* technique, which is one of the powerful capacity-achieving code construction technique.

Spatially coupled TCs (SC-TCs) were constructed by combining a sequence of TCs in a structured way through several independent block permutors [9]. On a binary erasure channel (BEC), it was proven analytically and observed numerically that the sub-optimal BP decoding performance of a SC-TC ensemble achieves the MAP decoding performance of its underlying TC for very large block lengths. In literature, this remarkable observation is known as the *threshold saturation* [11]. It was further proved in [12] that under certain conditions on the permutors, SC-TCs either preserve or improve on the minimum distances of their underlying TCs. For these reasons, spatially coupled ensembles hold a significant potential as these can be optimized for performance in both the water-fall and the error-floor regions simultaneously.

The objective of this thesis is to investigate the thresholds of TCs (SC-TCs) on the AWGN channel, to introduce a single convolutional permutor based design of SC-TCs instead of several independent block permutors, and to understand the connection of TCs to LDPC codes. This thesis includes five papers. In Paper I and Paper II, the time-consuming Monte-Carlo DE (MC-DE) methods are used to compute the AWGN thresholds of randomly punctured TCs, and SC-TCs, with iterative BP decoders. An efficient alternative to predict the AWGN thresholds of these TCs is introduced, and the predicted thresholds are compared with the MC-DE thresholds. In Paper III, a block-wise periodic time-varying convolutional permutor based design of SC-SCCs is introduced to alleviate the error-floor problem of independent permutors based SC-SCCs at short block lengths. In Paper IV and Paper V, a class of generalized LDPC codes with convolutional code constraints, abbreviated as CC-GLDPCs, is introduced. The performance of regular CC-GLDPCs in the water-fall and the error-floor regions is investigated, and it is compared with the performance of the corresponding LDPC code graphs. The thresholds of CC-GLDPCs are optimized by introducing irregularities in their graphs. BER performance of the optimized CC-GLDPCs is compared with their iterative decoding thresholds.

1.1 Thesis Outline

This thesis is written in paper collection format and consists of two parts. The first part provides an overview of the coding theory field in relation to the research contributions of this thesis, and the second part provides the selection of research papers as research contributions. The first part of the thesis contains eight chapters. Chapter 2 explains fundamental coding concepts, and an overview of block and convolutional codes, as well as their optimal decoding schemes. Chapter 3 explores the idea of codes on graphs, and their iterative decoding principles. Chapter 4 details the techniques to determine the decoding performance of codes on graphs in the water-fall region. Chapter 5 covers the techniques to determine the performance of codes on graphs in the error-floor region. Chatper 6 gives a brief overview of the construction and the performance analysis of TCs. Chapter 7 explains the concepts of spatial coupling

with an example of SC-LDPCs, and SC-BCCs. Chapter 8 summarizes the included papers, as well as the main conclusion of the thesis, and suggests some future research areas.

Chapter 2

Preliminaries

In this chapter, an information theoretic view of channel coding is presented. Furthermore, encoding and optimal decoding of linear block codes and convolutional codes are discussed.

2.1 Binary Input Memoryless Channel

Model of a typical communication system is shown in Figure 2.1. Consider a binary information source $U \in \{0, 1\}$. Information bits in sequence $\boldsymbol{U}$ are encoded via the channel encoder to a binary codeword sequence $\boldsymbol{C}$, which is then mapped to a sequence $\boldsymbol{X}$, $X \in \mathcal{X} = \{+1, -1\}$. The mapped sequence is next transmitted over a channel.

A channel with binary input has an input alphabet with two symbols, $X \in \{+1, -1\}$. The channel produces a noisy sequence $\boldsymbol{Y}$ at its output according to some channel transition probability distribution $P_{Y|X}(Y = y|X = x)$. The channel output Y has an alphabet $\mathcal{Y}$, which could be a finite set or a continuous range of values. After receiving the noisy sequence $\boldsymbol{Y}$, the channel decoder generates the hard decision $\hat{\boldsymbol{U}}$ for the information bits.

2.1.1 Binary Erasure Channel

A binary erasure channel (BEC) is the simplest among the channels used to study the communication system model. The BEC channel either erase the symbol at its input with a probability of erasure ε, or let it through perfectly with a probability $1 - \varepsilon$. The channel has input alphabets $\mathcal{X} = \{+1, -1\}$,

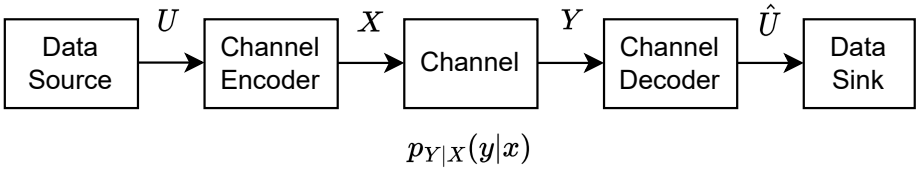


Figure 2.1: Communication System Model.

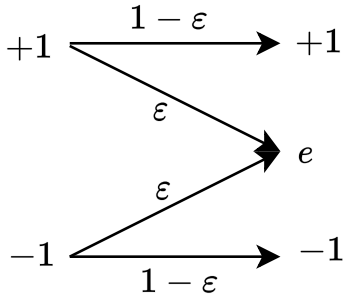


Figure 2.2: Binary erasure channel with erasure probability ε.

output alphabets $\mathcal{Y} = \{+1, \mathrm{e}, -1\}$ and probability mass functions (pmf), which are labeled along the edges from the input to the output in Figure 2.2.

Investigations of coded systems using the BEC have their merits in terms of mathematical simplicity and broader understanding but these are not well suited for many real world scenarios of communication systems.

2.1.2 Binary Memoryless Symmetric Channel

Consider a discrete time channel model

$$Y_t = X_t + N_t, \tag{2.1}$$

where t is a discrete time index, $X_t \in \mathcal{X}$, and N_t is sampled from Gaussian distribution $\mathcal{N}(0, \sigma^2)$. For this model,

$$P(Y_t|X_t) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(Y_t - X_t)^2}{2\sigma^2}\right) \tag{2.2}$$

is the probability density of the binary-input additive white Gaussian noise (BI-AWGN) channel. For length N sequences $\boldsymbol{X} = (X_1, ...X_N)$ and $\boldsymbol{Y} = (Y_1, ...Y_N)$, the AWGN channel is memoryless when

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{t=1}^{N} P(Y_t|X_t), \tag{2.3}$$

and symmetric when

$$P(Y_t = y|X_t = +1) = P(Y_t = -y|X_t = -1). \tag{2.4}$$

In this thesis, the BI-AWGN channel is parametrized with the unit energy per information bit, on average, versus the noise power spectral density (E_b/N_0) for the studies of considered codes.

2.2 Entropy, Mutual Information and Channel Capacity

Channel capacity plays a fundamental role in the assessment and comparison of the strength of various coding schemes. Channel capacity is linked to the

entropy and mutual information. These concepts are described in the following subsections.

2.2.1 Entropy

Entropy is the number of average bits required to describe a random variable. It is also used as a measure to quantify the average uncertainty of a random variable. Entropy of a discrete random variable (RV), $X \in \mathcal{X}$, with a probability mass function (pmf) $P(x)$ is obtained from

$$H(X) = -\sum_{x \in \mathcal{X}} P(x)\log(P(x)).$$

Entropy of a continuous RV X with a probability density function $f(x)$ is obtained from

$$h(X) = -\int_{\mathcal{S}} f(x)\log(f(x))dx,$$

here $\mathcal{S}$ is the support set of the RV X. Let Y be another discrete RV, then the conditional entropy of X given Y, $H(X|Y)$, is defined as

$$H(X|Y) = -\sum_{x \in \mathcal{X}} P(x) \sum_{y \in \mathcal{Y}} P(y|x)\log(P(y|x)),$$

and it describes the uncertainty of X given the knowledge of Y. Similarly, $H(X,Y)$—the joint entropy of X and Y—can be defined. The chain rule of entropy is useful in the analysis of joint entropy, which is

$$H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y).$$

2.2.2 Mutual Information

Consider two discrete RVs X and Y, the reduction in uncertainty of X given the knowledge of Y is defined as the mutual information. The mutual information $I(X,Y)$ is computed by using

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X).$$

Conditional mutual information of two RVs X and Y, given the knowledge of a RV Z, is computed by using

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z).$$

Mutual information of continuous RVs can be computed in a similar manner as described above.

2.2.3 Channel Capacity

Channel capacity governs the constraint under which a reliable transmission over a noisy channel is possible. For a discrete memoryless channel (DMC) with input and output alphabets $\mathcal{X}$ and $\mathcal{Y}$, and a set of channel crossover probabilities $P(y|x)$, the channel capacity C is defined as [2, 13]

$$C = \max_{P(x)} I(X;Y), \tag{2.5}$$

where $I(X;Y)$ is maximized over all the possible input probability distributions $P(x)$. Shannon further describes the concept of channel coding in his paper [2]. For Shannon's channel coding theorem, basic definition of (M, N) code for the channel $(X, P(y|x), Y)$ [13] is:

1. A message W is drawn from the index set $\{1, 2, \cdots, M\}$.
2. An encoding function maps the message W to a bit vector $\boldsymbol{x}(W)$ of length N, call the mapping as $f : \{1, 2, \cdots, M\} \to \mathcal{X}^N$. The set of M codewords $\{\boldsymbol{x}(1), \boldsymbol{x}(2), \cdots, \boldsymbol{x}(M)\}$ constructs the codebook.
3. A decoding function $g : \mathcal{Y}^N \to \{1, 2, \cdots, M\}$ assigns an estimate of the index to each received noisy sequence.

The *code rate* R for (M, N) code is defined as

$$R = \frac{\log_2 M}{N}.$$

Shannon's channel coding theorem defines the constraint under which reliable communication is possible. It states that for a DMC, all rates below the capacity C are achievable. Shannon showed the existence of capacity approaching codes with vanishing error probability as $N \cdot R \to \infty$. The channel capacity for the BEC is

$$C_{\text{BEC}} = 1 - \varepsilon,$$

and for the AWGN channel is defined as

$$C_{\text{AWGN}} = -\int_{-\infty}^{+\infty} p(y) \log_2(p(y)dy - 0.5 \log_2(2\pi e\sigma^2).$$

2.3 Block Codes

Block codes are defined by mapping a block of information bits denoted as $\boldsymbol{u} = (u_1, u_2, ..., u_K)$ to a block of codeword bits denoted as $\boldsymbol{v} = (v_1, u_2, ..., u_N)$. The length of the information block is K, and of the codeword is N, where $N > K$. The code dimension K indicates that there are 2^K distinct information and codeword blocks. The set of all codeword blocks forms an (N, K) block code. $N-K$ number of redundant bits are used for the error correction and detection. The ratio $R = K/N$ is known as the *code rate*. The redundant or parity bits are generated by using an encoder. In practice, linear block codes—which is a sub-class of block codes—are used due to their efficient encoders realization. A block code of length N is called a linear block code $\mathcal{C}(N, K)$ if and only if its set of 2^K number of codewords forms a K-dimensional subspace of the vector space V of all N-tuples over $\mathbb{F}_2^N$. The encoding of linear block codes can be described by using the matrix multiplication as

$$\boldsymbol{v} = \boldsymbol{u}\boldsymbol{G}, \tag{2.6}$$

where $\boldsymbol{G} \in \mathbb{F}_2^{K \times N}$ is the generator matrix of size $K \times N$. The matrix $\boldsymbol{G}$ contains K linearly independent codewords which form the basis of $\mathcal{C}(N, K)$ subspace. Its *null* (or *dual*) space, denoted by $\mathcal{C}_d$, is obtained by

$$\mathcal{C}_d = \{\boldsymbol{h} \in V : \langle \boldsymbol{h}, \boldsymbol{v} \rangle = 0 \text{ for all } \boldsymbol{v} \in \mathcal{C}\}. \tag{2.7}$$

The dual space $\mathcal{C}_d$ can be treated as the binary $(N, N-K)$ linear block code and it is called the dual code of $\mathcal{C}$. The generator matrix $\boldsymbol{H}$ of the dual code $\mathcal{C}_d(N, N-K)$ contains $N-K$ linearly independent basis vectors that span the subspace $\mathcal{C}_d$. $\boldsymbol{H}$ is called the parity check matrix of $\mathcal{C}$, and it satisfies the following relationship

$$\boldsymbol{G}\boldsymbol{H}^{\mathrm{T}} = \mathrm{O}, \tag{2.8}$$

where O is a $K \times (N-K)$ zero matrix, and the equation is known as the syndrom condition.

Example 2.3.1. *A single parity-check (SPC) code is an example of a linear block code. An encoder of the SPC code compute one parity bit v by performing binary addition of the bits of $\boldsymbol{u}$. This resultant bit after the binary addition is then appended at the end of information block to form a codeword of $\mathcal{C}_{SPC}(K+1, K)$. An example of $\boldsymbol{G}$ and $\boldsymbol{H}$ for $\mathcal{C}_{SPC}(3,2)$ is*

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad \boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}.$$

The dual space of $\mathcal{C}_{SPC}$ defines a $(3,1)$ repetition code.

The *Hamming weight* $w_H(\boldsymbol{v})$ of a binary codeword $\boldsymbol{v}$ is equal to the number of non-zero elements. The *Hamming distance* of two binary sequences $\boldsymbol{v}$ and $\boldsymbol{w}$, denoted as $d_H(\boldsymbol{v}, \boldsymbol{w})$, is equal to the number of positions in which these sequences are different. The minimum distance d_{min} of a linear block code $\mathcal{C}$ is the smallest Hamming weight of any non-zero codeword.

Optimal maximum likelihood (ML) decoding of the received noisy sequence $\boldsymbol{r}$ is performed by maximizing the *likelihood function* $p(\boldsymbol{r}|\boldsymbol{v})$ as

$$\hat{\boldsymbol{v}} = \arg\max_{\boldsymbol{v} \in \mathcal{C}} p(\boldsymbol{r}|\boldsymbol{v}).$$

An ML decoder estimates an input sequence $\hat{\boldsymbol{u}}$.

2.4 Convolutional Codes

A convolutional encoder is a finite state machine, which takes in the message bits for encoding as a continuous stream. Let us denote by $\boldsymbol{u}$ and $\boldsymbol{v}$ the sequences of message and codeword as

$$\boldsymbol{u} = (\boldsymbol{u}_0, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_t, \cdots),$$

$$\boldsymbol{v} = (\boldsymbol{v}_0, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_t, \cdots),$$

subscript t above denotes the time index, $\boldsymbol{u}_t = (u_t^{(1)}, u_t^{(2)}, \cdots, u_t^{(k)})$ and $\boldsymbol{v}_t = (v_t^{(1)}, u_t^{(2)}, \cdots, v_t^{(n)})$. For a linear block code, the blockwise encoding of message block at time t is performed as $\boldsymbol{v}_t = \boldsymbol{u}_t\boldsymbol{G}$. A CC, on the other hand, generates a code block $\boldsymbol{v}_t$ as a linear function of several input blocks. This idea is described as

$$\boldsymbol{v}_t = f(\boldsymbol{u}_t, \boldsymbol{u}_{t-1}, \cdots, \boldsymbol{u}_{t-m}).$$

The value m is called the encoder memory and defines how many past messages are considered to generate a code block at time t. The linear mapping f for a general feed forward convolutional encoder is defined as

$$\boldsymbol{v}_t = \boldsymbol{u}_t\boldsymbol{G}_0 + \boldsymbol{u}_{t-1}\boldsymbol{G}_1 + \cdots + \boldsymbol{u}_{t-m}\boldsymbol{G}_m, \tag{2.9}$$

where $\boldsymbol{G}_l \in \mathbb{F}_2^{K\times N}$ is the generator matrix for the message block $\boldsymbol{u}_{t-l}$ for $l = 0, \cdots, m$. The encoding can be described in terms of matrix multiplication as

$$\boldsymbol{v} = \boldsymbol{u}\boldsymbol{G} \tag{2.10}$$

with

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_m & & \\ & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_m & \\ & & \ddots & \ddots & & \ddots \end{bmatrix}$$

Example 2.4.1. *An example of a rate-1/2, $m = 2$ feed forward encoder is shown in Figure 2.3. Its constituent generator matrices $\boldsymbol{G}_l$ are defined as $\boldsymbol{G}_0 =$ [11], $\boldsymbol{G}_1 =$ [10] and $\boldsymbol{G}_2 =$ [11].*

Alternatively, these generator matrices can be viewed as a generator vector $\boldsymbol{g}_i^{(j)} = \left(g_{i,0}^{(j)}, g_{i,1}^{(j)}, \cdots, g_{i,m}^{(j)}\right)$ whose entries define the generator coefficients of the input i and output j for $i = 1, \cdots, K$ and $j = 1, \cdots, N$. Following this, the convolutional encoding can be expressed as

$$\boldsymbol{v}^{(j)} = \boldsymbol{u}^{(1)} \circledast \boldsymbol{g}_1^{(j)} + \boldsymbol{u}^{(2)} \circledast \boldsymbol{g}_2^{(j)} + \cdots + \boldsymbol{u}^{(K)} \circledast \boldsymbol{g}_K^{(j)} = \sum_{i=1}^{K} \boldsymbol{u}^{(i)} \circledast \boldsymbol{g}_i^{(j)} \tag{2.11}$$

where $\circledast$ is a convolution operator.

Since convolution in time-domain is multiplication in the transform domain, it is convenient to transform the time sequence (2.11) by means of the delay operator D(D-transform) as

$$\boldsymbol{v}(D) = \boldsymbol{u}(D)\boldsymbol{G}(\boldsymbol{D}). \tag{2.12}$$

Input and output sequences of the convolutional encoder are

$$\boldsymbol{u}(D) = \boldsymbol{u}_0 + \boldsymbol{u}_1(D) + \boldsymbol{u}_2(D^2) + \cdots,$$

$$\boldsymbol{v}(D) = \boldsymbol{v}_0 + \boldsymbol{v}_1(D) + \boldsymbol{v}_2(D^2) + \cdots,$$

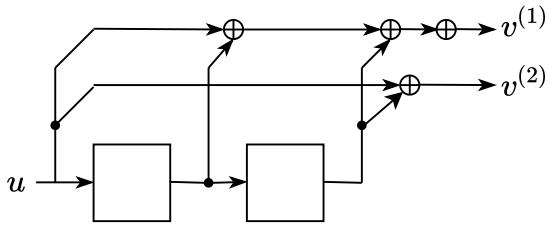


Figure 2.3: A rate-1/2 feed forward encoder of convolutional code.

and the generator matrix in terms of transformed generator matrix is

$$\boldsymbol{G}(D) = \boldsymbol{G}_0 + \boldsymbol{G}_1(D) + \boldsymbol{G}_2(D^2) + \cdots + \boldsymbol{G}_m(D^m),$$

which in terms of the generator polynomials $\boldsymbol{g}_i^{(j)}(D)$, is described as

$$\boldsymbol{G}(D) = \begin{bmatrix} \boldsymbol{g}_1^{(1)}(D) & \cdots & \boldsymbol{g}_1^{(N)}(D) \\ \boldsymbol{g}_2^{(1)}(D) & \cdots & \boldsymbol{g}_2^{(N)}(D) \\ \vdots & & \vdots \\ \boldsymbol{g}_K^{(1)}(D) & \cdots & \boldsymbol{g}_K^{(N)}(D) \end{bmatrix}.$$

In practice, each generator polynomial $\boldsymbol{g}_i^{(j)}(D)$ is represented in octal notation.

Example 2.4.2. *For the rate-1/2 encoder described in Example 2.4.1, the generator vectors can be expressed as*

$$\boldsymbol{g}_1^{(1)} = (1, 1, 1), \qquad \boldsymbol{g}_1^{(2)} = (1, 0, 1).$$

In terms of delay domain, these are written as

$$\boldsymbol{g}_1^{(1)}(D) = 1 + D + D^2, \qquad \boldsymbol{g}_1^{(2)}(D) = 1 + D^2.$$

The generator vectors of the considered example are expressed as $(7, 5)$ *in octal notation.*

The free distance d_{free} of a convolutional code $\mathcal{C}$ is defined as

$$d_{free} = \min_{\boldsymbol{v},\boldsymbol{v'} \in \mathcal{C}: \boldsymbol{v} \neq \boldsymbol{v'}} d_H(\boldsymbol{v}, \boldsymbol{v'}) = \min_{\boldsymbol{v},\boldsymbol{v'} \in \mathcal{C}: \boldsymbol{v} \neq \boldsymbol{0}} w_H(\boldsymbol{v}).$$

Similar to the d_{min} of block codes, d_{free} is the smallest of the Hamming weights of all non-zero codewords of the convolutional code.

CCs are typically described by a state transition diagram. When this state transition diagram is unrolled along the time t, then the trellis representation of the convolutional code is obtained. A trellis section at each time level has 2^m number of state nodes, where each node is connected via 2^K number of edges to the nodes of next time level. In practice, *terminated* CCs are used by starting and terminating the CC encoder in a zero state. At an expense of a rate loss, the terminated CCs offer an extra protection for the last m message bits at the tail of the trellis. The rate loss can be avoided by either *puncutring* some code bits or by employing *tail biting* CCs. The puncturing requires careful consideration in order to avoid a catastrophic encoder. A catastrophic encoder produces an finite weight output for a infinite weight input. This can be avoided by using a *systematic* encoder with a feedback. A systematic encoder maps message bits directly to the code bits. For the CC considered in the example, the recursive systematic code (RSC) encoder is obtained as

$$\boldsymbol{G}(D) = \left(1, \frac{1 + D^2}{1 + D + D^2}\right)$$

which is only realizable for rational functions.

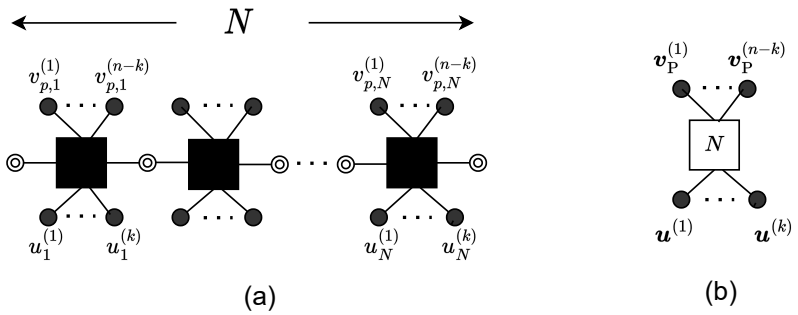


Figure 2.4: (a)Factor graph representation of Convolutional Codes (b) Compact graph representation of Convolutional codes

2.4.1 Graph Representation of Convolutional Codes

Consider a length-N trellis of a rate k/n systematic convolutional encoder i.e., each trellis section contain k number of information bits and $n-k$ number of parity bits. The factor graph of a convolutional code is shown in Figure 2.4(a), where each black shaded circle denote variable node corresponding to code bits, and black shaded squares represent the code constraints. Double circles represents the hidden state variable nodes. A black colored square with its connected state variable nodes corresponds to a code constraint of one trellis section.

A compact graph representation [9] of the factor graph of a systematic convolutional code is shown in Figure 2.4(b). In a compact graph, the black circle represents a length-N sequence of information or parity bits, $\boldsymbol{u}^i$ for $i = 1, \cdots, k$ and $\boldsymbol{v}^i$ for $i = 1, \cdots, n$, respectively, for the corresponding variable node type-i. The code constraint is represented by a black empty square, which is called a factor node. The factor node is labeled by the length N of the trellis. We will see in later chapters that the compact graph notation is useful in simplifying the construction, and asymptotic analysis of codes structures that are constructed by combining several convolutional codes.

2.5 Optimal Decoding Methods

Linear block codes and convolutional codes can be decoded in a blockwise or bitwise fashion by employing *maximum a posteriori* (MAP) decoders. Suppose a codeword sequence $\boldsymbol{v}$ of length N is perturbed by a noisy sequence $\boldsymbol{e}$, then the resulting received sequence $\boldsymbol{r}$ at the decoder is $\boldsymbol{r} = \boldsymbol{v} + \boldsymbol{e}$. A blockwise MAP decoder delivers at its output the estimated message sequence $\hat{\boldsymbol{u}}$ according to the rule

$$\hat{\boldsymbol{u}} = \arg\max_{\boldsymbol{u}} \; P(\boldsymbol{u}|\boldsymbol{r}) = \arg\max_{\boldsymbol{u}} p(\boldsymbol{r}|\boldsymbol{u})P(\boldsymbol{u}),$$

where $P(\boldsymbol{u})$ are the *a priori* probabilities of the message sequence. Blockwise MAP rule for CCs is implemented with the *Viterbi algorithm* [14]. This MAP rule minimizes the block error probability (BLER).

A bit-wise MAP decoder delivers at its output the hard decision of the

transmitted bit u_i according to the rule

$$\hat{u}_i = \arg\max_{u_i \in \{0,1\}} P(u_i|\boldsymbol{r}). \tag{2.13}$$

This hard-decision rule minimizes the bit-error probability (BER). $P(u_i|\boldsymbol{r})$ is the *a posteriori probability* (APP) of the information bit u_i conditioned on received noisy sequence $\boldsymbol{r}$. A decoder that delivers APP at its output for all input symbols $u_i \in \boldsymbol{u}$ is called an APP Decoder, and it is implemented with the rule

$$p(u_i = x|\boldsymbol{r}) = \sum_{\boldsymbol{u}:u_i=x} p(\boldsymbol{u}|\boldsymbol{r}) = \sum_{\boldsymbol{u}:u_i=x} \frac{p(\boldsymbol{r}|\boldsymbol{u})p(\boldsymbol{u})}{p(\boldsymbol{r})}, \qquad x \in \{0,1\}.$$

In practice, the APP rule is implemented in log-domain to avoid numerical instabilities. The log-APP decoder yields soft outputs $L(u_i)$ of the information bits u_i instead of the hard decision where,

$$L(u_i) = \log\frac{P(u_i=0)|\boldsymbol{r}}{P(u_i=1)|\boldsymbol{r}} = \log\frac{\sum_{\boldsymbol{u}:u_i=0} p(\boldsymbol{r}|\boldsymbol{u})P(\boldsymbol{u})}{\sum_{\boldsymbol{u}:u_i=1} p(\boldsymbol{r}|\boldsymbol{u})P(\boldsymbol{u})}.$$

The sign of $L(u_l)$ corresponds to the hard decision, and $|L(u_l)|$ corresponds to the reliability of the decision, where

$$L(u_l) = \log\left[\frac{P(u_l=+1|\boldsymbol{r})}{P(u_l=-1|\boldsymbol{r})}\right]. \tag{2.14}$$

$L(u_i)$ is known as the log-likelihood ratios of the bit u_i. The APP decoder is implemented by the BCJR algorithm [15]. Channel LLRs of received bit over an AWGN channel is computed by using

$$L_{ch}(r) = \frac{2}{\sigma^2}r.$$

An APP decoder can be converted into soft-input/soft-output (SISO) decoder with slight changes, which is discussed in the next section.

2.6 SISO Decoders with the BCJR Algorithm

SISO decoders play a fundamental role in the performance analysis of coding schemes studied in this thesis. In this section, we first review the basic principles of the trellis based APP decoder implementation via the BCJR algorithm. Next, we describe the modification in the BCJR algorithm based APP decoder resulting in a SISO decoder.

The BCJR algorithm is implemented using the code trellis. Consider a branch connecting a state node σ' at time $t-1$, denoted by $\sigma_{t-1} = \sigma'$, and a state node σ at time t, denoted by $\sigma_t = \sigma$, in a trellis section of the code trellis. The joint probability of a branch leaving the the state node at $\sigma_{t-1} = \sigma'$ to the node $\sigma_t = \sigma$, and a received noisy sequence $\boldsymbol{r}$ can be factorized as

$$\begin{aligned} p(\sigma', \sigma, \boldsymbol{r}) &= p(\sigma', \boldsymbol{r}_{<t}) \cdot p(\sigma', \boldsymbol{r}_t|\sigma') \cdot p(\boldsymbol{r}_{>t}|\sigma) \\ &:= \alpha_{t-1}(\sigma') \cdot \gamma_t(\sigma', \sigma) \cdot \beta_t(\sigma), \end{aligned}$$

The BCJR algorithm uses the channel observations of the code bits, and *a priori* probabilities of the information bits to compute the branch metric $\gamma_t(\sigma', \sigma)$ as

$$\gamma_t(\sigma', \sigma) = p(\boldsymbol{r}_t|(\sigma', \sigma)) \cdot P(\sigma|\sigma') = p(\boldsymbol{r}_t|\boldsymbol{v}_t) \cdot P(\boldsymbol{u}_t).$$

The algorithm next computes $\alpha_t(\sigma)$ recursively in the forward direction for $t = 1, \cdots, N-1$ via

$$\alpha_t(\sigma) = \sum_{\sigma'} \gamma_t(\sigma', \sigma) \cdot \alpha_{t-1}(\sigma'),$$

and $\beta_t(\sigma')$ recursively in the backward direction for $t = N, \cdots, 2$ via

$$\beta_{t-1}(\sigma') = \sum_{\sigma} \gamma_t(\sigma', \sigma) \cdot \beta_t(\sigma).$$

The boundary conditions used in the recursive computations assumes that the CC encoder starts and ends in zero state. The log-APP or soft output of the information bits is then obtained by using

$$L(u_t^{(i)}) = \log \frac{\sum_{(\sigma', \sigma): u_t^{(i)}=0} \alpha_{t-1}(\sigma') \cdot \gamma_t(\sigma', \sigma) \cdot \beta_t(\sigma)}{\sum_{(\sigma', \sigma): u_t^{(i)}=1} \alpha_{t-1}(\sigma') \cdot \gamma_t(\sigma', \sigma) \cdot \beta_t(\sigma)}.$$

A SISO decoder delivers the extrinsic information about a code bit $v_t^{(i)}$ as its output. In essence, the extrinsic information about a bit conveys the information that is available solely from the structure of the code itself. A SISO decoder is obtained by modifying the BCJR based APP decoder [16].

To get the extrinsic LLR of a code bit $v_t^{(i)}$, the observations, and the *a priori* probabilities corresponding to the $v_t^{(i)}$ are excluded during the branch metric computations of BCJR based APP implementation method. Denote by $\gamma_t^e(\sigma', \sigma)$ the modified branch metric, then the extrinsic LLR $L_e(v_t^{(i)})$ of the code bit $v_t^{(i)}$ is obtained by

$$L_e(v_t^{(i)}) = \log \frac{\sum_{(\sigma', \sigma): v_t^{(i)}=0} \alpha_{t-1}(\sigma') \cdot \gamma_t^e(\sigma', \sigma) \cdot \beta_t(\sigma)}{\sum_{(\sigma', \sigma): v_t^{(i)}=1} \alpha_{t-1}(\sigma') \cdot \gamma_t^e(\sigma', \sigma) \cdot \beta_t(\sigma)}.$$

SISO decoders are of interest as they are used as component decoders of coding schemes constructed by combining multiple component codes.

2.7 Transfer Functions of the SISO Decoders

A decoder transfer function maps the probabilities of the messages at the input and output of the decoder. The derivation of the transfer functions of SISO decoders rely on the Markov chain properties of the code, and the method of derivation is discussed extensively in [9, 17–19]. These transfer functions play a key role in the performance analysis of codes investigated in this thesis. We summarize the technique to derive the transfer functions of the SISO module of the component convolutional code on the BEC as discussed in [9].

Let p_l^{ext} be the extrinsic erasure probability of l-th code bit at the output of the SISO decoder. The p_l^{ext} depends on the channel and a-priori erasure probabilities of the code bits at the input of the SISO decoder, and denoted in terms of transfer function f_l as

$$p_l^{ext} = f_l(p_1, p_2, \cdots, p_n),$$

where the argument of f_l takes both channel and a-priori erasure probabilities. Derivation of f_l for the information and code bits is performed in two steps. In the first step, set of unique metric vectors that forward and backward recursions of the BCJR decoder can take are identified and their steady state distributions are determined. In the second step, the extrinsic erasure probabilities of code bits are computed by using the steady state distributions obtained in the first step.

The forward and backward metric vectors are defined in terms of 2^m number of trellis states $(\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_{2^m})$ in a trellis section at time t as

$$\boldsymbol{\alpha}_t = \left(\alpha_t(\boldsymbol{s}_1), \cdots, \alpha_t(\boldsymbol{s}_{2^m})\right),$$

and

$$\boldsymbol{\beta}_t = \left(\beta_t(\boldsymbol{s}_1), \cdots, \beta_t(\boldsymbol{s}_{2^m})\right),$$

respectively. For an all-zero transmitted codeword, the sets of vectors that $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ can take are denoted as

$$\mathcal{M}_\alpha = \{\boldsymbol{m}_\alpha^{(1)}, \cdots, \boldsymbol{m}_\alpha^{(|\mathcal{M}_\alpha|)}\},$$

and

$$\mathcal{M}_\beta = \{\boldsymbol{m}_\beta^{(1)}, \cdots, \boldsymbol{m}_\beta^{(|\mathcal{M}_\beta|)}\},$$

respectively. The cardinalities $|\mathcal{M}_\alpha|$ and $|\mathcal{M}_\beta|$ are finite. Sequences of forward vectors $(\cdots, \boldsymbol{\alpha}_{t-1}, \boldsymbol{\alpha}_t, \boldsymbol{\alpha}_{t+1}, \cdots)$, and backward vectors $(\cdots, \boldsymbol{\beta}_{t-1}, \boldsymbol{\beta}_t, \boldsymbol{\beta}_{t+1}, \cdots)$ form Markov chains. The probability of state transitions in Markov chains is described by the probability transition matrices $\boldsymbol{M}_\alpha$ and $\boldsymbol{M}_\beta$. The (i,j)-th element of these matrices corresponds to the probability of state transition from $\boldsymbol{m}_{\alpha(\beta)}^{(i)}$ to $\boldsymbol{m}_{\alpha(\beta)}^{(j)}$. From these state transition matrices, the steady state distribution vectors $\boldsymbol{\pi}_{\alpha(\beta)}$ of the Markov chain are obtained by solving

$$\boldsymbol{\pi}_\alpha = \boldsymbol{M}_\alpha \cdot \boldsymbol{\pi}_\alpha, \qquad \text{and} \qquad \boldsymbol{\pi}_\beta = \boldsymbol{M}_\beta \cdot \boldsymbol{\pi}_\beta.$$

Finally, the erasure probability of the extrinsic information of l-th code bit, which corresponds to $p(L_{e,t}^l = 1)$ is determined as

$$\begin{aligned} p_l^{\text{ext}} &= f_l(p_1, p_2, \cdots, p_n) = p(L_{e,t}^l = 1) \\ &= \sum_{i=1}^{|\mathcal{M}_\alpha|} \sum_{j=1}^{|\mathcal{M}_\beta|} p\left(L_{e,t}^l = 1 | \boldsymbol{\alpha}_t = \boldsymbol{m}_\alpha^{(i)}, \boldsymbol{\beta}_{t+1} = \boldsymbol{m}_\beta^{(j)}\right) \cdot p(\boldsymbol{\alpha}_t = \boldsymbol{m}_\alpha^{(i)}) \cdot \\ &\qquad\qquad p(\boldsymbol{\beta}_{t+1} = \boldsymbol{m}_\beta^{(j)}) \\ &= p(\boldsymbol{\alpha}_t = \boldsymbol{m}_\alpha^{(i)}) \cdot \boldsymbol{T}_l \cdot p(\boldsymbol{\beta}_{t+1} = \boldsymbol{m}_\beta^{(j)}) \end{aligned}$$

where (i,j)-th element of the matrix $\boldsymbol{T}_l$ is computed from

$$T_l(i,j) = p\left(L_{e,t}^l = 1 | \boldsymbol{\alpha}_t = \boldsymbol{m}_\alpha^{(i)}, \boldsymbol{\beta}_{t+1} = \boldsymbol{m}_\beta^{(j)}\right).$$

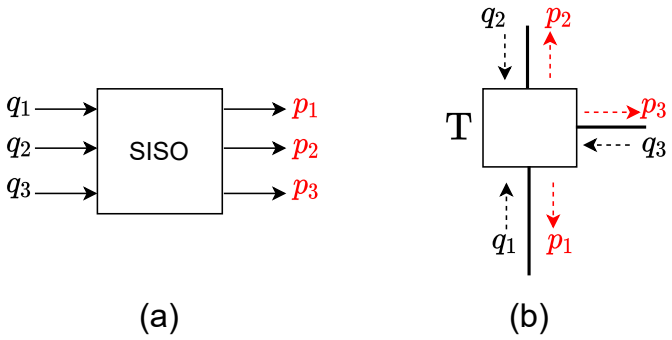


Figure 2.5: SISO decoder (a) input/output block diagram (b) Compact graph representation

Example 2.7.1. *Consider the rate-2/3 CC with generator matrix*

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 2/3 \\ 0 & 1 & 1/3 \end{bmatrix}$$

The input/output block diagram of SISO decoder for the trellis of this CC is shown in Figure 2.5(a), and its compact graph representation is shown in Figure 2.5(b). The code trellis is labeled as T in the compact graph. Incoming erasure probabilities to the SISO decoder are denoted as q_k for $k = 1, 2, 3$, and output erasure probabilities of the extrinsic messages are denoted as p_k. The transfer functions for the SISO decoder for each output are derived in [9]. For an averaged erasure probability q of incoming messages, the transfer functions of SISO decoder with inputs $q_1 = q_2 = q_3 \triangleq q$ are represented as

$$p_k = f_k(q_1, q_2, q_3),$$

where

$$f_1 = f_2 = \frac{q(q^5 - 4q^4 + 6q^3 - 5q^2 + 2q + 1)}{q^6 - 4q^5 + 6q^4 - 6q^3 + 5q^2 - 2q + 1},$$

and

$$f_3 = \frac{q^2(q^2 - 4q + 4)}{q^6 - 4q^5 + 6q^4 - 6q^3 + 5q^2 - 2q + 1}.$$

Chapter 3

Codes on Graphs and Iterative Decoders

Tanner presented the idea of "codes on graphs" in [7]. He showed how capacity-approaching codes—concatenated and LDPCs—could be viewed in a common graphical framework consisting of variable and constraint nodes. The framework facilitates an easy construction, and allow for generic decoding techniques of codes on graphs.

In this chapter, the construction of codes defined by graph structures, as well as the iterative message passing schemes for decoding them are discussed briefly. In next chapter, the techniques to estimate the iterative decoding performance of such codes under the asymptotic code block lengths are discussed. Improvement in the performance analysis techniques of such codes on graphs is the focus area of some of the contributions of this thesis.

3.1 Concatenated Codes

Concatenated codes [20] are combinations of several smaller component codes. Concatenated codes are useful when better error correction is required as these codes feature large block lengths. A simple concatenated code is constructed by combining two codes $\mathcal{C}_A(n_A, k_A)$ and $\mathcal{C}_B(n_B, k_B)$ as follows:

1. The data sequence is first placed in a $k_A \times k_B$ array.
2. Rows of the array are encoded by $\mathcal{C}_B$, and the resulting $n_B - k_B$ parity columns are appended to the array.
3. Lastly, the columns of the array are encoded by $\mathcal{C}_A$ to form an $n_A \times n_B$ array of final codewords of the resulting concatenated code.

The resulting concatenated code above is an example of product codes, where $\mathcal{C}_A$ and $\mathcal{C}_B$ are its *component codes*. An important characteristic of the class of concatenated codes is that these are usually defined by the *bi-partite* graph structures. The code bits and the component codes of the graph are represented by the varible and the constraint nodes respectively. Serially concatenated codes (SCC) [20] and parallel concatenated codes (PCC) are examples of concatenated codes that are adopted in standards for space and cellular communication.

3.1.1 Iterative Decoding of Concatenated Codes

For large message lengths, ML decoding of concatenated codes become computationally infeasible. Due to this reason, a sub-optimal iterative message passing decoding is used to decode the concatenated codes. In an iterative message passing decoding scheme, a collection of low-complexity decoders—at constraint nodes of the code graph—work in a distributed fashion on the received noisy sequence. After the first round of decoding, the local decoders exchange the output extrinsic messages with each other according to the code graph structure. The received extrinsic messages at the input of local constraint nodes are permuted. The permuted extrinsic sequences are used as the *a prior* sequences by the local decoders for the next round of decoding. The iterative decoding process continues in this fashion until some stopping criteria is reached.

An important message passing procedure that works iteratively is the *belief propagation*(BP) algorithm. SISO decoders are typically employed by the BP decoding algorithm at the local constraint nodes of the graph. Iterative decoding of concatenated codes under BP decoding is efficient than optimal ML decoding in terms of computational complexity. However, iterative BP decoding is sub-optimal due to the presence of *cycles* in the code graph. In general, 'the larger the cycle length or girth', the better the decoding performance before a certain number of decoding iterations are completed.

3.2 Low-Density Parity-Check Codes

R.G. Gallager invented Low-Density Parity-Check Codes (LDPC) [5] in his PhD thesis in 1960. Due to implementation difficulties, LDPC codes were neglected until they were revived by Mackay [6] in 1999. LDPC codes are block codes with a sparse parity check matrix $\boldsymbol{H}$. Sparsity implies that $\boldsymbol{H}$ contains a very low fraction of non-zero elements.

An $\boldsymbol{H}$ matrix with elements in a binary field defines a codeword constraint in matrix form. Each row of $\boldsymbol{H}$ is a single parity check (SPC) equation, and each column of $\boldsymbol{H}$ denotes a codeword bit. If the i-th codeword bit is connected to the j-th parity equation then the (j, i)-th entry of $\boldsymbol{H}$ is 1, otherwise it is zero. An LDPC code over a binary field with $N - K$ parity check equations and a codeword length of N is defined by means of a $N - K \times N$-sized matrix $\boldsymbol{H}$. A codeword vector $\boldsymbol{v} = \{v_1, \ldots, v_N\}$ is a valid codeword if it satisfies the syndrom condition $\boldsymbol{v}\boldsymbol{H}^T = 0$.

The matrix $\boldsymbol{H}$ is represented graphically by means of a Tanner graph [7]. This Tanner graph is analogous to the trellis of a convolutional code as it provides a complete description of the code and helps in defining the decoding algorithms. A Tanner graph is a *bipartite graph* with two types of nodes named *variable nodes* and *check nodes* (or *constraint nodes*), abbreviated as VNs and CNs respectively. VNs are associated to the codeword bits, and CNs are associated to the constraints. VNs and CNs are connected via edges according to the set of SPC equations in $\boldsymbol{H}$.

A (d_v, d_c) regular LDPC code corresponds to an LDPC code where all SPC equations in $\boldsymbol{H}$ include d_c number of VNs, and every VN is included in d_v number of SPC equations. A parity-check matrix of a regular-LDPC code thus contain exactly d_v number of ones in each column and d_c number of ones in

each row. The parameters d_v and d_c of a regular LDPC code are identified as variable node degree and check node degree respectively. The behavior of LDPC codes can be improved by allowing CNs and VNs of different degrees in the Tanner graph, and LDPC codes with such varying node degrees are known as irregular LDPCs.

For an efficient description of LDPC codes, a characterization of degree distribution of the Tanner graph from an edge perspective is defined in [21]. The fraction of edges that connect to variable(check) nodes of degree i is equal to $\lambda_i(\rho_i)$. This fraction is treated as the probability that an edge chosen uniformly at random from the graph is connected to a VN(CN) of degree i. The degree distribution polynomials $\lambda(x)$ and $\rho(x)$ are defined as [21]:

$$\lambda(x) = \sum_i \lambda_i x^{i-1},$$

and

$$\rho(x) = \sum_i \rho_i x^{i-1}.$$

As an example, degree distribution polynomials of a $(2, 3)$-regular code are $\lambda(x) = x$ and $\rho(x) = x^2$. An LDPC code ensemble LDPC$(n, \lambda(x), \rho(x))$, characterized by the code length n and the degree distribution polynomials, is defined by randomly connecting the edges between the VNs and CNs of the Tanner graph.

To obtain the design rate of LDPC$(n, \lambda(x), \rho(x))$, first the average variable and check node degrees , denoted as $\mathrm{l_{avg}}$ and $\mathrm{r_{avg}}$,respectively, are computed [21] from

$$\mathrm{l_{avg}} = \frac{1}{\int_0^1 \lambda(x)\mathrm{d}x},$$

and

$$\mathrm{r_{avg}} = \frac{1}{\int_0^1 \rho(x)\mathrm{d}x}.$$

The design rate $r(\lambda, \rho)$ of LDPC$(n, \lambda(x), \rho(x))$ with linearly independent rows in $\boldsymbol{H}$ is

$$r(\lambda, \rho) = 1 - \frac{\mathrm{l_{avg}}}{\mathrm{r_{avg}}} = 1 - \frac{\int_0^1 \rho(x)\mathrm{d}x}{\int_0^1 \lambda(x)\mathrm{d}x}.$$

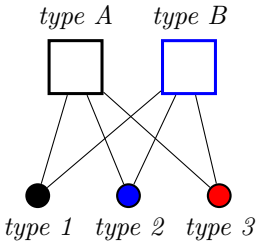
If $\boldsymbol{H}$ is not a full rank matrix then the actual rate of the code will be larger than the design rate. Iterative decoding of LDPC exploits the Tanner graph structure, and it is performed using the BP decoding algorithm.

3.2.1 Protograph LDPC Codes

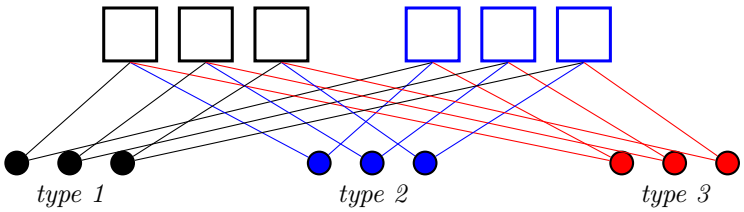
The randomly constructed LDPC codes are *unstructured*, which leads to high complexity in the decoding operations. To facilitate low complexity encoding and decoding, protograph LDPC codes, a class of *structured* LDPC codes, were introduced [22]. A protograph is a relatively small bi-partite graph from which a larger Tanner graph of an LDPC code is derived by following a *copy and permute* procedure. In the procedure, first the protograph is copied M times, and then the copies of *each edge in the protograph* are permuted among its M copies. The copy and permute operation is also known as *lifting* with a lift

factor M. The resulting Tanner graph after the lifting operation is called the *derived graph.*

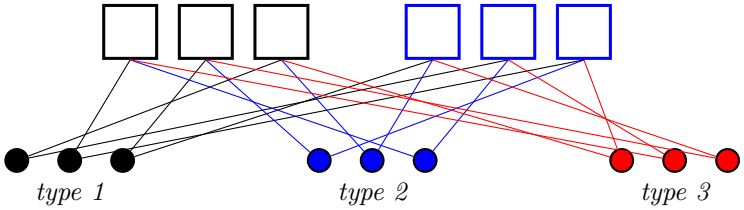
Example 3.2.1. *Consider the protograph as shown below. The three VNs in this protograph are denoted by "type 1,2,3" and the two check nodes by "type A,B". The protograph can be recognized as the Tanner graph of an LDPC code.*



Lifting above protograph with a lift factor $M = 3$ is obtained by first copying the protograph 3 times, which results in



and then permuting the copies of each edge in the protograph as



In example above, the connection of a $(2, 3)$ protograph is described in terms of the adjacency base matrix is

$$\boldsymbol{B} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

where "1" represent an edge connection between the corresponding VN and CN, and "0" goes in for the case when there is no connection between certain VN and CN of the protograph. The rows of $\boldsymbol{B}$ represent the CNs, and the columns represent the VNs. The parity check matrix $\boldsymbol{H}$ of the derived graph for the example is obtained by replacing each connection with a permutation matrix of size $M \times M$ as

$$\boldsymbol{H} = \left(\begin{array}{ccc:ccc:ccc} \cdot & 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot \\ \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & 1 \\ 1 & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot & \cdot \\ \hdashline \cdot & \cdot & 1 & \cdot & 1 & \cdot & \cdot & \cdot & 1 \\ 1 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & 1 & \cdot & \cdot & 1 & \cdot & \cdot \end{array}\right).$$

Zeros in the $\boldsymbol{H}$ matrix of the derived graph have been replaced with dots for the ease of visibility. Observe that the row and column weights of $\boldsymbol{H}$ are same as the base graph. The selection of permutation matrices is vital in having a derived graph with a large enough girth. In practice, the selection of optimized permutation matrices —with an objective of maximizing the girth—is typically performed through the *Progressive Edge Growth* [23] or *Approximated Cycle Extrinsic Message Degree* [24] algorithms.

3.3 Generalized LDPC Codes

Generalized LDPC (GLDPC) codes are the generalization of LDPC codes. In the Tanner graph of GLDPCs, the CNs can accommodate an arbitrary block code, and the VNs may represent more complex codes. For simplicity, we restrict to the generalization of CNs only and define the role of the VNs of GLDPCs in same way as the VNs of the Tanner graph of LDPC codes. The construction method of a GLDPC code, given a length N (d_v, d_c) regular LDPC code, and a Hamming code $\mathcal{C}_{Ham}(\mu, \kappa)$, is described in the following example.

Example 3.3.1. *Consider a length $N = 21$ regular $(2, 7)$ LDPC code, whose parity-check matrix H_L is*

$$H_L = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

We want to construct a GLDPC code with a $(7, 4)$ Hamming code as the component code of the $(2, 7)$ Tanner graph of H_L. The parity-check matrix H_{Ham} of the $(7, 4)$ Hamming code is

$$H_{Ham} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Observe that the d_c of the given H_L and the length of the Hamming code has to be the same.

To get a GLDPC code, the Hamming expansion of the first row of H_L is performed by replacing $d_c = 7$ 1s in it with the columns of H_{Ham} taken in any order. During the Hamming expansion, the zeros in the first row of H_L are replaced by an all-zero column vector of length $\lambda = \mu - \kappa = 3$. The procedure is repeated until the Hamming expansion of all the rows of H_L is performed. An example of one of the possible parity check matrix of the GLDPC code H_G with the Hamming code as the component code is:

$$\left(\begin{array}{cccccccccccccccccccc}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
\hdashline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
\hdashline
\vdots & & & & & & & & & \vdots & & & & & & & & & \vdots & \\
\end{array}\right).$$

The Hamming expansion of only the first two rows of H_L is shown above. The expansion of the remaining rows is done in a similar fashion.

The component code expansion of an irregular LDPC code can be performed in a similar fashion. However, the component code expansion of a row in a parity check matrix of an LDPC code can only be performed if the number of '1's in the row matches the length of the desired component code. The said restriction limits the degree of freedom in designing the GLDPCs , and therefore hinders the exploitation of the full potential of this class of codes. The reason is that this restriction leads to less desirable graph structures and code design rates. For instance, in Example 3.3.1 above, we start with a regular LDPC code with a design rate of $r^L = 1 - 2/7 = 5/7$. After the Hamming expansion, the resultant G-LDPC code becomes irregular with a design rate of [25]

$$r_G = 1 - \frac{d_v(\mu - \kappa)}{\mu} = \frac{1}{7}.$$

3.4 Turbo Codes

Turbo codes were introduced by Berrou, Glavieux, and Thitimajshima in 1993 [4]. Turbo codes consist of the parallel concatenation of two identical binary rate-1/2 recursive systematic convolutional (RSC) codes separated by a permutor.

A block diagram of a standard parallel concatenated convolutional code (PCCC) encoder is shown in Figure 3.1. A length-K message bit sequence $\boldsymbol{u}$ and its permuted version $\boldsymbol{u}'$, using a permutor Π, is encoded by the upper $\mathcal{C}^U$ and lower $\mathcal{C}^L$ convolutional encoders with generator polynomials $g_2(D)/g_1(D)$ each. The encoding results in two parity sequences $\boldsymbol{v}^U$ and $\boldsymbol{v}^L$ at the encoders outputs. The code bits are denoted by $\boldsymbol{v} = (\boldsymbol{u}, \boldsymbol{v}^U, \boldsymbol{u}^L)$.

With non-terminated encoders, the PCCC maps K message bits to $3K$ code bits, resulting in a code rate of 1/3. A family of rate-compatible PCCCs is obtained by puncturing the parity bit sequences $\boldsymbol{v}^U$ and $\boldsymbol{v}^L$. A family of turbo codes that are adopted in LTE standards has the generator polynomails $(g_2(D), g_1(D)) = (15, 13)$, and a set of optimized permutors and puncturers as specified in [26].

3.4.1 Compact Graph Representation of PCCC Codes

The block diagram of a PCCC encoder, Figure 3.1(a), is represented in terms of the factor graph in Figure 3.2(b). A factor graph of a PCCC is obtained

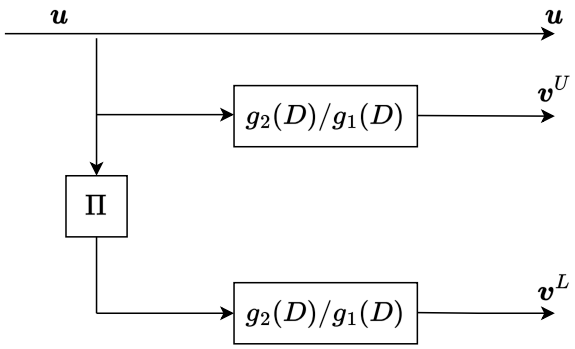


Figure 3.1: Encoder block diagram of PCCC.

by combining the factor graphs of its component convolutional codes through a permutor Π, which is shown in Figure 3.2(b).

A procedure for obtaining a compact graph from the factor graph of a convolutional code is discussed in Section 2.4.1. By using this procedure, a factor graph of a PCCC is described in terms of a compact graph as in Figure 3.2(c). The compact graph has three variable nodes which represent the sequences $\boldsymbol{u}$, $\boldsymbol{v}^U$, and $\boldsymbol{v}^L$, and two factor or constraint nodes which represent the trellises T^U and T^L. The trellises T^U and T^L correspond to the upper and lower encoders, $\mathcal{C}^U$ and $\mathcal{C}^L$, of PCCC respectively. The node $\boldsymbol{u}$ is connected to T^L via permutor $\boldsymbol{\Pi}$, which is shown as a slanted little line. In this manner, the compact graph notation can be obtained for codes obtained by concatenating convolutional codes.

3.4.2 Iterative Decoding of PCCC

The block diagram of the BP decoder of a PCCC is shown in Figure 3.3(a). The BP decoder of a PCCC consists of an upper SISO decoder, which corresponds to $\mathcal{C}^{\mathrm{U}}$, and a lower SISO decoder corresponding to $\mathcal{C}^{\mathrm{L}}$. At the start of the decoding,

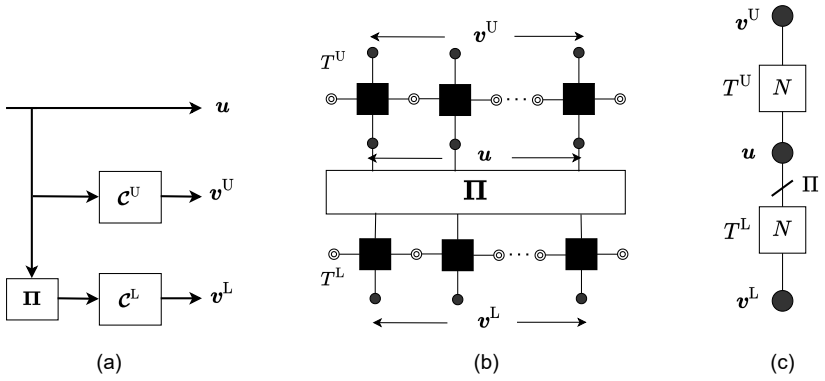


Figure 3.2: PCCC (a) Encoder Block diagram (b) Factor Graph representation (c) Compact Graph Representation

upper SISO decoder receives channel LLRs $L_{ch}(\boldsymbol{u})$, $L_{ch}(\boldsymbol{v}^U)$, corresponding to the sequences $\boldsymbol{u}$ and $\boldsymbol{v}^U$, and the *a prior* LLRs $L_a(\boldsymbol{u})$,corresponding to the sequence $\boldsymbol{u}$. After decoding, the upper decoder produces the extrinsic LLRs $L_e(\boldsymbol{u})$, which are permuted via permutor $\boldsymbol{\Pi}$. The permuted sequence $L_a(\boldsymbol{u}')$ becomes the *a prior* sequence for the lower SISO decoder.

The lower SISO decoder receives the channel LLRs $L_{ch}(\boldsymbol{u}')$, $L_{ch}(\boldsymbol{v}^L)$, the *a prior* LLRs $L_a(\boldsymbol{u}')$, and produces the sequence $L_e(\boldsymbol{u}')$ after the decoding. This completes one round of the iterative decoding. For the next iteration, the sequence $L_e(\boldsymbol{u}')$ through $\boldsymbol{\Pi}^{-1}$ to becomes the *a prior* sequence $L_a(\boldsymbol{u})$ for the upper decoder. The iterative decoding process continue in this manner until a certain number of iterations are performed. Note that *a prior* sequences are initialized to $\mathbf{0}$ before the start of the decoding.

Next, we translate the decoding process in the block diagram of the PCCC to the compact graph of the PCCC. Let us consider that the variable nodes of the compact graph of the PCCC receive the channel LLRs $L_{ch}(\boldsymbol{u})$, $L_{ch}(\boldsymbol{v}^U)$ and $L_{ch}(\boldsymbol{v}^L)$. In the first iteration, the SISO decoder at T^{U} receives the incoming messages $L_{ch}(\boldsymbol{u})+\mathbf{0}$ and $L_{ch}(\boldsymbol{v}^U)$ and produces the extrinsic LLRs $L_e(\boldsymbol{u})$ along its edge towards the variable node $\boldsymbol{u}$ as shown in Figure 3.3(b). Similarly, the decoding of T^{L} is performed, after which the SISO decoder at T^{L} produces the extrinsic LLRs $L_e(\boldsymbol{u}')$. At the end of desired number of iterations, the turbo decoder outputs the APP LLRs corresponding to the message bits $\boldsymbol{u}$ by

$$L_{\mathrm{APP}}(\boldsymbol{u}) = L_{ch}(\boldsymbol{u}) + L_a(\boldsymbol{u}) + L_e(\boldsymbol{u}).$$

The compact graph notation especially aids in simplifying the construction and decoding of the complex combinations of convolutional coding schemes. Furthermore, the compact graph unifies the concept of iterative message passing decoding of concatenated convolutional codes and LDPCs. The topic of developing a unified perspective for LDPCs and concatenated convolutional codes is one of the areas explored in this Ph.D. work.

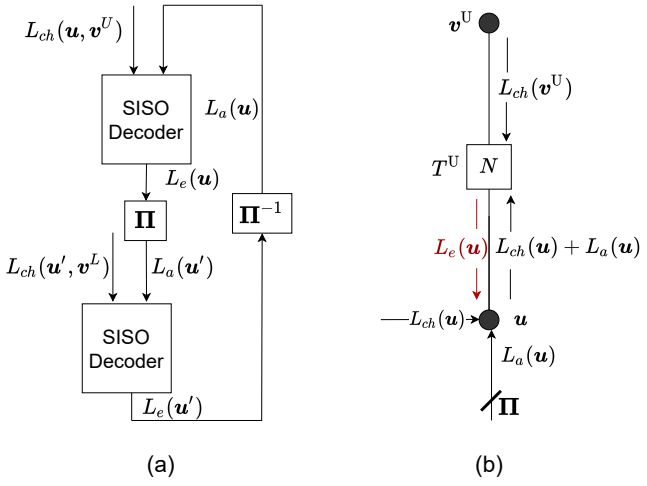


Figure 3.3: (a) BP Decoding Block diagram of a Turbo Code (b) BP Decoding of T^{U} in Compact Graph

Chapter 4

Iterative Decoding Thresholds

Iterative decoding schemes use decoding algorithms, which proceed in an iterative manner on a code graph. A sub-class of iterative decoding algorithms is a message-passing algorithm. In message-passing algorithms, nodes of the code graph exchange messages along its edges in an iterative fashion. The message-passing algorithms that exchange probabilities of messages along the edges of a code graph are known as belief propagation (BP) decoding algorithms.

BP decoding of long codes exhibit a threshold effect which partitions the channel space into two regions. In one region, the reliable communication is possible, whereas in the other region, the reliable communication is not possible. In literature, two major techniques are used to determine the thresholds of codes defined by the graph structures. These are known as the density evolution (DE) analysis and the extrinsic information transfer characteristics (EXIT) chart analysis techniques respectively. This chapter reviews these threshold analysis techniques, and their applications in determining the thresholds of LDPC codes and concatenated convolutional codes.

4.1 Threshold Analysis Techniques

In this section, a general overview of the threshold analysis techniques of iterative decoding is presented. The conditions under which these analysis techniques operate are also stated.

4.1.1 Density Evolution

The DE analysis technique is used to determine the iterative decoding performance of a code ensemble in the water-fall, low SNR, region. Mathematically, the DE analysis technique is expressed by a set of DE equations derived under the constraints of certain code ensemble properties. During the iterative BP decoding procedure, the DE analysis tracks the evolution of messages probabilities along the edges of the code graph via these DE equations. The DE analysis yields an ensemble average performance, as it is carried out over an entire code ensemble in the asymptotic code length.

The development of DE equations relies on following properties of code ensembles [21]:

1. During the iterative decoding process, the extrinsic LLRs at the output of the constituent decoders are symmetric for a symmetric channel. This property makes the decoders independent of the transmitted codeword, and simplifies the DE analysis by analyzing the performance of an all-zero transmitted codeword sequence.

2. Investigation of ensemble average performance is meaningful due to the concentration property [21, 27]. This property shows that the iterative decoding performance of a randomly chosen code from an ensemble is close to the ensemble average performance with a high probability.

3. An average performance a code ensemble, which uses the iterative BP decoding procedure, becomes optimal in the asymptotic limit of a code length. The sub-optimal performance of the iterative message passing decoders happens due to presence of cycles in the code graph. Selection of large enough code length during the DE results a cycle free graph for a certain number of decoding iterations. A cycle-free graph ensures that incoming messages to each node of the graph are statistically uncorrelated.

These properties are proven in [21] for turbo and LDPC codes. The stated properties of a code ensemble enables us to simplify the DE analysis by studying an all-zero transmitted codeword with a BPSK mapping. A channel introduces errors in the transmitted sequence according to some channel crossover probability. The outcome of the DE analysis is a threshold that partition the channel parameter space.

For the BEC, the channel parameter is the erasure probability (ε), for the BSC, it is the cross-over probability (δ), and for the AWGN channel it is the noise variance (σ^2) of the zero mean Gaussian probability density. For the BEC, the threshold is the maximum channel erasure probability ε, and for the AWGN channel, the threshold is the maximum σ^2, or alternatively the minimum E_b/N_0, for which the reliable communication is possible.

4.1.2 EXIT Chart Analysis

An *extrinsic information transfer* (EXIT) chart technique was pioneered by Ten Brink [8]. EXIT chart technique is a single-parameter approximation of the DE analysis technique, and it depicts the iterative BP decoding performance of the code—in the asymptotic code block length—on a chart. The EXIT technique tracks the evolution of mutual information of an averaged extrinsic message, w.r.t the transmitted sequence, in an unstructured graph. An EXIT chart shows the evolution of mutual information of each component decoder. The EXIT function of a component decoder can be modeled as

$$I_e = f(I_a),$$

where I_a and I_e are the mutual information of the *a prior* and extrinsic LLRs w.r.t. the transmitted sequence, and $f(\cdot)$ is the component decoder transfer function. The curve associated with the decoder transfer function on an EXIT chart is known as the *transfer characteristics* curve. Alternatively, entropy can be displayed instead of the mutual information on an EXIT chart.

EXIT analysis is an efficient technique to study the convergence behavior of a code ensemble. However, the EXIT technique may not suitable for estimating

the thresholds with high accuracy for *multi-edge* type structured code ensembles. This is where the DE analysis becomes advantageous over the EXIT chart technique, and it is argued in Paper I of this thesis.

4.1.3 MAP Thresholds

The MAP thresholds $\varepsilon^{\mathrm{MAP}}$ of a rate-R code can be obtained by applying the area theorem [21] on the converged erasure probability ($\bar{p}_{\mathrm{extr}}$) of an averaged extrinsic message during the iterative decoding. The area theorem yields MAP thresholds by relating the $\bar{p}_{\mathrm{extr}}$, and the code rate R as

$$\int_{\varepsilon^{\mathrm{MAP}}}^{1} \bar{p}_{\mathrm{extr}}(\varepsilon) d\varepsilon = R.$$

4.2 Decoding Thresholds of LDPCs

Let us briefly review the techniques to compute the iterative BP decoding thresholds of LDPC codes on the BEC. The threshold analysis method for the BEC can then be extended to the AWGN channel. The details of threshold computation of LDPC codes on the AWGN channel in this thesis are omitted, and can be found in [28].

4.2.1 Density Evolution of LDPCs on the BEC

Consider a (d_v, d_c) regular LDPC code ensemble. Denote by $e_{j,k}$ the edge connecting a VN j to the CN k, where $j = 1, \ldots, d_v$ and $k = 1, \ldots, d_c$. During the i-th iteration, let $p^{(i)}(e_{j,k})$ denote the erasure probability of the message from CN to VN along the edge $e_{j,k}$, and $q^{(i)}(e_{j,k})$ the erasure probability of the message from VN to CN along the edge $e_{j,k}$. At the start of the iterative DE procedure, the erasure probabilities of messages along the edges in the graph are initialized to $q^{(0)}(e_{j,k}) = \varepsilon$, and $p^{(0)}(e_{j,k}) = 0$. DE update at the k-th CN c_k is shown in Figure 4.1(a), and it is expressed as

$$\begin{aligned} p^{(i)}(e_{j,k}) &= 1 - \prod_{j' \setminus j} \left(1 - q^{(i-1)}(e_{j',k})\right), \\ &= 1 - \left(1 - q^{(i-1)}\right)^{d_c - 1}, \end{aligned} \tag{4.1}$$

where the second step follows from the symmetry of the regular code. The k-th CN update equation (4.1) states that the outgoing message on $e_{j,k}$ is erased with probability $p^{(i)}(e_{j,k})$, if an incoming message to the k-th CN along the edge $e_{j',k}$, where $j' \neq j$, is erased. Next, at the variable node the DE update is shown in Figure 4.1(b), and it is expressed as

$$q^{(i)}(e_{j,k}) = \varepsilon \cdot \prod_{j' \setminus j} p^{(i)}(e_{j',k}) = \varepsilon \cdot \left(p^{(i)}\right)^{d_v - 1}. \tag{4.2}$$

The VN update (4.2) states that the outgoing message from the j-th VN at the edge $e_{j,k}$ will only be erased, with probability $q^{(i)}(e_{j,k})$, if all incoming messages to the j-th VN through edges $e_{j,k'}$, where $k' \neq k$, are erased. For an

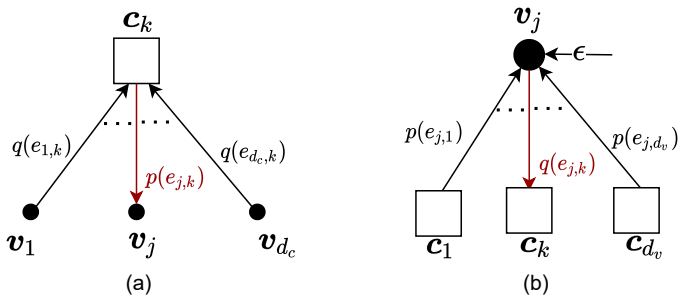


Figure 4.1: DE updates of (a) Check node (b) Variable node

unstructured or single-edge type(SE-type) regular LDPC code the check node outputs are uniformly distributed over all code symbols of the variable node. In this case $p^{(i)}(e_{j,k})$ and $q^{(i-1)}(e_{j,k})$ are equal along all edges of the graph in above DE equations.

By substituting (4.1) into (4.2), a one-dimensional DE recursion for a single-edge type regular LDPC code ensemble is obtained, which is

$$q^{(i)} = \varepsilon \cdot \left(1 - \left(1 - q^{(i-1)}\right)^{d_c-1}\right)^{d_v-1}. \tag{4.3}$$

For a SE-type LDPC code with an arbitrary regular or irregular degree distribution pair of (λ, ρ), the following DE recursion is used [21]

$$q^{(i)} = \varepsilon\lambda(1 - \rho(1 - q^{(i-1)})). \tag{4.4}$$

The threshold ε^{BP} is determined from the DE equations by observing the maximum ε for which the BP decoder correct all the erasures. For channel parameter $\varepsilon \leq \varepsilon^{BP}$, reliable communication is possible to achieve with a sufficiently large number of iterations, whereas for $\varepsilon > \varepsilon^{BP}$ reliable communication is not possible, no matter the number of iterations.

For an analysis purpose, a more formal expression of the threshold is introduced in [21], which uses the fixed point characterization of the threshold obtained from the DE recursion (4.4). A fixed point of a function f is a point in the function domain that is mapped to itself under the mapping f. Mathematically, the condition of fixed point x of f is expressed as $f(x) = x$. Let us denote the DE recursion (4.4) as a function

$$f(\varepsilon, x) = \varepsilon\lambda(1 - \rho(1 - x)). \tag{4.5}$$

If (4.5) has any fixed point x in $(0, \varepsilon)$ then the erasure probability of the extrinsic message emitted by the VNs does not converge to zero asymptotically with the number of iterations. On the other hand, if (4.5) has no fixed point in interval $(0, \varepsilon)$ then the erasure probability of the extrinsic message emitted by VNs converges to zero. Based on this, ε^{BP} is characterized as

$$\varepsilon^{\mathrm{BP}}(\lambda, \rho) = \sup\{\varepsilon \in [0, 1] : \text{No fixed point of } f(\varepsilon, x) \text{ for } x \in (0, \varepsilon)\}$$

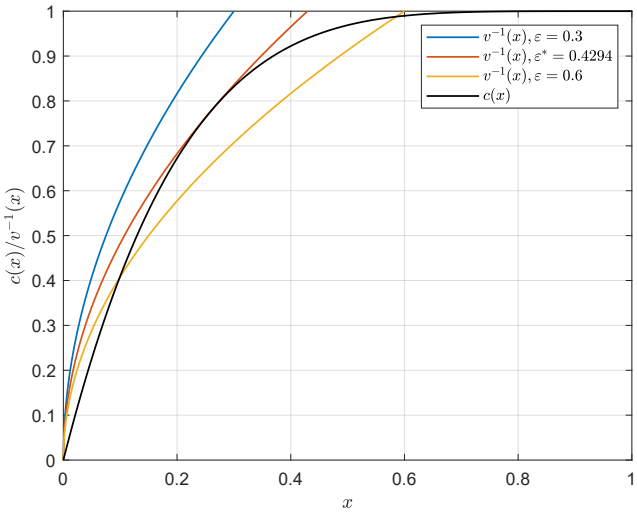


Figure 4.2: EXIT chart of regular $(3, 6)$ LDPC ensemble

4.2.2 EXIT Chart Analysis of LDPC codes

For unstructured LDPCs on the BEC, the EXIT chart analysis of LDPC codes becomes equivalent to the DE analysis. To obtain EXIT chart of an LDPC code ensemble with degree distribution pair (λ, ρ), it is helpful to represent (4.5) as the composition of two functions. These functions are: $v_\varepsilon(x)$ that shows the action of VNs, and $c(x)$ that shows the actions of CNs, defined as [21]:

$$v_\varepsilon(x) = \varepsilon\lambda(x),$$

and

$$c(x) = 1 - \rho(1 - x).$$

$c(x)$ corresponds to the transfer characteristics curve of the CNs and $v(x)$ to the VNs. The function $f(\varepsilon, x)$ in (4.5) is expressed in composition form as

$$f(\varepsilon, x) = v_\varepsilon(c(x)).$$

The BP decoder converges when $f(\varepsilon, x) < x$ [21]. This condition translates to

$$c(x) < v_\varepsilon^{-1}(x), \ x \in (0, 1).$$

This has a nice visual representation of convergence of an iterative decoder at $\varepsilon = \varepsilon^{\mathrm{BP}}$ in the EXIT chart, where $v_\varepsilon^{-1}(x)$ just touches the $c(x)$.

Example 4.2.1. *EXIT chart of a regular* $(3, 6)$ *LDPC ensemble over the BEC is shown in Figure 4.2. The transfer curve (TC) of a CN is represented by* $c(x)$*, and the TC of a VN, represented by* $v^{-1}(x)$*, are shown for the channel parameters* $\varepsilon = (0.3, 0.4294, 0.6)$*. In the displayed chart, the curve* $v^{-1}(x)$ *intersect with* $c(x)$ *at* $\varepsilon = 0.6$*, meaning the iterative decoder doesn't converge for this channel condition. At* $\varepsilon = 0.4294$*, the* $v^{-1}(x)$ *and* $c(x)$ *just touches each other, and therefore* $\varepsilon^* = 0.4294$ *defines the iterative decoding threshold of regular* $(3, 6)$ *LDPC ensemble. For channel parameters* $\varepsilon <= 0.4294$*, the iterative decoder converges.*

Both $c(x)$, and $v_\varepsilon(x)$ are shown to have interpretation in terms of entropy. The entropy of an averaged extrinsic message is equal to its erasure probability.

4.2.3 MAP threshold computations

Consider the received channel observations $\boldsymbol{Y}$ through a BEC(ε), parameterized by the erasure probability ε, for a transmitted codeword $\boldsymbol{X}$ that is selected with probability $p(x)$ from a length N code $\mathcal{C}$. The EXIT function associated with the j-th bit of $\mathcal{C}$ is

$$h_j(\varepsilon) = H(X_j|\boldsymbol{Y}_{\setminus j}),$$

If X_j is chosen with a uniform probability from $\{0,1\}$, then $H(X_j|\boldsymbol{Y}_j) = \varepsilon H(X_j|\boldsymbol{Y}_j =?) = \varepsilon$. The EXIT function characterizes how entropy is transferred from input to the output. The average EXIT function is defined as

$$h(\varepsilon) = \frac{1}{N}\sum_{j=1}^{N} h_j(\varepsilon).$$

The EXIT function is also characterized as the erasure probability of an extrinsic MAP decoder. Let $\hat{x}_j^{\mathrm{MAP}}(\boldsymbol{Y}_{\setminus j})$ denote the MAP decoder function of the j-th bit given the channel observation $\boldsymbol{Y}_{\setminus j}$, then the characterization

$$h_j(\varepsilon) = \Pr(\hat{x}_j^{\mathrm{MAP}}(\boldsymbol{Y}_{\setminus j}) =?)$$

is useful during the computations.

EXIT functions are connected to the code rate via the *Area Theorem*, which states that

$$H(\boldsymbol{X}|\boldsymbol{Y}(\delta)) = N\int_0^\delta h(\varepsilon)d\varepsilon, \tag{4.6}$$

where the channel observation $\boldsymbol{Y}(\delta)$ is parameterized with the channel parameter $\boldsymbol{\varepsilon}$. Assuming a uniform prior on the set of codewords, and integrating the above expression from 0 to 1,the expression in (4.6) yields the code rate.

EXIT functions connect the code performance under the BP decoder to its performance under the MAP decoder. That enables us to compute the MAP thresholds from the extrinsic erasure probability of the BP decoder. Let $\hat{x}_j^{\mathrm{BP},l}(\boldsymbol{Y}_{\setminus j})$ be the extrinsic estimate of j-th bit delivered by the BP decoder during the l-th iteration. Then the BP EXIT function is defined as

$$h_j^{\mathrm{BP}}(\varepsilon) = \Pr(\hat{x}_j^{\mathrm{BP},l}(\boldsymbol{Y}_{\setminus j}) =?).$$

Let us consider an LDPC code ensemble with degree distribution pair (λ,ρ). For the received channel observations $\boldsymbol{Y}$ at the BP decoder, the BP EXIT function in parameterized form is expressed as

$$h^{\mathrm{BP}}(\varepsilon) = \begin{cases} (\varepsilon, 0) & \varepsilon \in [0, \varepsilon^{\mathrm{BP}}) \\ (\varepsilon(x), L(1-\rho(1-x)) & \varepsilon \in (\varepsilon^{\mathrm{BP}}, 1] \leftrightarrow x \in (x^{\mathrm{BP}}, 1], \end{cases}$$

where

$$\varepsilon(x) = \frac{x}{\lambda(1-\rho(1-x))},$$

and

$$L(x) = \frac{\int_0^x \lambda(z)dz}{\int_0^1 \lambda(z)dz}. \tag{4.7}$$

The area below the curve of the BP EXIT function is called the trial entropy, and it is defined as

$$P(x) = \int_0^x L(1-\rho(1-z))\varepsilon'(z)dz.$$

Let x^{MAP} be the unique positive solution of $P(x) = 0$, and define $\varepsilon^{\mathrm{MAP}} = \varepsilon(x^{\mathrm{MAP}})$, then $\varepsilon^{\mathrm{MAP}}$ is the MAP threshold of the given ensemble. The EXIT function is defined as

$$h(\varepsilon) = \begin{cases} 0 & \varepsilon \in [0, \varepsilon^{\mathrm{MAP}}), \\ h^{\mathrm{BP}}(\varepsilon) & \varepsilon \in (\varepsilon^{\mathrm{MAP}}, 1]. \end{cases}$$

In this way, the design rate and the EXIT functions are related as

$$R(\lambda, \rho) = \int_{\varepsilon_{\mathrm{MAP}}}^1 \bar{p}_{\mathrm{extr}}(\varepsilon)d\varepsilon = \int_{\varepsilon_{\mathrm{MAP}}}^1 h^{\mathrm{BP}}(\varepsilon)d\varepsilon = \int_0^1 h(\varepsilon)d\varepsilon.$$

4.3 Decoding Thresholds of Turbo Codes

In this section, the techniques to compute the iterative decoding thresholds of turbo codes on both the BEC, and the AWGN channel are discussed.

4.3.1 DE Analysis of Turbo Codes on the BEC

A compact graph representation of a turbo code allows us to perform its DE analysis on the BEC in a similar manner as for the LDPC codes on the BEC. A CN update of a turbo code requires the transfer functions of the component SISO decoder. A method of deriving input/output transfer functions of the SISO decoder of a rate-k/n convolutional code on the BEC is described in [9, 19]. By using the transfer functions of the convolutional codes, the DE equations for the concatenated convolutional codes on the BEC were also derived in [9]. We summarize the derivation of the DE equations presented in [9] with an example of turbo code threshold computation for the BEC.

Consider the compact graph representation of turbo code in Figure 3.3(b). Let $p_{\mathrm{U,s}}^{(i)}$ and $p_{\mathrm{U,p}}^{(i)}$ denote the average erasure probability of extrinsic message from trellis T^U to VNs $\boldsymbol{u}$ and $\boldsymbol{v}^{\mathrm{U}}$, respectively. Similarly, $p_{\mathrm{L,s}}^{(i)}$ and $p_{\mathrm{L,p}}^{(i)}$ denote the average erasure probability of extrinsic message from trellis T^L to VNs $\boldsymbol{u}$ and $\boldsymbol{v}^{\mathrm{L}}$, respectively. These erasure probabilities are initialized to one before the start of iterative decoding procedure.

During the i-th iteration, the updates at VNs $\boldsymbol{u}$ and $\boldsymbol{v}^{\mathrm{U}}$ along the edges connected to T^{U} are performed by using

$$q_{\mathrm{L,s}}^{(i)} = \varepsilon \cdot p_{\mathrm{L,s}}^{(i-1)}$$
$$q_{\mathrm{U,p}}^{(i)} = \varepsilon.$$

Similarly, the updates at VNs $\boldsymbol{u}$ and $\boldsymbol{v}^{\mathrm{L}}$ along the edges connected to T^{L} are performed by using

$$q_{\mathrm{U,s}}^{(i)} = \varepsilon \cdot p_{\mathrm{U,s}}^{(i-1)}$$
$$q_{\mathrm{L,p}}^{(i)} = \varepsilon.$$

Next, update for T^{U} within the i-th iteration is performed with

$$p_{\mathrm{U,s}}^{(i)} = f_{\mathrm{U,s}}(q_{\mathrm{L,s}}^{(i)}, q_{\mathrm{U,p}}^{(i)})$$
$$p_{\mathrm{U,p}}^{(i)} = f_{\mathrm{U,p}}(q_{\mathrm{L,s}}^{(i)}, q_{\mathrm{U,p}}^{(i)}),$$

where $f_{\mathrm{U,s}}$ and $f_{\mathrm{U,p}}$ are the transfer functions of the SISO decoder at T^{U} for the systematic and parity bits, respectively. Similarly, the DE update for T^{L} is performed by using

$$p_{\mathrm{L,s}}^{(i)} = f_{\mathrm{L,s}}(q_{\mathrm{U,s}}^{(i)}, q_{\mathrm{L,p}}^{(i)})$$
$$p_{\mathrm{L,p}}^{(i)} = f_{\mathrm{L,p}}(q_{\mathrm{U,s}}^{(i)}, q_{\mathrm{L,p}}^{(i)}),$$

where $f_{\mathrm{L,s}}$ and $f_{\mathrm{L,p}}$ are the transfer functions of the SISO decoder at T^{L} for the systematic and parity bits, respectively.

4.3.2 DE Analysis of Turbo Codes on the AWGN Channel

Monte-Carlo (MC) method is used to perform the DE analysis of turbo codes on the AWGN channel. In the MC-DE analysis, the component SISO decoders are simulated, and probability densities of extrinsic LLRs message sequences are approximated. A histogram approximation method is often used to approximate the probability densities of the simulated LLRs.

We consider Figure 3.3(b) to describe the MC-DE analysis of the turbo code for the upper trellis, T^{U}, during an iteration of the iterative decoding. The MC-DE method comprises the three important steps described as follows:

1. *Variable node update:* Channel LLR sequence at the VN $\boldsymbol{u}$ and incoming LLR sequence to the VN $\boldsymbol{u}$ from the lower trellis T^{L} are combined and transmitted to the upper decoder T^{U}. These LLRs are sampled from their corresponding probability distributions before combining.

2. *Constraint node update:* The trellis at T^{U} receives LLRs sequences from VNs $\boldsymbol{u}$ and $\boldsymbol{v}^{\mathrm{U}}$. After decoding the received message, the SISO decoder at T^{U} produces extrinsic LLRs at its outputs.

3. *Density estimation and sampling:* The probability density of the extrinsic message $f(L_e(\boldsymbol{u}))$ from T^{U} is estimated. An *a prior* LLR sequence L_a is then sampled from $f(L_e(\boldsymbol{u}))$, and shared with the variable node $\boldsymbol{u}$. The sequence L_a is used in the update of lower CN T^{L}.

The processing of the lower trellis T^{L} is performed in a similar manner as T^{U}. The decoding process continues in this manner until the turbo-decoder converges.

4.3.3 EXIT Chart Analysis of Turbo-Codes on the AWGN Channel

The EXIT analysis of a turbo code is carried out in a similar manner as its MC-DE analysis on the AWGN channel. The main difference is that the EXIT analysis tracks the mutual information (MI) between the extrinsic messages and the transmitted symbols, whereas the MC-DE tracks the probability density of extrinsic messages during the iterative decoding. In addition to the transfer characteristics curves, the EXIT analysis can also be used to produce a *decoding trajectory* of an overall iterative decoder. A decoding trajectory traces a path between the transfer characteristics curves of the individual component decoders.

Let us review the method of computing the MI of a sequence of LLRs w.r.t the transmitted symbols. Consider a sequence of extrinsic LLRs L_e obtained by simulating the SISO decoder during the EXIT analysis. In EXIT analysis, the extrinsic messages (or *a prior* messages $L_a = L_e(\Pi)$) are assumed to be Gaussian with distribution $\mathcal{N}(\sigma_a^2/2, \sigma_a^2)$ [8]. Mean and variance of this Gaussian distribution are related due to the consistency property of LLRs. By using the variance σ_a^2, the mutual information is computed numerically by solving

$$\begin{aligned} I(L_a; X) &= J(\sigma_a) \\ &= 1 - \int \frac{1}{\sqrt{2\pi}\sigma_a} \exp - \frac{(l - \sigma_a^2/2)^2}{2\sigma_a^2} \log_2(1 + e^{-l}) dl. \end{aligned}$$

From the mutual information, σ_a is computing numerically from

$$\sigma_a = J^{-1}(I(L_a; X)).$$

The transfer characteristic curve of a component decoder is obtained by simulating the decoder for $I(L_a; X) \in [0, 1]$, and obtained with the following steps:

1. First σ_a is determined from the given $I(L_a; X)$.
2. A sequence of *a prior* L_a is sampled from distribution $\mathcal{N}(\sigma_a^2/2, \sigma_a^2)$.
3. The component decoder is simulated using the sampled L_a that produces an extrinsic message sequence L_e at its output.
4. MI $I(L_e; X)$ is determined by using the variance of the L_e.

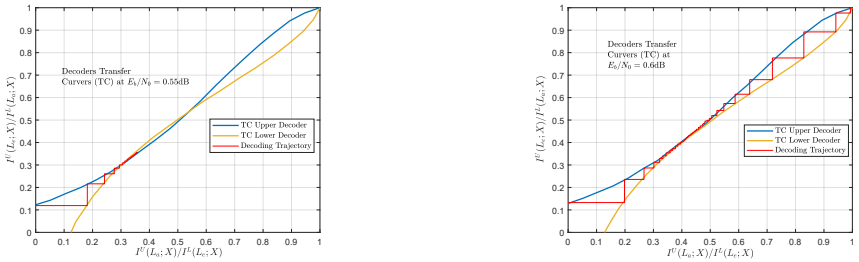
By viewing $I(L_e; X)$ as a function of $I(L_a; X)$, we define

$$I(L_e; X) = T(I(L_a; X)) \tag{4.8}$$

where $T(\cdot)$ represents the decoding function. When component decoder is connected to the channel, then $I(L_e; X)$ is viewed as a function of both $I(L_a; X)$ and E_b/N_0 both, and expressed as

$$I(L_e; X) = T(I(L_a; X), E_b/N_0). \tag{4.9}$$

In the context of turbo-codes, the transfer curves of both upper and lower decoders are obtained by using (4.9). If these transfer curves intersect on an



(a) $E_b/N_0 = 0.55$dB (b) $E_b/N_0 = 0.6$dB

Figure 4.3: Transfer Curves and decoding trajectories of memory 3 component convolutional codes of rate-1/2 Turbo codes on the AWGN Channel

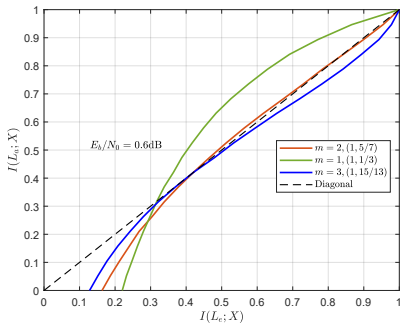


Figure 4.4: Transfer curves of memory $\{1, 2, 3\}$ component convolutional codes of rate-1/2 turbo codes on the AWGN Channel

EXIT chart, then the decoding trajectory of the iterative decoder get stuck at an intersection point. This intersection point also characterizes the fixed point of the decoder. For an iterative decoder to converge, the transfer curves of the component decoders must not intersect.

Example 4.3.1. *Consider a rate-1/2 turbo code consisting of two identical memory 3 component convolutional codes with generator polynomials* $(1, 15/13)$. *Rate-1/2 is achieved by puncturing the odd parity bits of sequences* $\boldsymbol{v}^U$ *and* $\boldsymbol{v}^L$. *The EXIT charts for the code on the AWGN channel are shown in Figure 4.3 for two different* E_b/N_0. *It can be seen that at* $E_b/N_0 = 0.55dB$, *the transfer curves of both component codes intersect, shown in Figure 4.3(a), which can be observed in the decoding trajectory as well. At* $E_b/N_0 = 0.6dB$, *the iterative decoder converges as the TCs of both component codes just touches each other as shown in Figure 4.3(b), which is visible in the decoding trajectory as well.*

Next, we investigate the effect of the component codes trellises strengths on the convergence of the iterative decoder by changing the trellises strengths in the rate-$1/2$ *turbo code. Transfer characteristics of the component code trellises of memory* $(1, 2, 3)$*—with generator polynomials* $(1, 1/3)$, $(1, 5/7)$, $(1, 15/13)$, *respectively—at* $E_b/N_0 = 0.6dB$ *are shown in Figure 4.4. Since identical component codes are used in the overall code, it suffice to analyze the performance*

of just one component code. In this case, the convergence behavior of the iterative decoder can be remarked by checking if the transfer characteristics of the component decoder intersect with the diagonal or not. From this, we can see that rate-1/2 turbo codes with memories $(1, 2)$ *component CCs don't converge, as their TCs intersect the diagonal. However, the transfer curve of the memory 3 CC barely touches the diagonal indicating the convergence of the iterative decoder, which is consistent with the observation in Figure 4.3(b).*

DE analysis may offer more advantages than the EXIT chart technique when it comes to punctured high rate codes, and multi-edge type codes. This is argued in Paper I of this thesis contribution.

Chapter 5

Weight Enumerator Analysis for Performance Bounds

Weight enumerator (WEM) analysis of a linear block code $\mathcal{C}(N, K)$ is the computation of its weight enumerator function (WEF) that can be used to upper bound the maximum likelihood (ML) decoding performance of $\mathcal{C}(N, K)$. A WEF enumerates the number of codewords in $\mathcal{C}(N, K)$ according to their Hamming weights w in a polynomial form $A(W) = \sum_{w=1}^{N} A_w W^w$, where A_w denotes the number of codewords in $\mathcal{C}(N, K)$ with weight w.

For a ML decoder, the performance of a code depends only on its codewords weight distribution. For the performance analysis on the AWGN channel, code linearity and the symmetry of channel makes the error probability calculation independent of the transmitted codeword $\boldsymbol{c}$. This allows us to simplify the performance analysis by studying only the all-zero transmitted codeword. Pairwise probability of a codeword error P_{cw} under the ML decoding is computed from the union bound as

$$\begin{aligned} P_{cw} &= \sum_{\boldsymbol{c}} \mathrm{P}(\text{error}|\boldsymbol{c})\mathrm{P}(c), \\ &= \mathrm{P}(\text{error}|\mathbf{0}), \\ &= \mathrm{P}(\cup_{\hat{\boldsymbol{c}}\neq\mathbf{0}} \text{ decision } \hat{\boldsymbol{c}}|\mathbf{0}), \\ &\leq \sum_{\hat{\boldsymbol{c}}\neq\mathbf{0}} \mathrm{P}(\text{decision } \hat{\boldsymbol{c}}|\mathbf{0}). \end{aligned} \tag{5.1}$$

Since error probability of each codeword with weight w is the same, (5.1) can be expressed as a function of weight-w by

$$P_{cw} \leq \sum_{w=d_{\min}}^{N} A_w P_w. \tag{5.2}$$

P_w denotes the probability that the ML decoder selects a codeword sequence of weight w instead of the all-zero codeword sequence. A_w can be easily obtained from the input-output WEF (IO-WEF) polynomial $A(I, W)$ of the encoder for

input Hamming weight i and output Hamming weight w as

$$A(I,W) = \sum_{w=1}^{N} \sum_{i=1}^{K} A_{i,w} I^i W^w. \tag{5.3}$$

For the AWGN channel, P_w is obtained from

$$P_w = Q\left(\sqrt{2wR\frac{E_b}{N_0}}\right) \tag{5.4}$$

The WEF of the $(7,4)$ Hamming code is provided in [1], which is

$$A(W) = 7W^3 + 7W^4 + W^7. \tag{5.5}$$

(5.5) shows that there are seven codewords with weight 3, another seven codewords with weight 4, and one codeword with weight 7. Furthermore, d_{min}—which is the lowest degree of the $A(W)$ polynomial—is 3. The IO-WEF of $(7,4)$ code is

$$A(I,W) = I(3W^3 + W^4) + I^2(3W^3 + 3W^4) + I^3(W^3 + 3W^4) + I^4W^7.$$

It can be observed that $A(W) = A(I,W)|_{I=1}$.

WEM analysis is useful in designing codes that employ sub-optimal iterative decoders. If a component code performs poorly in ML-decoding sense, then we can expect it to perform poorly in iterative decoding as well. In the following, we will briefly review the WEM analysis of convolutional, turbo and LDPC codes.

5.1 WEM Analysis of Convolutional Codes

Consider a rate-k/n convolution code that encodes a message block of length K and produces $N = K \cdot n/k$ code bits. k and n denote the number of information and parity bits per trellis section. Similar to block codes, the WEM analysis of convolutional codes requires computation of A_w. In the context of convolutional codes, A_w represents the path, with weight w, that diverges/reemerges from/to the all-zero paths in $L = K$ trellis sections. We refer to all such paths as the error events. IO-WEF can be determined via some algorithm that goes through L trellis sections, and updates $A_{i,w}$ table as it go through each possible path in a trellis section. For a systematic code, input-parity WEF (IP-WEF) is computed. The IP-WEF is defined as

$$A(I,P) = \sum_{i=1}^{K} \sum_{p=0}^{N-K} A_{i,p} I^i P^p, \tag{5.6}$$

for a weight-i encoder input bits and weight-p parity bits.

In this thesis, *transfer matrix* method discussed in [9,21] is used to compute the $A_{i,p}$ numerically. A *transfer matrix* $\boldsymbol{M}$ encodes the effect of state transition at each step. For a trellis with s number of states, $\boldsymbol{M}$ is of size $s \times s$, and each element of the matrix represents a monomial $I^i P^p$. The rows of $\boldsymbol{M}$ represent the present state in the trellis, and the columns represent the state after the

transition. For a transition from the i-th state to the j-th state in a length one trellis, the element $M_{i,j}$ of $\boldsymbol{M}$ encodes the associated weight of the input and output. For a length-K path, The weights of all possible length-K paths, start from the i-th state to the j-th state in the length-K trellis, are obtained from the $M_{i,j}$-th element of the matrix $\boldsymbol{M}^K$.

It is possible to determine the WEF analytically without using the computer algorithm. Analytical method uses the signal flow-graph technique [1] on the state diagram of the code and produces an entire distance spectrum. However, due to cumbersome calculations for codes with larger memories or larger block lengths, the signal flow-graph techniques is well suited for convolutional codes with small memories and block lengths.

5.2 Ensemble Enumerators for PCC

The ensemble of PCC is obtained by fixing its component codes and then using $K!$ different possible length K permutors. Both convolutional codes or block codes can be used as sub-codes for the analysis. To explain the derivation of ensemble enumerator we give an example of ensemble enumerator derivation for the PCC with the $(7, 4)$ Hamming code as its component codes.

5.2.1 PCC with Hamming code

For an averaged WEM analysis of concatenated codes, the *conditional parity-weight enumerator* (C-PWE), given the information weight, is useful and it is obtained from (5.6) as

$$A_i(P) = \sum_{p=0}^{N-K} A_{i,p} P^p. \tag{5.7}$$

C-PWE for the $(7, 4)$ Hamming code [1] for four input weights are

$$\begin{aligned} A_1(P) &= 3P^2 + P^3 \\ A_2(P) &= 3P + 3P^2 \\ A_3(P) &= 1 + 3P \\ A_4(P) &= P^3 \end{aligned}$$

Consider a weight one input message to both T^U and T^L of a PCC. For the four possible weight-1 inputs, either of the encoders produces three weight-2 parity outputs, and one weight-3 parity output, which can be checked from the $A_1(P)$ of the Hamming code. The parity bits at the of output of the T^U and T^L of a PCC are paired to form the parity bits of the PCC for a given input message and a $\boldsymbol{\Pi}$. For a short message length, it is possible to enumerate all the possible pair of parity bits along with their message bits. But this task becomes increasingly difficult with an increasing message length. Therefore, an ensemble *averaged WEM* analysis is recommended. In an averaged analysis, the T^L receives the permuted input message via permutor $\boldsymbol{\Pi}$, which is picked randomly with a uniform distribution from a set of all $K!$ possible interleavers. All of these interleavers permutes a weight-i input to $\binom{K}{i}$ possible permutations with the same probability. In this manner, the C-PWE $A_1^{PCC}(P)$ of PCC,

with the Hamming codes as its component codes, is computed by pairing given terms in $A_1^{T^U}$ and $A_1^{T^L}$ with probability $1/\binom{4}{1}$. For a PCC with an arbitrary component codes, the C-PWE $A_i^{PCC}(P)$ is obtained as

$$A_i^{PCC}(P) = \frac{A_i^{T^U} A_i^{T^L}}{\binom{K}{i}}. \tag{5.8}$$

Using (5.8), the averaged bit-wise and codeword enumerators, denoted as $A^{PCC}(I,P)$ and $A^{PCC}(W)$ respectively, are obtained from

$$A^{PCC}(I,P) = \sum_{i=1}^{K} I^i A_i^{PCC}(P),$$

and

$$A^{PCC}(W) = A^{PCC}(I,P)|_{I=P=W}.$$

5.2.2 Simplified computation of $A^{PCC}(I,P)$

Computation of polynomial $A^{PCC}(I,P)$ can be further simplified. To do this, (5.8) and (5.6) are used in $A^{PCC}(I,P)$, which results in

$$A^{PCC}(I,P) = \frac{\sum_{i=1}^{K} I^i \left(\sum_{p_1=0}^{N-K} A_{i,p_1}^{T^U} P^{p_1} \sum_{p_2=0}^{N-K} A_{i,p_2}^{T^L} P^{p_2}\right)}{\binom{K}{i}},$$

which can be expressed in the form

$$A^{PCC}(I,P) = \sum_{i=1}^{K} \sum_{p=0}^{2(N-K)} \bar{A}_{i,p}^{PCC} I^i P^p.$$

Let $p = p_1 + p_2$, where $p = 0, 1, \cdots, 2(N-K)$ denotes the weight of concatenated parity bits of PCC. The coefficients $\bar{A}_{i,p}^{PCC}$ of $A^{PCC}(I,P)$ are then obtained by using the scaled convolution operation

$$\bar{A}_{i,p}^{PCC} = \frac{1}{\binom{K}{i}} \sum_{p_1=0}^{2(N-K)} A_{i,p_1}^{T^U} A_{i,p-p_1}^{T^L}. \tag{5.9}$$

5.3 Ensemble Enumerators for Finite Length LDPC Codes

For an ensemble of random linear codes of length N and dimension K, the averaged output WEF is expressed as

$$A(W) = \sum_{w=1}^{N} \sum_{i=1}^{K} \bar{A}_w W^w,$$

where

$$\bar{A}_w = \binom{N}{w} 2^{-N(1-R)}.$$

For a regular (d_v, d_c) Gallager ensemble of LDPC codes [5], a parity check matrix H consists of d_v sub-matrices H_i, $i = 1, \cdots, d_v$. The first sub-matrix of Gallager ensemble is structured, whereas the remaining sub-matrices are obtained by randomly permuting the columns of the structured sub-matrix. The WEF of a SPC code is defined as

$$A^{\mathrm{SPC}}(W) = \sum_{w \text{ even}} \binom{d_c}{w} W^w,$$

which is used in deriving $A^{\mathrm{G}}(W)$ of the WEF of H_i in the Gallager ensemble as

$$A^{\mathrm{G}}(W) = \left(A^{\mathrm{SPC}}(W)\right)^M = \sum_{w=0}^{N} A_w^G W^w, \tag{5.10}$$

where A_w^G represents the number of weight-w binary sequences $\boldsymbol{x}$ that satisfies the syndrom condition $\boldsymbol{x}H_i^T = 0$. Note that (5.10) is same for all sub-matrices. The probability $p(w)$ that a weight-w random codeword sequence of length N satisfies the syndrom condition for a given H_i is $A_w^G / \binom{N}{w}$. By observing the structure of Gallager ensemble, the coefficients $\bar{A}_w^G$ of averaged WEF of a Gallager's ensamble is obtained by

$$\bar{A}_w^G = \binom{N}{w} (p(w))^{d_v} = \frac{(A_w^G)^{d_v}}{\binom{N}{w}^{d_v - 1}}. \tag{5.11}$$

Chapter 6

Turbo-like Codes

The class of turbo codes in a broader picture can be considered as any concatenated arrangements of the component CCs. By following this, the PCCCs, SCCCs and the hybrid of PCCCs and SCCCs etc can be considered as examples of codes belonging to the class of turbo codes. Loosely speaking, there are two important characteristics that separate the class of turbo codes from the rest of the classes of codes: The first one is the usage of random/pseudo random permutors in the code construction, and the second one is the implementation of iterative decoding by following the turbo decoding principle. This principle follows the decoding method described for the iterative decoding of turbo codes. The turbo decoding principle facilitates low complexity decoding by engaging the local component decoders of the code graph independently, and then exchanging their probabilistic beliefs of the code symbols in an iterative fashion.

Turbo-like codes (TCs) are the generalization of the class of turbo codes in the sense that the component codes can take an arbitrary block code while retaining the key characteristics of their parent class. In this chapter, we briefly review the design of some examples of TCs that were considered by Moludi *et al.* in [9]. We describe these TCs by means of their compact graph structures. For details on the conventional encoder/decoder block diagrams of the examples TCs and their connection with compact graphs, we refer the interested reader to [9, 10].

6.1 Serially Concatenated Codes

A serial concatenation of two rate-1/2 recursive systematic convolutional (RSC) codes, referred to as the outer and inner codes, results in SCCCs [29]. Its compact graph representation is shown in Figure 6.1(a), where the outer and inner codes are respresented by the CNs T^{O} and T^{I} respectively. The small rectangle that connects T^{O} and T^{I} in the graph represents a multiplexer/demultiplexer. A length-N information sequence $\boldsymbol{u}$ is first encoded by the outer encoder that produces the parity sequence $\boldsymbol{v}^{\mathrm{O}}$. The combined sequence $(\boldsymbol{u}, \boldsymbol{v}^{\mathrm{O}})$ from the multiplexer is reordered via the permutor $\boldsymbol{\Pi}$, which results in an intermediate sequence $\tilde{\boldsymbol{v}}^{\mathrm{O}}$ of length $2N$. At the last stage of encoding, the inner encoder encodes $\tilde{\boldsymbol{v}}^{\mathrm{O}}$ and produces the length-$2N$ parity sequence $\boldsymbol{v}^{\mathrm{I}}$. The transmitted length-$4N$ codeword sequence is $\boldsymbol{v} = (\boldsymbol{u}, \boldsymbol{v}^{\mathrm{O}}, \boldsymbol{v}^{\mathrm{I}})$. The code rate of this concate-

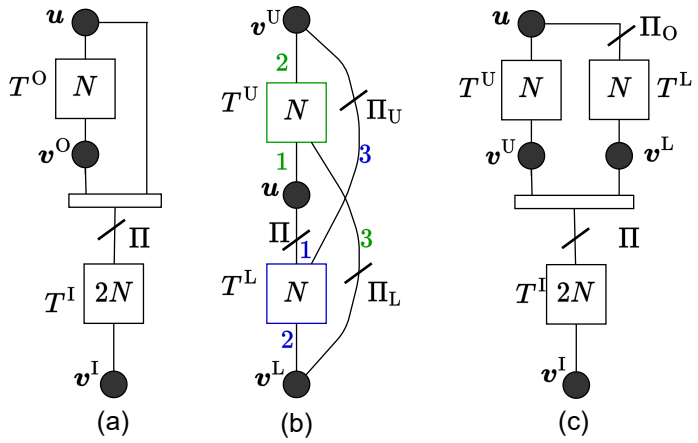


Figure 6.1: Compact graph representation of (a) SCCs (b) BCCs (c) HCCs

nated scheme is 1/4.

6.2 Braided Convolutional Codes

Similar to PCCCs and SCCCs, the braided convolutioanl codes (BCCs) employ two identical RSC codes of rate-2/3 as their component codes. The major difference is that the parity sequence of one component encoder is used as the input of the other component encoder. By following this construction, a tail-biting version of the BCCs is obtained. The compact graph representation of BCCs is shown in Figure 6.1(b). Two component codes are represented by the upper and lower CNs denoted as T^{U} and T^{L}, respectively. The length-N parity sequences of the upper and lower encoders are represented by the $\boldsymbol{v}^{\mathrm{U}}$ and $\boldsymbol{v}^{\mathrm{L}}$ respectively. The upper encoder receives a length N information sequence $\boldsymbol{u}$ and reordered copy of parity sequence $\boldsymbol{v}^{\mathrm{L}}$ via permutor $\boldsymbol{\Pi}_{\mathrm{U}}$ and produces a parity sequence $\boldsymbol{v}^{\mathrm{U}}$. Similarly, the lower encoder produces a parity sequence $\boldsymbol{v}^{\mathrm{L}}$ after receiving a reordered copy of information sequence $\boldsymbol{u}$ via the permutor $\boldsymbol{\Pi}$, and a reordered copy of parity sequence $\boldsymbol{v}^{\mathrm{U}}$ via permutor $\boldsymbol{\Pi}_{\mathrm{L}}$. Similar to the PCCCs, the transmitted codeword is $\boldsymbol{v} = (\boldsymbol{u}, \boldsymbol{v}^{\mathrm{U}}, \boldsymbol{v}^{\mathrm{L}})$, and the overall code rate is 1/3. Note that the tail-biting version of the BCCs may not be well suited for practical implementations due to the parity sequences feedback within the same compact graph, but the code is well defined, and DE analysis can be easily performed.

6.3 Hybrid Concatenated Codes

A *Hybrid Concatenated Convolutional Code* (HCCC) is obtained by serially concatenating a PCCC with an RSC code. The PCCC in this structure represents the outer code, and the RSC code represents the inner code. The compact graph representation of the HCCC is shown in Figure 6.1(c). The component codes of the PCCCs are two identical rate-1/2 RSC codes, and the inner code is a rate-1/2 RSC code which needs not necessarily be the same as the component

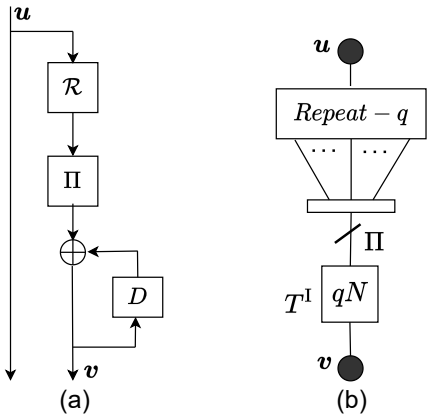


Figure 6.2: Repeat-accumulate codes (a) Encoder block diagram (b) Compact graph representation

codes of the PCCC. The component codes trellises of the PCCC are denoted by T^{U} and T^{L}, and of the inner code trellis is represented by T^{L} in the compact graph. The information sequence $\boldsymbol{u}$ after encoding through the PCCC results in the parity sequences $\boldsymbol{v}^{\mathrm{U}}$ and $\boldsymbol{v}^{\mathrm{L}}$. These parity sequences from the PCCC are multiplexed and reordered. The resulting sequence is then encoded by the inner encoder which results in a parity sequence $\boldsymbol{v}^{\mathrm{I}}$. The transmitted length-$5N$ codeword sequence is $\boldsymbol{v} = (\boldsymbol{u}, \boldsymbol{v}^{\mathrm{U}}, \boldsymbol{v}^{\mathrm{L}}, \boldsymbol{v}^{\mathrm{I}})$, and the code rate of this concatenated scheme is $1/5$.

6.4 Repeat-Accmulate Codes

Repeat-accumulate (RA) codes, introduced by Divsalar *et al.* [30], are perhaps the simplest among the class of TCs. An interesting aspect of RA codes is that they can be viewed as a sub-class of either LDPCs or TCs. The RA based LDPC codes, therefore, have an advantage of easy encoding. Furthermore, these can be seen as a step towards bridging the gap in unifying the classes of LDPCs and TCs.

The RA code is a serial concatenation of a rate-$1/q$ repetition code $\mathcal{R}$ with an accumulator of transfer function $1/(1+D)$ through a permutor Π as shown in the block diagram in Figure 6.2(a). The compact graph representation of RA code is shown in Figure 6.2(b), where the outer code represents the repetition code and the inner code represents the accumulator that is implemented by a memory one trellis. The accumulator is implemented through a simple memory one CC with generator polynomial $(1, 1/(1+D)$. In the compact graph, a length-N message sequence $\boldsymbol{u}$ is encoded by an outer repetition code encoder that generates q-copies of each symbol in the message sequence $\boldsymbol{u}$. The q-copies are next combined through a multiplexer, which results in a combined codeword sequence of length-qN. The combined sequence is reordered via the permutor Π, and encoded by the accumulator T^{I}, which results in a length-qN parity sequence $\boldsymbol{v}$. The transmitted codeword sequence is $(\boldsymbol{u}, \boldsymbol{v})$, and the overall code rate is $1/(q+1)$.

6.5 Thresholds of TCs

Exact DE equations for TCs over the BEC were derived in [9] and used to compute their BP thresholds $\varepsilon_{\mathrm{BP}}$. The DE equation of PCCCs are summarized in Section 4.3.1. The General principle of the derivation of DE equations for SCCCs, BCCs, and HCCCs is same as for the PCCCs. The derivations of the DE equations for the examples of TCs are provided in [9,10]. In this section, we discuss the threshold computation of BCCs on the BEC with channel erasure probability ε.

6.5.1 Thresholds of BCCs on the BEC

Consider the compact graph of tail-biting BCC in Figure 6.1(b). Let $p_{\mathrm{U},k}^{(i)}$ and $p_{\mathrm{L},k}^{(i)}$ denote the erasure probabilities of extrinsic messages from CNs T^{U} and T^{L} along their k-th connected edges, where $k = 1, 2, 3$ color coded edge labels are shown in Figure 6.1(b). Each CN in the BCC compact graph implements an identical rate-2/3 RSC CC with a generator polynomial of

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 1/7 \\ 0 & 1 & 5/7 \end{bmatrix}$$

We first derive the DE equations of CN T^{U}. The node T^{U} receives following erasure probabilities (VN updates) along its edges during the i-th iteration

$$\begin{aligned} q_{\mathrm{L},1}^{(i)} &= \varepsilon \cdot p_{\mathrm{L},1}^{(i-1)}, \\ q_{\mathrm{L},2}^{(i)} &= \varepsilon \cdot p_{\mathrm{L},3}^{(i-1)}, \\ q_{\mathrm{L},3}^{(i)} &= \varepsilon \cdot p_{\mathrm{L},2}^{(i-1)}. \end{aligned} \tag{6.1}$$

The exact DE equation as a function of the received erasure probabilities is expressed as

$$p_{\mathrm{U},k}^{(i)} = f_{\mathrm{U},k}\left(q_{\mathrm{L},1}^{(i)}, q_{\mathrm{L},2}^{(i)}, q_{\mathrm{L},3}^{(i)}\right) \text{ for } k = 1, 2, 3, \tag{6.2}$$

where $f_{\mathrm{U},k}$ denotes the transfer function of the k-th edge of SISO decoder of rate-2/3 trellis at T^{U}.

The DE equations of T^{L} can be obtained by swapping the indices U and L in (6.1) and (6.2). Both CNs T^{U} and T^{L} are updated in parallel during the same iteration. After the first iteration, the updated erasure probabilities $p_{\mathrm{U},1}^{(i)}$ and $p_{\mathrm{L},1}^{(i)}$ towards the information node are received, and *a-posteriori* erasure probability of $\boldsymbol{u}$ during the i-th iteration is obtained as

$$p_a^{(i)} = \varepsilon \cdot p_{\mathrm{U},1}^{(i)} \cdot p_{\mathrm{L},1}^{(i)}.$$

6.6 Performance and Minimum Distance Bounds of TCs

The BER performance of the TCs with component convolutional codes under the ML decoding is upper bounded in [12] by

$$P_b = \sum_{i=1}^{N} \sum_{p=1}^{N(1/R-1)} \frac{i}{N} \bar{A}_{(i,p)} Q\left(\sqrt{2(i+p)R\frac{E_b}{N_0}}\right),$$

where the averaged WEM $\bar{A}_{(i,p)}$ of the corresponding TCs in terms of input i and parity weights p are given by

$$\bar{A}^{\mathrm{SCC}}_{(i,p)} = \sum_{p1} \frac{A^{\mathrm{T^O}}_{(i,p_1)} \cdot A^{\mathrm{T^I}}_{(i,p-p_1)}}{\binom{2N}{i+p_1}},$$

$$\bar{A}^{\mathrm{BCC}}_{(i,p)} = \sum_{p1} \frac{A^{\mathrm{T^U}}_{(i,p_1,p-p_1)} \cdot A^{\mathrm{T^L}}_{(i,p-p_1,p_1)}}{\binom{N}{i}\binom{N}{p_1}\binom{N}{p-p_1}},$$

$$\bar{A}^{\mathrm{HCC}}_{(i,p)} = \sum_{p1}\sum_{p2} \frac{A^{\mathrm{T^U}}_{(i,p_1)} \cdot A^{\mathrm{T^L}}_{(i,p_2)} \cdot A^{\mathrm{T^L}}_{(i,p_1+p_2,p-p_1-p_2)}}{\binom{N}{i}\binom{2N}{p_1+p_2}}.$$

Since TCs can be seen as a class of protograph based GLDPCs, their IP-WEF can be equivalently computed by the method described in [31, 32].

Bounds on the minimum distance of ensembles of the TCs were also given in [12]. The derivation of the minimum distance bound in [12] assumes an equiprobable code selection among Ω number of codes in an ensemble. The number of codewords with weight w in an ensemble is computed as $\Omega\bar{A}_w$, where $\bar{A}_w$ is an averaged WEM of the TC ensemble. From this, the total number of codewords in an ensemble with weight $w < \tilde{d}$ for some integer $\tilde{d}$ is computed as

$$\Omega_{w<\tilde{d}} = \Omega \sum_{w=1}^{\tilde{d}-1} \bar{A}_w.$$

An upper bound on the number of codes with minimum distance $d_{min} \geq \tilde{d}$ in an ensemble is computed by

$$\Omega_{w\geq\tilde{d}} < \Omega - \Omega \sum_{w=1}^{\tilde{d}-1} \bar{A}_w.$$

Let α denote the fraction of codes with $d_{min} \geq \tilde{d}$, then this fraction is upper bounded by

$$\alpha < 1 - \sum_{w=1}^{\tilde{d}-1} \bar{A}_w.$$

For a given α and $\bar{A}_w$, a bound on the minimum distance of an ensemble is obtained by solving for largest $\tilde{d}$ that satisfies the above equation. For a smaller α, the resultant $\tilde{d}$ may result in a larger number. This corresponds to the idea of *expurgation*, in which some codes with poor minimum distance properties in an ensemble are eliminated to increase the minimum distance of the resulting expurgated ensemble.

The ideas behind the bounds on performance and minimum distances of TCs are extended to one of the contributions in Paper IV, where we generalize the regular BCC graph structure to the class of GLDPC codes with connvolutional code constraints, and investigate its minimum distance growth with the increasing block length.

Chapter 7

Spatially Coupled Codes

The spatial coupling technique [33] interconnects a sequence of codes on graphs. Spatially coupled ensembles has attracted a lot of attention due to their excellent performance under the iterative decoding. The concept of spatial coupling goes back to the idea of *LDPC convolutional codes* (LDPC-CC) by Felström and Zigangirov in 1999 [34]. In terms of spatial coupling, the LDPC-CCs can be seen as a collection of standard LDPC code ensembles, which are connected to each other along the spatial dimension. Lentmaier *et al.* in 2005 reported a significantly improved performance of terminated LDPC-CCs [35] when compared to their underlying LDPC codes. The iterative BP decoding performance of regular LDPC-CCs was observed to perform close to its MAP decoding performance in 2005 by Lentmaier *et al.* in [36]. Kudekar and Urbanke later showed that this observation is due to a vary general phenomenon called the *threshold saturation via spatial coupling*, which they proved in [11].

When SC-LDPC codes were gaining attention of the research community, it was identified that the braided codes—introduced by Truhachev, Lentmaier and Zigangirov in 2002 [37]—were also part of the class of spatially-coupled codes. Unlike the tail-biting BCCs presented in the previous chapter, the originally proposed braided codes are inherently spatially coupled. Braided codes are classified into two major categories depending on their component codes. The first one is *braided block codes* (BBCs) with block component codes, and second one is *braided convolutional codes* (BCCs) with convolutional component codes. BBCs with Bose-Chaudhuri-Hocqenghem (BCH) component codes are used in high speed optical communications due to their excellent performance under the iterative hard decision decoding [38, 39].

In [40], BCCs were characterized by a regular compact graph structure, and their performance on the BEC was investigated by means of the DE equations. Furthermore, the existence of threshold saturation of BCCs on the BEC was proven analytically by Moloudi, Lentmaier, and Amat in [9]. The concept of spatial coupling of regular BCCs was generalized to irregular TCs as well in the same work. However, the analysis of coupled and uncoupled versions of the TCs on the AWGN channel remained, which is one of the focus area of this thesis. In this chapter, a brief summary of the construction and analysis techniques of related spatially coupled ensembles is presented.

7.1 SC-LDPC Codes

We explain the construction of SC-LDPC codes with an example. Consider a protograph of a regular $(3, 6)$ LDPC code. The SC-LDPC code ensemble with an underlying $(3, 6)$ protograph LDPC code is constructed as follows:

1. We first place L copies of $(3, 6)$ protograph in a sequence of a coupled chain, where L denotes the coupling length, see Figure 7.1(a). Each individual protograph in the sequence is time indexed as $t = 1, \cdots, L$.

2. A SC-LDPC code ensemble of coupling memory m is obtained by interconnecting the edges of a protograph at time t in the coupled chain to its $m + 1$ protographs at indices $(t, t + 1, \cdots, t + m)$.

The interconnection is performed by randomly selecting an edge emanating from the VN of the protograph at time t and connecting it to the CN of any of the $m + 1$ of the neighboring graphs with a uniform probability. This is shown in Figure 7.1(b), this sort of interconnection procedure is known as *edge spreading*. After assignment of edges in this fashion, CNs of the first and the last m number of protographs at the boundaries of a coupled chain become irregular. VNs of the graphs with irregular CNs at the boundaries are initialized with known message bits during the encoding, and this *structured irregularity* is the reason of the remarkable threshold saturation phenomenon in the spatially coupled codes.

To determine the parity check matrix of the SC-LDPC with underlying $(3, 6)$ regular LDPC block code, consider the base matrix of the $(3, 6)$ protograph

$$\boldsymbol{B} = [3, 3].$$

It shows that a CN of the protograph is connected to two VNs, each via three edges. The edge spreading of photographs in a coupled chain become equivalent to splitting a block base matrix $\boldsymbol{B}$ of each protograph into $m + 1$ component sub-matrices $(\boldsymbol{B}_0, \cdots, \boldsymbol{B}_m)$, such that $\boldsymbol{B} = \boldsymbol{B}_0 + \cdots + \boldsymbol{B}_m$. For a coupled chain with $m = 2$, when edge spreading is performed in a way that the degree of all VNs in a coupled chain is preserved, then the component sub-matrices can be written as

$$\boldsymbol{B}_0 = \boldsymbol{B}_1 = \boldsymbol{B}_2 = [1, 1]$$

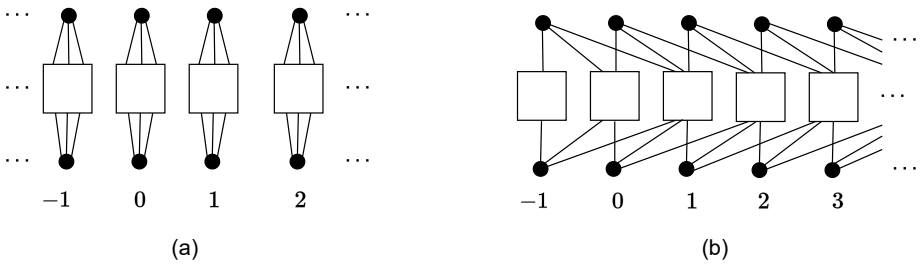


Figure 7.1: Construction of SC-LDPC ensemble from $(3, 6)$ protograph

The base matrix of the corresponding SC-LDPC ensemble $\boldsymbol{B}_{\text{sc}}$ is expressed as

$$\boldsymbol{B}_{\text{sc}} = \begin{bmatrix} \boldsymbol{B}_0 & & & \\ \boldsymbol{B}_1 & \boldsymbol{B}_0 & & \\ \vdots & \boldsymbol{B}_1 & \ddots & \\ \boldsymbol{B}_m & \vdots & \ddots & \boldsymbol{B}_0 \\ & \boldsymbol{B}_m & \ddots & \boldsymbol{B}_1 \\ & & \ddots & \vdots \\ & & & \boldsymbol{B}_m \end{bmatrix},$$

which also serves as the parity check matrix $\boldsymbol{H}_{\text{sc}}$ of the corresponding terminated SC-LDPC ensemble. Note that the edge spreading procedure described above is equivalent to the *unwrapping* procedure used to obtain LDPC-CCs in [34]. By noting this connection between edge spreading and unwrapping, we have proposed a new novel construction of SC-TCs in Paper III, which alleviates some of the shortcomings of the blockwise design methods of SC-TCs at smaller block lengths as observed in [41].

7.2 Spatially Coupled Turbo-Like Codes

Spatially coupled versions of turbo-like codes—BCC, SCC, PCC, and HCC—were introduced in [9]. The general principle behind the construction of SC-TCs is same as the SC-LDPCs. The construction techniques and DE equations of these SC-TCs for the BEC were comprehensively covered in [9,10], therefore we briefly revisit the construction principles of spatially coupled TCs, and give an example of threshold computation of SC-BCC on the BEC in a computationally efficient way. The efficient threshold computation principle is applicable for SC-TCs on the AWGN channel as well. The threshold computation of SC-TCs on the AWGN channel is part of this thesis work, and it is extensively covered in Paper II of this thesis.

7.2.1 Braided Convolutional Codes

Compact graph representation of original BCCs—a memory $m = 1$ coupled chain—is shown in Figure 7.2(a), along with its underlying uncoupled BCC in Figure 7.2(b). Each individual compact graph in a BCC chain contains an information node $\boldsymbol{u}_t$, an upper parity node $\boldsymbol{v}_t^{\text{U}}$, and a lower parity node $\boldsymbol{v}_t^{\text{L}}$, with upper and lower trellises denoted by T^{U} and T^{L} respectively. The parity sequences $\boldsymbol{v}_{t-1}^{\text{U}}$, $\boldsymbol{v}_{t-1}^{\text{L}}$ are used by the lower and upper encoders T^{L} and T^{U} of the compact graph at time t. Furthermore, these encoders also receive the corresponding message sequences from the node $\boldsymbol{u}_t$. Together with the message and parity sequences according to the compact graph of BCC in Figure 7.2(a), the encoders at time t produces a code sequence $\boldsymbol{v}_t = (\boldsymbol{u}_t, \boldsymbol{v}_t^{\text{U}}, \boldsymbol{v}_t^{\text{L}})$. The encoding process continues in a blockwise fashion until all the message sequences in a coupled chain are encoded. The structured irregularity at the boundaries of the BCC chain is introduced in a similar manner as for the SC-LDPCs.

A memory m SC-BCC can be obtained by splitting each parity sequence at time t into m equally sized sub-sequences, and spreading these sub-sequences

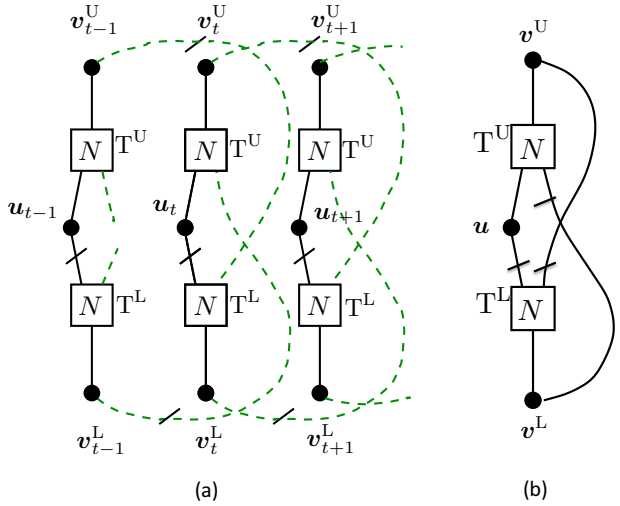


Figure 7.2: Compact graph representation of (a) $m = 1$ SC-BCC (b) Uncoupled BCC

.

to compact graphs at time $t+1, \cdots, t+m$. The spreading is performed by following the uniform edge spreading rule.

7.3 Threshold Computation of SC-BCCs on the BEC

DE analysis of a SC-BCC in its core implements the same DE equations on a BCC compact graph at time t in a chain as the tail biting or uncoupled BCC, which are described in Section 6.5.1. A DE iteration updates the CNs $\mathrm{T}^{\mathrm{U}}_t$ and $\mathrm{T}^{\mathrm{L}}_t$ after receiving the incoming erasure probabilities from VNs, and produces the erasure probabilities along the edges towards the VNs. The computation of incoming erasure probabilities to the CNs in case of SC-BCC, however, are slightly more involved than the simple BCCs. This is due to the interconnection of edges across the spatial positions in a chain, which the edge spreading rule poses. Let us see how these interconnections affect the DE equations of $\mathrm{T}^{\mathrm{U}}_t$ of a SC-BCC. The incoming erasure probabilities to the $\mathrm{T}^{\mathrm{U}}_t$ (VN updates) are obtained from

$$\begin{aligned} q^{(i,t)}_{\mathrm{L},1} &= \varepsilon \cdot p^{(i-1)}_{\mathrm{L},1} \\ q^{(i,t)}_{\mathrm{L},2} &= \frac{\varepsilon}{m} \sum_{j=1}^{m} p^{(i-1,t-j)}_{\mathrm{L},3} \\ q^{(i,t)}_{\mathrm{L},3} &= \frac{\varepsilon}{m} \sum_{j=1}^{m} p^{(i-1,t-j)}_{\mathrm{L},2}. \end{aligned} \tag{7.1}$$

From the above equation, we can observe that compared to the BCC, the incoming erasure probabilities along each parity edge to the CN $\mathrm{T}^{\mathrm{U}}_t$ in a coupled

chain are averaged from their corresponding de-multiplexers in the code structure. These averaged probabilities are then used during the DE update of $\mathrm{T}_t^{\mathrm{U}}$ to obtain the output erasure probabilities as

$$p_{\mathrm{U},k}^{(i,t)} = f_{\mathrm{U},k}\left(q_{\mathrm{L},1}^{(i,t)}, q_{\mathrm{L},2}^{(i)}, q_{\mathrm{L},3}^{(i,t)}\right) \text{ for } k = 1, 2, 3. \tag{7.2}$$

DE update of $\mathrm{T}_t^{\mathrm{L}}$ is analogous to $\mathrm{T}_t^{\mathrm{U}}$, and both CNs are processed in parallel. After the first iteration, the updated erasure probabilities $p_{\mathrm{U},1}^{(i,t)}$ and $p_{\mathrm{L},1}^{(i,t)}$ towards the information node are received, and a-posteriori erasure probility on $\boldsymbol{u}_t$ at time t and iteration i is obtained as

$$p_a^{(i,t)} = \varepsilon \cdot p_{\mathrm{U},1}^{(i,t)} \cdot p_{\mathrm{L},1}^{(i,t)}.$$

Chapter 8

Summary and Contributions

In this chapter, main contributions and results of the included papers are summarized. Furthermore, an overall conclusion of the papers, and future research options are also discussed.

8.1 Research Contributions

In this section, we briefly present the research topics that were investigated in this thesis, list research papers belonging to the research topics, and discuss the contributions of each paper in relation to the topic.

8.1.1 Part 1: Thresholds of SC-TCs on the AWGN channel

In this first research topic, we have performed the performance analysis of the TCs on the AWGN channel. To do this, we first review the techniques for computing iterative decoding thresholds of codes on the AWGN channel via the Monte-Carlo methods, and extend these to TCs. An efficient approach to predict the thresholds of TCs is proposed and benchmarked against the state-of-art threshold computation techniques and simulation results. The following papers are published under this research topic.

Paper I: Thresholds of Braided Convolutional Codes on the AWGN Channel

Turbo-like codes (TCs), which can be viewed as the class of GLDPCs with the component convolutional codes, were introduced earlier by our research group. It was concluded that the TCs optimized for MAP decoding performance are much more attractive than the turbo codes that are optimized for the BP performance. The conclusions, however, were primarily drawn by studying the thresholds of the considered codes on the binary erasure channel (BEC).

The AWGN thresholds computations of TCs and their spatially coupled versions is a formidable task compare to their BEC thresholds computations. The BEC thresholds can be determined by means of simple DE equations. The AWGN thresholds computation relies on the MC method which requires a significant computational effort. Furthermore, the reliability of the uncoupled and

spatially coupled TCs thresholds estimated via the commonly used thresholds analysis techniques is a question that requires investigation as well.

In this paper, we propose a computationally efficient approach of predicting the AWGN thresholds of the rate-compatible randomly punctured family of considered TCs. The prediction method is based on a simple transformation that uses the BEC threshold of a code, and return the predicted AWGN threshold. Accuracy of the prediction methods is determined by comparing the predicted thresholds with the MC-DE thresholds. Different versions of MC-DE techniques are benchmarked with each other to answer the question of reliability as well. These versions of MC-DE differs in the types of graphs of a considered ensemble, and the metric that is tracked during the iterative DE.

The results show that the MC-DE method that tracks individual messages probability densities within the graph is more reliable compared to the one that tracks the density or mutual information of an averaged message. The estimated thresholds were computed by collecting the statistics until the error rate of the extrinsic message became stabilized These statistics were used to approximate the probability density of the messages during the iterative MC-DE. The predicted thresholds are observed to be very accurate in case of randomly punctured spatially coupled BCCs. For high rate uncoupled BCCs obtained by randomly puncturing the mother code, the predicted thresholds are improved by incorporating the estimated AWGN threshold of the mother code ensemble into the threshold prediction method.

Paper II: Threshold Computation for Spatially Coupled Turbo-Like Codes on the AWGN Channel

The investigations in our first paper were restricted to randomly punctured uncoupled, and memory one spatially coupled BCCs. The prediction methods produce a fairly correct prediction of the thresholds of uncoupled BCCs, and a close to accurate prediction of SC-BCCs. In this article paper, we answer the question of reliability of the prediction methods for a broader range of TCs by observing the AWGN thresholds of coupled and uncoupled SCCs, PCCs, BCCs, and HCCs. The thresholds of the considered ensembles are obtained through the prediction methods introduced in the first paper, and the MC-DE method with histogram approximation of the probability density. Furthermore, we have introduced a *base matrix* characterization of the compact graph representation of the TCs. This characterization is useful in describing the structure of the spatially coupled TCs by means of a matrix.

The results show that the predicted thresholds are close to the MC-DE thresholds for strong spatially coupled TCs. By strong SC-TCs, we mean that the underlying uncoupled ensembles possess a strong MAP threshold. The analysis in this paper leads to a conjecture that the similarity between the predictions and the MC-DE thresholds is linked to the strength of an ensemble, and its threshold saturation capability. Performance simulations of some spatially coupled ensembles are also included. In this article, we have devoted a significant effort in computing the thresholds of the considered ensembles, to observe the threshold saturation phenomena, and the validation of the threshold prediction method.

8.1.2 Part 2: Periodic Convolutional Permutor Design for SC-SCCs

An effect of coupling memory, block length, window size, and a number of iterations on the performance, complexity, and latency of TCs was investigated in [41]. SC-SCCs codes were picked to perform these investigations due to their superior overall performance among other TCs. During the investigations, a high error floor was observed in the BER performance of SC-SCCs at short block lengths. This part is devoted on finding the solution to the high error-floor problem of SC-SCCs at short block lengths. Following paper is published under this research topic.

Paper III: Spatially-Coupled Serially Concatenated Codes with Periodic Convolutional Permutors

In this paper, we have investigated the problem of high error floor at shorter block lengths, and large coupling memories under a fixed decoding latency for the SC-SCCs. Classical construction methods for SC-TCs focus on block-wise design, and specifies independent permutors at each spatial position in a coupled chain. Sizes of these independent component permutors at each spatial position in a coupled chain is related to the message block length. Short messages length leads to a short component permutors which may entails an overall code with a poor minimimum distance properties. This situation can be avoided by a joint design of the permutors, which is a formidble task.

As an alternate to a joint design, we address the stated problem by proposing a family of blockwise periodically time-varying convolutional permutors. The proposed convolutional permutor for an entire coupled chain is constructed by applying an unwrapping technique to an optimized permutor of size equal to the desired code constraint. It is shown that the convolutional permutor obtained via unwrapping inherits the properties of its constituent permutor. In this way, the family of block-wise spatially coupled codes of varying block lengths constructed via convolutional permutors share the same permutor properties. We have used both pseudo-random and quadratic permutation polynomials permutors in the derivation of convolutional permutors for the SC-SCCs. We then bencmarked the performance of convolutional permutor based SC-SCCs with the classical SC-SCCs via simulations. Our results show that with convolutional permutor based SC-SCCs, it is possible to vary the component code block length and coupling memory at a fixed decoding latency and complexity without any noticeable performance loss.

The idea of convolutional permutor based spatially coupled ensemble construction can be further extended to the remaining TCs e.g., BCCs, PCCs, and HCCs. Furthermore, convolutional permutor based spatially coupled ensemble construction allows the message block length to go down to a bit level. This bit-wise spatially coupled ensemble may admits a a very large coupling memory. The investigation the bit-wise spatially coupled ensemble in the limits of coupling memory is an open task.

8.1.3 Part 3: Connection of TCs and LDPC codes

In the third research project, we generalize the BCCs, defined by a regular graph to a protograph of GLDPC codes with convolutional code constraints (CC-GLDPCs). We have first investigated thresholds and minimum distances of regular CC-GLDPCs and compared them to the thresholds and minimum distances of regular LDPC codes. Later, we have introduced irregularities in the CC-GLDPCs ensembles, and determined the design parameters of BP and MAP optimized irregular CC-GLDPCs by an exhaustive grid search using the BEC thresholds of the family of CC-GLDPCs. Following papers are published under this research topic.

Paper IV: Generalized LDPC Codes with Convolutional Code Constraints

This paper focuses on connecting the TCs with the LDPC codes. This connection requires a unified code construction framework. The LDPC codes are represented via the Tanner graphs, whereas the TCs are represented via the compact graphs. Both Tanner and compact graphs belongs to the class of generalized LDPC (GLDPC) codes. A GLDPC code is represented via *bi-partite* graph, that consists of generalized constraint nodes (CNs), and variable nodes (VNs). An LDPC codes are obtained by using component SPC codes at the CNs of a GLDPC graph. Similarly, TCs is obtained by using a component CCs at the CNs of GLDPCs graph. The GLDPCs offer a unified framework to view both TCs and LDPC codes from the same perspective.

The BCC compact graph is similar to the regular $(2, 3)$ Tanner graph. By generalizing the BCC compact graph, we introduce a family of regular GLDPC *bi-partite* graphs. Such regular *bi-partite* graphs offer a unified view for both CC-GLDPCs and LDPC codes, and allows a one-one comparison of both classes of codes.

The design of CC-GLDPCs is convenient than the conventional block codes based GLDPCs. In conventional GLDPCs, finding a component block code of rate matching with the code rate of the local constraints may pose challenges. This limitation often leads to altered graph structure, and a changed graph rate. A rate-compatible family of component convolutional codes can be easily matched to rates of the CNs of the GLDPC graph, thereby preserving the graph structure. Additionally, the CC-GLDPCs have the advantage of altering the component code strength without changing the graph rate.

In this paper, we have focused on the rate-1/3, and rate-1/2 regular CC-GLDPCs with component code trellises of memories of $(1, 2, 3)$. We have computed the thresholds and minimum distances of the considered CC-GLDPCs, and compared them to those of the corresponding LDPC codes. Following two objectives were sought from this investigation: first one was to compare the performance of CC-GLDPCs with a memory-1 component code trellis to that of the corresponding LDPCs, and the second one was to identify the performance trade-off in having a CC-GLDPC with a strong component code and smaller variable node degree to the one with a weak component code and larger variable node degree.

Results show that the considered CC-GLDPC graphs with a degree two variable node, and a memory one component code trellis outperform their counter-

part LDPCs in the BP decoding performance. Increasing the trellis memory of CC-GLDPCs resulted in a strong code with an improved minimum distance and MAP performance compared to their counterpart LDPCs, but with the degraded BP performance. The loss in BP performance is compensated by using spatial coupling to these strong ensembles. These spatially coupled ensembles were then observed to exhibit the threshold saturation phenomenon from the numerical results. We further observed that CC-GLDPCs of degree-2 VNs and stronger trellises excelled in terms of BP decoding and minimum distance performance compared to the LDPCs with larger variable node degrees.

Paper V: Improving the Thresholds of Generalized LDPC Codes with Convolutional Code Constraints

In the previous paper, we have investigated the regular CC-GLDPCs. Generalization to the irregular CC-GLDPCs and their thresholds analysis remained an open task. We expected further improvement in the performance by introducing irregularity to the CC-GLDPCs.

In this paper, we propose a generalized framework for the design of convolutional code (CC) based GLDPCs. A novel construction method of class of CC-GLDPCs is proposed. We leverage the BP EXIT function technique from the LDPC codes, and compute the BP and MAP thresholds of the family of CC-GLDPCs. By using these thresholds, design configurations of CC-GLDPCs with competitive thresholds are determined through an exhaustive grid search in the code space. The AWGN channel thresholds computations, and the BER simulations of the BP and MAP optimized CC-GLDPCs ensembles are performed as well.

8.2 General Conclusions

In this thesis, we have investigated the performance of TCs, and their spatially coupled versions on the AWGN channel through time consuming MC-DE methods. An efficient method to predict the AWGN thresholds of randomly punctured TCs and their spatially coupled versions is introduced. The prediction method connect the BP decoding performance of code ensembles on the BEC channel with their BP decoding performance on the AWGN channel. For capacity achieving SC-TCs, the predicted thresholds are observed to be very accurate compared to their MC-DE thresholds. It was demonstrated through simulations that the BP decoding performance of the SC-TCs on the AWGN channel achieves capacity, when their underlying TCs are optimized for MAP decoding performance, and these TCs possess strong minimum distance properties. The high error-floor problem at the short block lengths of the SC-TCs is solved by introducing a periodic block-wise time-varying convolutional permutor in this thesis. Lastly, a class of CC-GLDPCs is introduced, which connect the TCs with the LDPC codes, and offer a flexible code design framework than the conventional block codes based GLDPCs.

8.3 Future Research

Some interesting topics of future research on the SC-TCs are:

- Asymptotic analysis of bit-wise convolutional permutor based SC-TCs
- Performance, and complexity comparison of optimized CC-GLDPCs and optimized LDPC codes
- Performance analysis of CC-GLDPCs with optimized puncturing patterns, and optimized graphs girth
- Performance analysis of tail-biting CCs based GLDPCs

References

[1] W. Ryan and S. Lin, *Channel Codes*. USA: Cambridge University Press, 2012.

[2] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[3] P. Elias, "Error free coding," *IRE Trans. Inf. Theory*, vol. PGIT-4, pp. 29–37, 1954. Also in *Key Papers in Development of Coding Theory*, IEEE press, New York, NY, 1974.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings of ICC '93 - IEEE International Conference on Communications*, vol. 2, pp. 1064–1070, 1993.

[5] R. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[6] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar 1999.

[7] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. IT-27, pp. 533–547, Sep. 1981.

[8] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, 2001.

[9] S. Moloudi, M. Lentmaier, and A. G. i Amat, "Spatially coupled turbo-like codes," *IEEE Trans. Inf. Theory*, vol. 63, pp. 6199–6215, Oct. 2017.

[10] S. Moloudi, M. Lentmaier, and A. G. i Amat, "Spatially coupled hybrid concatenated codes," in *Proc. 11th Int. ITG Conf. Systems, Commun. and Coding (SCC)*, (Hamburg, Germany), 2017.

[11] S. Kudekar, T. Richardson, and R. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. Inform. Theory*, vol. 57, pp. 803 –834, Feb. 2011.

[12] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Spatially coupled turbo-like codes: A new trade-off between waterfall and error floor," *IEEE Trans. Commun.*, vol. 67, pp. 3114–3123, May 2019.

[13] T. M. Cover and J. A. Thomas, *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, July 2006.

[14] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[15] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, Mar. 1974.

[16] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "A soft-input soft-output maximum a posteriori (MAP) module to decode parallel and serial concatenated codes," *JPL TDA Progress Report*, vol. 42, pp. 1–20, Nov. 1996.

[17] M. Lentmaier, D. Truhachev, and K. Zigangirov, "Analytic expressions for the bit error probabilities of rate-1/2 memory 2 convolutional encoders," *IEEE Transactions on Information Theory*, vol. 50, pp. 1303–1311, June 2004.

[18] M. R. Best, M. V. Burnashev, Y. Y., A. Rabinovich, P. C. Fishburn, A. R. Calderburn, and D. J. Costello, Jr., "On a technique to calculate the exact performance of a convolutional code," *IEEE Transactions on Information Theory*, vol. IT-41, pp. 441–447, March 1995.

[19] B. Kurkoski, P. Siegel, and J. Wolf, "Exact probability of erasure and a decoding algorithm for convolutional codes on the binary erasure channel," in *GLOBECOM '03. IEEE Global Telecommunications Conference (IEEE Cat. No.03CH37489)*, vol. 3, pp. 1741–1745 vol.3, 2003.

[20] G. D. Forney Jr., *Concatenated Codes*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1965.

[21] T. Richardson and R. Urbanke, *Modern Coding Theory*. USA: Cambridge University Press, 2008.

[22] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," in *IPN Progress Report 42-154, JPL*, Aug. 2003.

[23] J. Hou, P. Siegel, L. Milstein, and D. Pfister, "Multilevel coding with low-density parity-check component codes," in *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)*, vol. 2, pp. 1016–1020 vol.2, 2001.

[24] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 670–678, 2004.

[25] R. Johannesson and K.Sh. Zigangirov, *Fundamentals of Convolutional Coding*. Piscataway, NJ: IEEE Press, 2nd ed., 2015.

[26] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding," Technical Specification (TS) 36.212, 3rd Generation Partnership Project (3GPP), Sep 2019. Version 15.3.0.

[27] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, 2001.

[28] Jilei Hou, P. H. Siegel, L. B. Milstein, and D. Pfister, "Multilevel coding with low-density parity-check component codes," in *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)*, vol. 2, pp. 1016–1020 vol.2, 2001.

[29] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 909–926, 1998.

[30] D. Divsalar, J. Jin, and R. McEliece, "Coding theorems for turbo-like codes," *Proc. 36th Annu. Allerton Conf. Communication, Control, and Computing*, Sep. 1998.

[31] S. Abu-Surra, D. Divsalar, and W. E. Ryan, "Enumerators for protograph-based ensembles of ldpc and generalized ldpc codes," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 858–886, 2011.

[32] S. Abu-Surra, W. E. Ryan, and D. Divsalar, "Ensemble enumerators for protograph-based generalized ldpc codes," in *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*, pp. 1492–1497, 2007.

[33] D. J. Costello, L. Dolecek, T. E. Fuja, J. Kliewer, D. G. Mitchell, and R. Smarandache, "Spatially coupled sparse codes on graphs: theory and practice," *IEEE Communications Magazine*, vol. 52, no. 7, pp. 168–176, 2014.

[34] A. Jimenez and K. Zigangirov, "Periodic time-varying convolutional codes with low-density parity-check matrices," in *Proceedings. 1998 IEEE International Symposium on Information Theory (Cat. No.98CH36252)*, pp. 305–, 1998.

[35] M. Lentmaier, A. Sridharan, K. S. Zigangirov, and D. J. Costello, Jr., "Terminated LDPC convolutional codes with thresholds close to capacity," in *Proc. IEEE International Symposium on Information Theory*, (Adelaide, Australia), pp. 1372–1376, Sep. 2005.

[36] M. Lentmaier, A. Sridharan, D.J. Costello, Jr., and K.Sh. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes." *IEEE Trans. Inf. Theory*, accepted for publication. See also `http://www.vodafone-chair.com/staff/lentmaier/LDPCCCThresholds_TransIT.pdf`.

[37] A. Feltström, D. Truhachev, M. Lentmaier, and K. Zigangirov, "Braided block codes," *IEEE Trans. Inform. Theory*, vol. 55, pp. 2640–2658, June 2009.

[38] Y.-Y. Jian, H. D. Pfister, K. R. Narayanan, R. Rao, and R. Mazahreh, "Iterative hard-decision decoding of braided bch codes for high-speed optical communication," in *2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 2376–2381, 2013.

[39] J. Justesen, K. J. Larsen, and L. A. Pedersen, "Error correcting coding for otn," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 70–75, 2010.

[40] S. Moloudi and M. Lentmaier, "Density evolution analysis of braided convolutional codes on the erasure channel," in *Proc. IEEE International Symposium on Information Theory*, (Honolulu, HI), July 2014.

[41] M. Mahdavi, M. Umar Farooq, L. Liu, O. Edfors, V. Öwall, and M. Lentmaier, "The effect of coupling memory and block length on spatially coupled serially concatenated codes," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pp. 1–7, 2021.

Part II

Included Papers

Paper I

Thresholds of Braided Convolutional Codes on the AWGN Channel

In this paper, we perform a threshold analysis of braided convolutional codes on the additive white Gaussian noise (AWGN) channel. The decoding thresholds are estimated by Monte-Carlo density evolution techniques and compared with approximate thresholds from an erasure channel prediction. The results show that, with spatial coupling, the predicted thresholds are very accurate and quickly approach capacity if the coupling memory is increased. For uncoupled ensembles with random puncturing, the prediction can be improved with help of the AWGN threshold of the unpunctured ensemble.

M. U. Farooq, S. Moloudi and M. Lentmaier,
"Thresholds of Braided Convolutional Codes on the AWGN Channel", *2018 IEEE International Symposium on Information Theory (ISIT)*, Vail, CO, USA, 2018.

1 Introduction

Braided convolutional codes (BCCs) [1–3] are a class of spatially-coupled (SC) turbo-like codes with regular graph structure. On the binary erasure channel (BEC), explicit density evolution (DE) equations have been derived for BCCs in [4], which can be used to efficiently compute exact decoding thresholds for that channel. The results show that BCCs have superior maximum-a-posteriori (MAP) decoding thresholds compared to parallel or serially concatenated codes on the BEC [5]. Furthermore it has been proven analytically in [5] that threshold saturation occurs, i.e., with spatial coupling a belief propagation (BP) decoder can achieve the same threshold as an optimal MAP decoder.

The aim of this paper is to compute the BP thresholds of BCCs on the additive white Gaussian noise (AWGN) channel. For this channel, exact DE equations are not available for turbo-like codes, and Monte Carlo (MC) methods are usually applied to estimate decoding thresholds. One of the difficulties of such an approach is that the graphs of spatially coupled systems contain a large number of edge types whose message densities have to be considered individually during DE. This requires significantly larger computational efforts than classical methods, like the single edge-type extrinsic information transfer characteristics (EXIT) chart analysis [6].

We use Monte-Carlo density evolution (MC-DE) to estimate the thresholds of uncoupled and coupled BCCs with and without puncturing. As an efficient alternative, we then consider the erasure channel approximation by Chung [7] for predicting the AWGN channel thresholds from those of the BEC and compare the results. Finally, we demonstrate that for randomly punctured ensembles, analogously to LDPC codes [8], the thresholds of BCCs of all higher rates can immediately be predicted from the unpunctured thresholds of the BEC and/or the AWGN channel. Some simulation results are also given.

2 Braided Convolutional Codes

BCCs were originally introduced in [1]. Their characteristics is that the parity symbols of one component encoder are used as information symbols of the other and vice versa. Due to this, both information and parity symbols are protected by both component codes in a symmetric way. In this paper we consider an example of BCCs of rate $R = 1/3$, which are defined by using two systematic component convolutional encoders of rate $R_{cc} = 2/3$ and three permutors of length N. The component code and the encoding method is same as used in [5]. In particular, we are using the blockwise BCCs, for which an encoder diagram is shown in Fig. 1. The parity symbols created by one encoder at time t pass a delay of one block, D^N and a permutor before entering the other encoder at time $t + 1$ [2].

To compare BCCs with parallel or serially concatenated codes, it sometimes can be useful to define an uncoupled equivalent of BCCs. This can be achieved by removing the delay in the encoder depicted in Fig. 1. Since the feedback now occurs without the delay, a straightforward encoding by means of the trellis is not possible. But the code is still well defined by the trellis constraints that the code symbols have to satisfy.

Turbo-like codes, like LDPC codes, can be described by factor graphs [9,10].

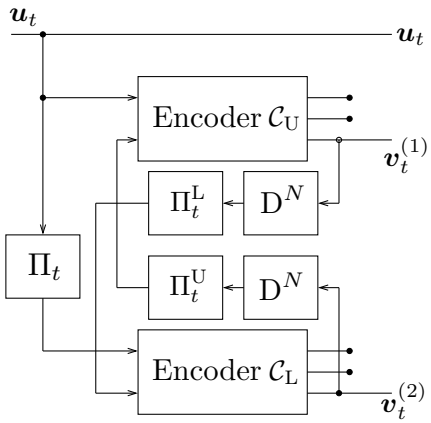


Figure 1: Blockwise BCCs: turbo-like codes with parity feedback ($R = 1/3$).

This way of expressing a code is useful in describing the exchange of messages in an iterative BP decoder, as well as for the corresponding DE analysis. Instead of a conventional factor graph, we use a compact graph representation as introduced in [5]. The compact graph of a BCC and its uncoupled equivalent is shown in Fig. 2. Each block of symbols is represented by a variable node and each trellis by a factor node. A permutor is indicated by a short line that crosses an edge in the graph.

Similar to SC-LDPC codes, a BCC code can be obtained by repeating the graph of an uncoupled code and spreading some edges across $m+1$ neighboring blocks, where m denotes the coupling memory. The original BCCs shown in Fig. 2(a) have coupling memory $m = 1$.

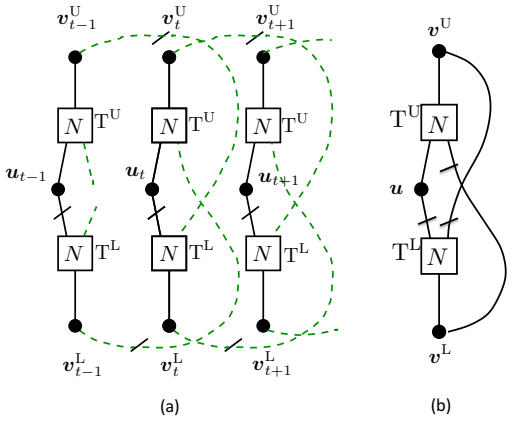


Figure 2: Compact graph representation.

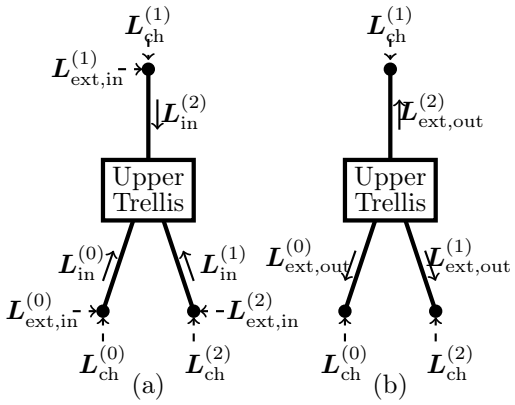


Figure 3: Monte Carlo density evolution process.

3 Monte Carlo Density Evolution

In order to describe the MC-DE process, let us consider the upper decoder of the uncoupled (UC) BCC shown in Fig. 2(b). The exchange of messages with the upper decoder is depicted in Fig. 3. Due to symmetry, the processing at the lower decoder follows analogously. In each iteration of MC-DE, assuming a flooding schedule, the decoding at the upper and lower decoder is performed independently in parallel and the *densities* of updated output messages are exchanged. Iterative exchange of densities continues in this fashion until the decoding error probability converges. The key points of the MC-DE process can be summarized in the following three major steps:

1. *Variable node update:* Each variable node $k = 0, 1, 2$ in Fig. 3(a) takes the sequence $\boldsymbol{L}_{\text{ch}}^{(k)}$ of channel LLRs and the sequence $\boldsymbol{L}_{\text{ext,in}}^{(j)}$ of incoming extrinsic LLRs received from output j of the connected lower trellis node and combines them to the updated message sequence $\boldsymbol{L}_{\text{in}}^{(i)} = \boldsymbol{L}_{\text{ch}}^{(k)} + \boldsymbol{L}_{\text{ext,in}}^{(j)}$, which forms the input i of the upper trellis node. All sequences are of equal length N.

2. *Trellis node update:* The trellis node receives the three input sequences from the different variable nodes corresponding to symbol blocks $\boldsymbol{u}$, $\boldsymbol{v}^{\text{U}}$ and $\boldsymbol{v}^{\text{L}}$. The node performs decoding and produces the updated sequences $\boldsymbol{L}_{\text{ext,out}}^{(i)}$ of extrinsic LLRs at each output $i = 0, 1, 2$ of the trellis node in Fig. 3(b). Finally, these output message sequences are used to estimate the message densities $f(L_{\text{ext,out}}^{(i)})$.

3. *Drawing samples from message densities:* In this step independent LLR sequences $\boldsymbol{L}_{\text{ch}}^{(k)}$, $k = 0, 1, 2$ and $\boldsymbol{L}_{\text{ext,in}}^{(j)}$, $j = 0, 1, 2$ are created from the channel density and the densities $f(L_{\text{ext,out}}^{(j)})$ received from the lower trellis node. These sequences are used in the next MC-DE iteration.

In the literature, the message densities $f(L_{\text{ext,out}}^{(i)})$ are often *approximated* by Gaussian densities as well in MC-DE. In this case, it is not necessary to estimate the exact $f(L_{\text{ext,out}}^{(i)})$ in step 2. Instead, some parameter like the mean,

Thresholds E_b/N_0 (dB)	**Pattern**	**Rate** 1/3	1/2	2/3
MC	P1	1.003	2.408	-
GA	P1	1.023	2.399	-
GA SE	P1	1.050	2.708	-
MC	P2	1.003	2.121	4.151
GA	P2	1.018	2.128	4.062
GA SE	P2	1.048	2.161	4.131

Table 1: Uncoupled BCC Thresholds

the variance or the mutual information corresponding to the LLR sequences is computed to characterize the Gaussian density.

Table 1 shows the uncoupled BCC thresholds obtained via MC-DE with two different puncturing methods. P2 puncturing refers to when uniform random puncturing is applied on both information and parity bits, whereas in P1 random puncturing is applied only on parity bits. MC refers to the threshold obtained via the MC-DE algorithm and GA refers to the threshold when $f(L^{(i)}_{\text{ext,out}})$ is approximated by a Gaussian density and $\boldsymbol{L}^{(i)}_{\text{in}}$ are drawn from it.

In determining MC and GA thresholds, the statistics along the incoming edges to the trellis nodes are different. It means that for a code with multi edge-types (ME), such as BCCs, the $f(L^{(i)}_{\text{in}})$ densities are different along each incoming edge. In the literature, these densities $f(L^{(i)}_{\text{in}})$ are often averaged to obtain a single density $f(L_{\text{in}})$, which then is used along all incoming edges to the trellis nodes. This reduces the complexity of DE analysis, but the corresponding single edge-type (SE) ensemble can have a different threshold.

In Table 1 it can be observed that the MC thresholds are closer to the GA thresholds for low rates. Whereas, the difference in these thresholds becomes high as the fraction of puncturing increases. With punctured bits, $f(L^{(i)}_{\text{in}})$ will be a mixture of the Gaussian and erasure densities. The erasure density will be dominant in this mixture at higher rate. Therefore we expect that the Gaussian approximation of the densities at higher rate results in inaccuracies. Furthermore, the difference from GA SE thresholds to other thresholds is relatively larger. The GA SE case is equivalent to the classical EXIT chart technique. In case of ME ensembles, the MC-DE process with GA of densities $f(L^{(i)}_{\text{in}})$ is a generalization of the protograph LDPC EXIT analysis technique [11] to component codes with trellis constraints.

In order to get a high accurancy of estimated thresholds via MC-DE, a large number of trellis node output messages are collected before computing the densities within each iteration. In our experiments, the statistics are considered stabilized when the bit error rate (BER) as a function of the number of simulated blocks within an iteration reaches a steady state with a difference of .0001. This requirement of accuracy control makes MC-DE take days to compute UC-BCCs thresholds and it takes even much longer for the SC-BCCs. The accuracy of the MC-DE thresholds can be increased by running MC-DE for a longer time.

Life can be much easier if we can predict the thresholds on the Gaussian

		Erasure		UC BCC		SC BCC	
Rate	Pattern	ϵ^1_{SC}	ϵ_{UC}	ECP	MC	ECP	MC
1/3	P1	0.6609	0.5541	1.21	1.00	-0.39	-0.39
1/2	P1	0.4932	0.3013	2.71	2.40	0.27	0.25
2/3	P1	0.3257	-	-	-	1.15	1.12
3/4	P1	0.2411	-	-	-	1.74	1.70
4/5	P1	0.1915	-	-	-	2.16	2.12
1/3	P2	0.6609	0.5541	1.21	1.00	-0.40	-0.39
1/2	P2	0.4914	0.3312	2.33	2.12	0.30	0.29
2/3	P2	0.3219	0.1083	4.33	4.15	1.20	1.18
3/4	P2	0.2371	-	-	-	1.80	1.77
4/5	P2	0.1862	-	-	-	2.24	2.21

Table 2: Predicted vs Estimated Thresholds (E_b/N_0) of BCC

channel reliably and much faster than the MC-DE does. One way would be to use the exact DE equations as is used in [5] for BCCs over the BEC. However, the DE equations for the BCCs on the AWGN channel are not available. Therefore, we have to look for an alternative solution to find the thresholds of BCCs much faster than MC-DE does.

4 Erasure Channel Prediction of AWGN Channel Thresholds

It has been observed in [7] that the thresholds of LDPC codes on the AWGN channel can be *approximated* by the corresponding thresholds on the BEC. Such an erasure channel prediction (ECP) is computationally attractive for turbo-like code ensembles, since BEC thresholds can be computed exactly with relatively small efforts. In this section we will apply this approach to uncoupled and coupled BCC ensembles and compare the resulting thresholds with those obtained from MC-DE.

Let $C_E(\varepsilon) = 1 - \varepsilon$ denote the capacity of a BEC with erasure probability ε and $C_G(\sigma)$ denote the capacity of a binary-input AWGN channel with noise variance σ^2. Let ε^* and σ^* denote the DE thresholds computed on the two types of channels for a given code ensemble. The ECP is based on the observation that $C_E(\varepsilon^*) \approx C_G(\sigma^*)$, which suggests the approximation

$$\sigma^* \approx C_G^{-1}\left(C_E(\varepsilon^*)\right) = C_G^{-1}\left(1 - \varepsilon^*\right) \ . \tag{1}$$

Using (1), the AWGN threshold of a given ensemble can now be predicted from the corresponding BEC threshold ε^*. Table 2 shows the resulting predicted thresholds for UC-BCCs and SC-BCCs along with the corresponding MC-DE thresholds. The original BEC thresholds are also given, where the values of puncturing pattern P1 are identical to Table II of [5]. We can see from Table 2 that the ECP thresholds of SC-BCCs are quite close to the MC thresholds. However, the difference is larger on higher rates than it is on lower rates as

we have observed in the UC-BCCs thresholds. The ECP and MC thresholds of the UC-BCCs have bigger difference than the SC-BCCs. Furthermore, the ECP and the MC thresholds are much closer to each other on P2 puncturing compared to P1 puncturing.

5 Thresholds of Randomly Punctured Ensembles

Can we predict the thresholds such that the difference of the predicted and the estimated threshold does not increase as the amount of puncturing increases? In [8], it has been shown that for the P2 punctured LDPC codes, it is possible to closely predict the thresholds on the AWGN channel, given just the threshold of the unpunctured code ensemble and the design rate R. In the following, we will apply the methods discussed in [8] to the randomly punctured BCC ensembles and discuss the results obtained by it.

5.1 θ_E Predictions

Consider random puncturing with a fraction $\alpha = p/n$, where p denotes the number of punctured bits and n the total number of bits before puncturing. For the BEC, the threshold $\epsilon^*(\alpha)$ of the punctured ensemble is equal to

$$\epsilon^*(\alpha) = 1 - \theta_E R(\alpha) \ , \tag{2}$$

where

$$R(\alpha) = \frac{R}{1-\alpha} \tag{3}$$

denotes the desired rate after puncturing and

$$\theta_E = \frac{1-\epsilon^*}{R} \tag{4}$$

and is a parameter that follows from the threshold ϵ^* and rate R of the unpunctured ensemble.

The elegance of this method is that it will provide the exact BEC thresholds for all rates $R(\alpha)$, $R \leq R(\alpha) \leq 1/\theta_E$, based on a single parameter θ_E and hence it is not required to perform DE on all different rates. However, this method is limited to the puncturing pattern P2 only.

Once we know the BEC thresholds, we can apply the ECP method discussed in Section 4. Equivalently, we can write

$$h_G(\sigma^*(\alpha)) \approx h_E(\varepsilon^*(\alpha)) = \epsilon^*(\alpha) = 1 - \theta_E R(\alpha) \ , \tag{5}$$

where $h_G(\sigma) = 1 - C_G(\sigma)$ and $h_E(\varepsilon) = 1 - C_E(\varepsilon) = \varepsilon$ denote the conditional entropy of the AWGN channel and the BEC, respectively. From (5) we can compute the AWGN thresholds in terms of standard deviation σ or signal-to-noise ratio E_b/N_0. The results obtained via this method for UC-BCCs are given in Table 3.

Thresholds E_b/N_0 (dB)	**Pattern**	**Rate** 1/3	1/2	2/3
MC-DE	P2	1.00	2.12	4.15
θ_E Predicted	P2	1.21	2.33	4.33
θ_G Predicted	P2	1.00	2.05	3.79
MP (θ_E and θ_G)	P2	1.00	2.19	4.19
Erasure Probability	P2	0.5541	0.3312	0.1083

Table 3: Uncoupled BCC Thresholds

5.2 θ_G Predictions

While the θ_E prediction works especially well for high code rates, it can be further improved for low rates if the unpunctured AWGN threshold σ^* is available. Then

$$h_G(\sigma^*(\alpha)) \approx 1 - \theta_G R(\alpha) \ , \tag{6}$$

where

$$\theta_G = \frac{1 - h_G(\sigma^*)}{R} \ . \tag{7}$$

It can be seen in Table 3 that for the lower code rates, the MC thresholds and the θ_G predictions are quite close. However, for a higher fraction of punctured bits, the MC thresholds are closer to the θ_E predictions.

5.3 Mixed Predictions

To account for the higher puncturing fraction in the prediction of thresholds on the AWGN channel, a mixed prediction (MP) method is suggested in [8], where both θ_E and $h_G(\sigma^*)$ are used.

The important idea behind the mixed prediction method is that all the punctured code ensemble thresholds on the BEC, when viewed in the entropy domain, lie on a straight line. The slope of this line is characterized by θ_E. It is further demonstrated in [8] that for P2 punctured LDPC codes on the AWGN channel the estimated entropies, corresponding to the E_b/N_0 thresholds, are observed to follow a straight line.

From the observation that at lower rate the estimated thresholds are closer to the θ_G predictions and at higher rates are closer to the θ_E predictions, we obtain the following mixed prediction

$$h_G(\sigma_{BP}(\alpha)) \approx \frac{(R(\alpha) - R) \cdot (1 - \theta_E - h_G(\sigma^*))}{1 - R} + h_G(\sigma^*), \tag{8}$$

where $R \leq R(\alpha) \leq R_{\max}$. $R_{\max}$ is an intercept that can be obtained by setting $h_G(\sigma_{BP}(\alpha)) = 0$ in (8). This straight line prediction passes through the unpunctured rate $R(0)$ and the associated threshold (entropy) on the AWGN channel and $R(\alpha) = 1$ and its associated threshold (entropy) on the BEC. The

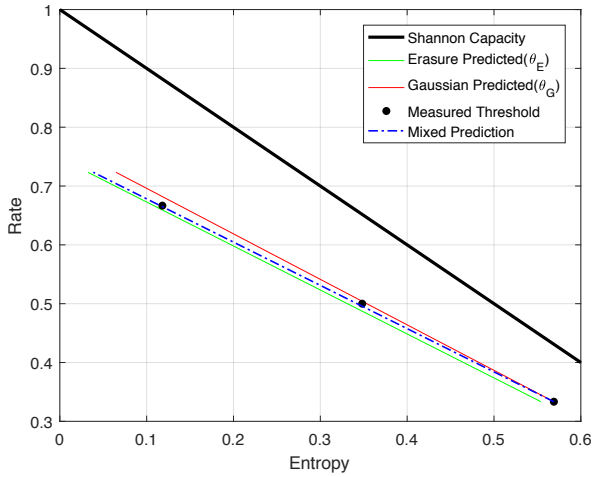


Figure 4: Entropy vs rate of UC-BCCs.

Thresholds			**Rate**		
E_b/N_0 (dB)	1/3	1/2	2/3	3/4	4/5
Shannon Capacity	-0.50	0.18	1.06	1.63	2.05
θ_E prediction	-0.4585	0.2307	1.1151	1.6924	2.1166
Erasure Probability	0.6644	0.4967	0.3289	0.2450	0.1947

Table 4: BCC coupled - $m = 3$

results of the mixed predictions are shown in Fig. 4 for the P2 punctured UC-BCCs ensembles. The estimated MC-DE thresholds almost follow the mixed prediction line in this figure. However, the estimated thresholds do not strictly lie on a straight line as was the case in [8]. The related E_b/N_0 thresholds of UC-BCCs are also given in Table 3.

It can be seen in Table 2 that for SC-BCCs, the MC-DE and predicted thresholds are almost identical. As a result of threshold saturation, the SC-BCCs performance is much better compared to the performance of UC-BCCs and the gap to capacity is much smaller for SC-BCCs, which can be seen in Fig. 5. θ_E and θ_G predictions for the punctured UC-BCCs have been made by using $\theta_E = 1.337$ and $\theta_G = 1.293$ respectively. Similarly, θ_E predictions for the punctured $m = 1$ SC-BCCs have been made by using $\theta_E = 1.017$. Since the θ_E predictions are very close to the MC-DE thresholds, θ_G and mixed predictions are hard to distinguish from them.

Consider now a larger coupling memory $m = 3$. The predicted thresholds are given in Table 4, where it can be seen that $m = 3$ SC-BCCs outperform $m = 1$ SC-BCCs and operate very close to the Shannon limit. Again, the θ_E predictions and MC-DE thresholds are very close to each other and we have only provided the θ_E predictions in this table.

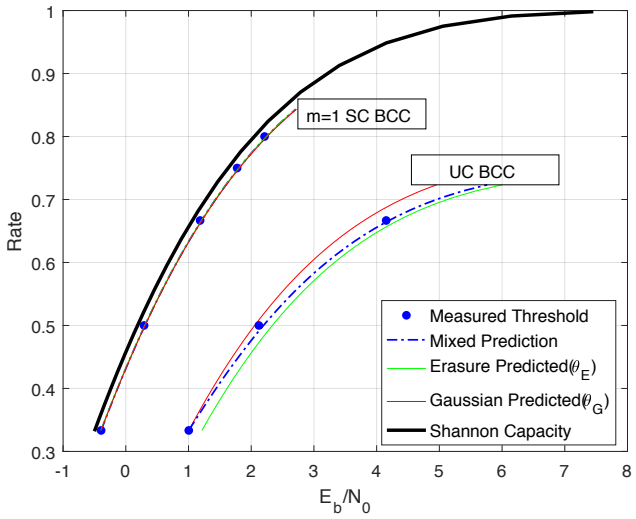


Figure 5: E_b/N_0 vs rate of SC and UC-BCCs.

5.4 Simulation Results

The simulated BER performance of P2 punctured $m = 1$ SC-BCCs on the AWGN channel is shown in Fig. 6. The simulations are obtained with a sliding window decoder [3] with window size $w = 5$, and 20 iterations at each window position. We can see from Fig. 6 that for all the considered rates $R = 1/3, 1/2, 2/3$ the BER curves are in accordance with their corresponding thresholds.

6 Conclusions

In this paper, have presented MC-DE as a technique to compute the AWGN channel thresholds of spatially-coupled turbo-like codes. Furthermore, we have applied the threshold prediction methods presented for LDPC codes in [8] for predicting BCC thresholds. The θ_E and θ_G predictions of the UC and SC-BCC ensembles have been compared with the estimated thresholds obtained by MC-DE. The results show that, with spatial coupling, the predicted thresholds are very accurate and quickly approach capacity if the coupling memory is increased. For uncoupled ensembles with random puncturing, the prediction can be improved with help of the AWGN threshold of the unpunctured ensemble.

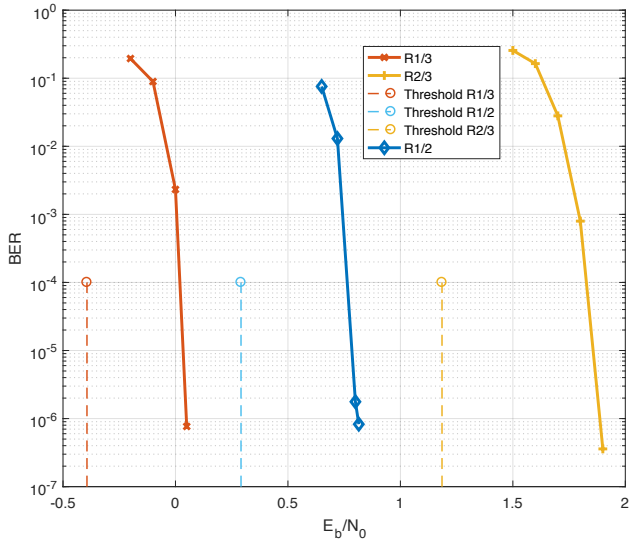


Figure 6: BER vs E_b/N_0 thresholds of P2 punctured SC-BCCs.

References

[1] W. Zhang, M. Lentmaier, K. S. Zigangirov, and D. J. Costello, Jr., "Braided convolutional codes: A new class of turbo-like codes," *IEEE Transactions on Information Theory*, vol. 56, pp. 316–331, Jan. 2010.

[2] D. J. Costello, Jr., M. Lentmaier, and D. G. M. Mitchell, "New perspectives on braided convolutional codes," in *Proc. IEEE 9th International Symposium on Turbo codes and Iterative Information Processing*, (Brest, France), Sep. 2016.

[3] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, Jr., and B. Bai, "Braided convolutional codes with sliding window decoding," *IEEE Transactions on Communications*, vol. 65, pp. 3645–3658, Sep. 2017.

[4] S. Moloudi and M. Lentmaier, "Density evolution analysis of braided convolutional codes on the erasure channel," in *Proc. IEEE International Symposium on Information Theory*, (Honolulu, HI), July 2014.

[5] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Spatially coupled turbo-liked codes," *IEEE Transactions on Information Theory*, vol. 63, pp. 6199–6215, Jan. 2017.

[6] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 40, pp. 1727–1737, Oct. 2001.

[7] S.-Y. Chung, *On the construction of some capacity-approaching coding schemes*. PhD thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, Sept. 2000.

[8] D. G. M. Mitchell, M. Lentmaier, A. E. Pusane, and D. J. Costello, Jr., "Randomly punctured LDPC codes," *IEEE Journal on Selected Areas in Communications*, vol. 34, pp. 408–421, Feb. 2016.

[9] N. Wiberg, H. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *European Transactions on Telecommunications*, vol. 6, pp. 513–525, Sep. 1995.

[10] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, Feb. 2001.

[11] G. Liva and M. Chiani, "Protograph LDPC codes design based on EXIT analysis," in *Proc. IEEE Globecom*, (Washington, DC), Nov. 2007.

Paper II

Threshold Computation for Spatially Coupled Turbo-Like Codes on the AWGN Channel

In this paper, we perform a belief propagation (BP) decoding threshold analysis of spatially coupled (SC) turbo-like codes (TCs) (SC-TCs) on the additive white Gaussian noise (AWGN) channel. We review Monte-Carlo density evolution (MC-DE) and efficient prediction methods, which determine the BP thresholds of SC-TCs over the AWGN channel. We demonstrate that instead of performing time-consuming MC-DE computations, the BP threshold of SC-TCs over the AWGN channel can be predicted very efficiently from their binary erasure channel (BEC) thresholds. From threshold results, we conjecture that the similarity of MC-DE and predicted thresholds is related to the threshold saturation capability as well as capacity-approaching maximum a posteriori (MAP) performance of an SC-TC ensemble.

M. U. Farooq, A. G. i. Amat and M. Lentmaier,
"Threshold Computation for Spatially Coupled Turbo-Like Codes on the AWGN Channel", *Entropy* 2021, 23, 240.

1 Introduction

Turbo-like codes (TCs) [1]—such as parallel concatenated codes (PCCs) and serially concatenated codes (SCCs)—and low-density parity-check (LDPC) codes [2] are widely used in communication systems due to their excellent performance and low-complexity decoding. In most cases, the design of these codes is based on the optimization of the iterative belief propagation (BP) decoding threshold, which can be performed via density evolution (DE).

The exact BP thresholds of LDPC codes over the binary erasure channel (BEC) can be easily obtained recursively from a set of DE equations using a scalar representation of the message densities, whereas for the AWGN channel they can be obtained via quantized DE [3]. Alternatively, the BP threshold may be estimated by means of extrinsic information transfer (EXIT) function analysis [4, 5], where the densities of the messages are approximated by a Gaussian distribution. For the AWGN channel and binary transmission, the Gaussian approximation yields thresholds close to those obtained via DE, while simplifying the computation. The BP thresholds of the major TC ensembles—PCCs, SCCs, braided convolutional codes (BCCs), and hybrid concatenated codes (HCCs)—over the BEC were computed by Moloudi et al. in [6, 7] by using the decoder transfer functions of the component codes that map the input and output erasure probabilities of the message sequences. Unfortunately, the decoder transfer functions are not available for the AWGN channel, which hinders the derivation of the exact DE equations. In [8], a Monte Carlo (MC)-based DE (MC-DE) was proposed for the threshold analysis of BCCs over the AWGN channel. The MC-DE, however, is computationally demanding compared to the simple DE for TCs over the BEC. Moreover, MC-DE for TCs, which entails running BCJR decoding of the component codes, is significantly harder than the quantized DE or the EXIT chart technique for LDPC codes.

Spatial coupling [9] allows us to construct particularly powerful codes. Thanks to the threshold saturation phenomenon, the BP decoding threshold of a spatially coupled ensemble can achieve the maximum a posteriori (MAP) decoding threshold of the underlying uncoupled ensemble. Remarkably, spatially coupled LDPC codes universally achieve capacity over the class of binary-input memoryless symmetric channels [10]. The concept of spatial coupling was extended to turbo-like codes in [6].

Quantized DE for SC-LDPC codes is time consuming, due to the large number of edge types in the corresponding graph. The complexity of MC-DE of spatially coupled (SC) TCs (SC-TCs) is even higher, and hence the computation of the thresholds becomes challenging. Efficient methods that allow to accurately predict the BP thresholds are therefore of practical interest. In [11], the BP thresholds of randomly-punctured LDPC codes over the AWGN channel were efficiently predicted by using their corresponding BEC thresholds. The same idea was later used in [8] to predict the thresholds of BCCs over the AWGN channel using the BEC thresholds of BCCs [6]. The resulting thresholds for SC-BCCs, which have a regular graph representation, are close to those obtained using MC-DE, which can be attributed to the universality of this ensemble.

In this paper, we perform a comprehensive threshold analysis of several classes of uncoupled and coupled TCs—PCCs, SCCs, BCCs, and HCCs—over the AWGN channel, by discussing several efficient methods for threshold computation. More concretely, we review MC-DE with Gaussian approximation

and MC-DE where the true densities are estimated using histograms. We also discuss the prediction of the BP thresholds using the thresholds of the corresponding ensembles over the BEC. Further, we discuss the efficient computation of the BP thresholds of punctured TCs from those of the corresponding mother code. We show that for spatially coupled TC ensembles with strong underlying uncoupled code, a very accurate prediction of the BP threshold over the AWGN channel can be efficiently obtained for a large range of coding rates from the BP threshold of the corresponding mother code ensembles over the BEC. We conjecture that the accurate predictions can be attributed to the universality of these code ensembles due to threshold saturation.

The rest of the paper is structured as follows. In Section 2, the construction of uncoupled and coupled TC ensembles is described using uncoupled and coupled SCCs as an example. A base matrix representation is introduced, which is then used to define the remaining ensembles. In Section 3, MC-DE for the computation of the BP thresholds of TCs is described in detail. In Section 4, we discuss threshold prediction methods for randomly punctured TCs. In Section 5, we compare and discuss the thresholds computed via the different methods for several classes of uncoupled and coupled TCs. Finally, Section 6 concludes this work.

2 Preliminaries

We consider the TC ensembles in [6, 7] with and without coupling. In this section, we describe these ensembles by discussing SCCs and SC-SCCs of coupling memory $m = 1$ and refer the reader to [6] and [7] for detailed description of other TCs. In order to efficiently describe the coupling for each of the ensembles, we introduce a new base matrix representation corresponding to the compact graph representation of the ensembles in [6]. Lastly, we briefly describe the sliding window decoder, which is used in this work to carry out the threshold analysis.

2.1 Code Ensembles

The block diagram of a rate-1/4 SCC encoder is shown in Figure 1a. The encoder is constructed from two recursive systematic convolutional encoders, referred to as outer and inner encoders. The information sequence $\boldsymbol{u}$ is first encoded by the outer encoder $\mathcal{C}^{\mathrm{O}}$, resulting in the encoded sequence $\boldsymbol{v}^{\mathrm{O}}$. The sequence $(\boldsymbol{u}, \boldsymbol{v}^{\mathrm{O}})$ is reordered by a permuter $\boldsymbol{\Pi}$ and then encoded by the inner encoder $\mathcal{C}^{\mathrm{I}}$, producing the parity sequence $\boldsymbol{v}^{\mathrm{I}}$. The codeword sequence $\boldsymbol{v} = \left(\boldsymbol{u}, \boldsymbol{v}^{\mathrm{O}}, \boldsymbol{v}^{\mathrm{I}}\right)$ is obtained at the output of the SCC encoder after following these encoding steps.

The compact graph of the SCC ensemble is shown in Figure 1b. The sequences $\boldsymbol{u}$, $\boldsymbol{v}^{\mathrm{O}}$ and $\boldsymbol{v}^{\mathrm{I}}$ are represented by the black circles in the compact graph, which are referred to as variable nodes, and the trellises corresponding to $\mathcal{C}^{\mathrm{O}}$ and $\mathcal{C}^{\mathrm{I}}$ are represented by squares, referred to as constraint nodes. Each constraint node is labeled with the corresponding trellis length. The sequences $\boldsymbol{u}$ and $\boldsymbol{v}^{\mathrm{O}}$ are connected to the outer constraint node T^{O}. Similarly, the sequence $(\boldsymbol{u}, \boldsymbol{v}^{\mathrm{O}})$, permuted through a permuter Π, and the sequence $\boldsymbol{v}^{\mathrm{I}}$ are connected to the inner constraint node T^{I}. The sequence $(\boldsymbol{u}, \boldsymbol{v}^{\mathrm{O}})$ in the compact graph is obtained by a multiplexer, which is indicated by the rectangle. The permuter

$\boldsymbol{\Pi}$ is shown as a line that crosses the edge connecting the inner constraint node with the multiplexer.

Figure 1c shows the compact graph of the spatially coupled SCC (SC-SCC) ensemble with coupling memory $m = 1$ at time t. Consider a collection of SCC compact graphs at times $t = 1, \ldots, L$, where L denotes the coupling length. Denote by $\boldsymbol{s}_\mathsf{t}$ the sequence $(\boldsymbol{u}_\mathsf{t}, \boldsymbol{v}_\mathsf{t}^\mathsf{O})$ and by $\widetilde{\boldsymbol{s}}_\mathsf{t}$ the reordered sequence, reordered by permutation $\boldsymbol{\Pi}_1$. The SC-SCC ensemble is constructed by dividing the sequence $\widetilde{\boldsymbol{s}}_\mathsf{t}$ into two sub-sequences, denoted as $\widetilde{\boldsymbol{s}}_{\mathsf{t},\mathsf{k}}$ for $k = 0, 1$, and spreading them over times t and $t+1$. The sequence $(\widetilde{\boldsymbol{s}}_{\mathsf{t},0}, \widetilde{\boldsymbol{s}}_{\mathsf{t}-1,1})$ at the input of T_t^I is permuted by permuter $\boldsymbol{\Pi}_2$ before producing the parity sequence $\boldsymbol{v}_\mathsf{t}^\mathrm{I}$. The information bits at time $t \leq 0$ are initialized to zero.

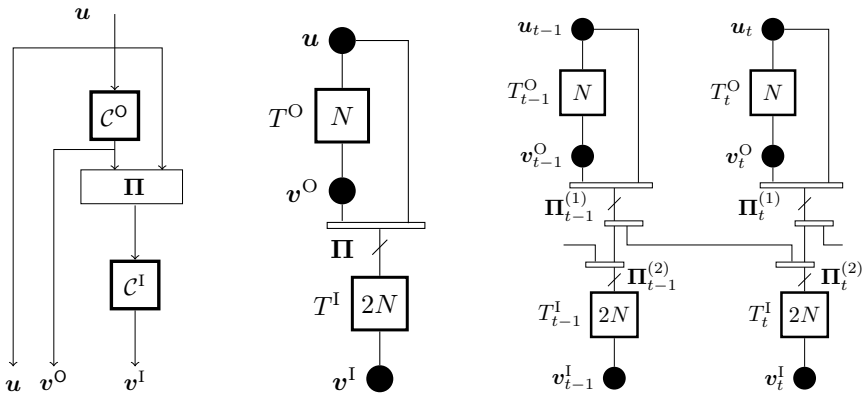


Figure 1: (**a**) Encoder block diagram of SCC, (**b**) Compact graph representation of SCC Encoder, (**c**) SC-SCC.

2.2 Representation of Spatially Coupled Turbo-Like Codes

We introduce a base-matrix representation corresponding to the compact graphs of TC ensembles, similar to that of protograph LDPC codes. Starting with the SCC ensemble in Figure 1, we define for each ensemble a connection matrix $\boldsymbol{P}$, which is the bi-adjacency matrix of the lifted compact graph. From $\boldsymbol{P}$, the base matrices of the coupled and uncoupled ensembles can be identified.

The outer constraint node T^O of the SCC in Figure 1b is connected to $\boldsymbol{v}^\mathrm{O}$ and $\boldsymbol{u}$, both representing N bits, as indicated by the label of the constraint node. These connections are represented in the first row of a connection matrix $\boldsymbol{P}_\mathsf{SCC}$ by the two $N \times N$ identity matrices $\boldsymbol{I}_N$. The edges from $\boldsymbol{v}^\mathrm{O}$ and $\boldsymbol{u}$ are first merged and then connected to the inner constraint node T^I, after permutation by $\boldsymbol{\Pi}$. This is represented by the matrix $\boldsymbol{P}_{2N} = \boldsymbol{\Pi}$ of size $2N \times 2N$ in the second row of $\boldsymbol{P}_\mathsf{SCC}$. Similarly, the connections along the edge of variable node $\boldsymbol{v}^\mathrm{I}$—representing $2N$ bits—to T^I is captured by the identity matrix $\boldsymbol{I}_{2N}$.

$$\boldsymbol{P}_\mathsf{SCC} = \begin{array}{c} \text{node} \\ T^\mathrm{O} \\ T^\mathrm{I} \end{array} \begin{array}{c} \begin{array}{ccc} \boldsymbol{u} & \boldsymbol{v}^\mathrm{O} & \boldsymbol{v}^\mathrm{I} \end{array} \\ \left[\begin{array}{cc|c} \boldsymbol{I}_N & \boldsymbol{I}_N & \boldsymbol{0} \\ \hline \multicolumn{2}{c|}{\boldsymbol{P}_{2N}} & \boldsymbol{I}_{2N} \end{array}\right] \end{array}. \tag{1}$$

The connection matrix representation allows to describe the ensemble in

terms of a base matrix, analogous to the base matrix of a protograph-based LDPC code. In the base matrix, the individual permutation matrices in the connection matrix are replaced by a 1 and the zero matrices by a 0, resulting in

$$\boldsymbol{B} = \begin{bmatrix} 1 & \circ & 1 & 0 \\ \circ & 1 & \circ & 1 \end{bmatrix} .$$

The base matrix represents an ensemble of codes, defined by the set of possible permutation matrices that can be used in the lifting procedure. In order to lift the base matrix $\boldsymbol{B}$ to a particular connection matrix $\boldsymbol{P}_{\mathsf{SCC}}$, each 1 is replaced by a permutation matrix and each 0 by an all-zero matrix. Note that the matrices have different sizes, which can be identified from the connection matrix (1). The entries denoted by $\circ$ are placeholders that are required because of the merging of two edges of width N into one edge of width $2N$ in the compact graph. To make our notation consistent, in the lifting procedure we replace each $\circ$ in the base matrix by an empty matrix with column dimension zero. There is a one-to-one correspondence between the base matrix and the compact graph representation provided that the lengths of the component encoder trellises are known.

A coupled ensemble can be obtained by partitioning $\boldsymbol{B}$ into submatrices $\boldsymbol{B}_i$ such that $\boldsymbol{B} = \sum_{i=0}^{m} \boldsymbol{B}_i$ [12]. For the ensemble in Figure 1c, we get

$$\boldsymbol{B}_0 = \begin{bmatrix} 1 & \circ & 1 & 0 \\ \circ & \frac{1}{m+1} & \circ & 1 \end{bmatrix}, \quad \boldsymbol{B}_{i>0} = \begin{bmatrix} 0 & \circ & 0 & 0 \\ \circ & \frac{1}{m+1} & \circ & 0 \end{bmatrix} .$$

The fraction $\frac{1}{m+1}$ in $\boldsymbol{B_i}$ indicates that $\frac{1}{m+1} \cdot 2N$ bits out of the $2N$ bits represented by the variable nodes of the SC-SCC graph at time t are connected with the trellises at time $t+i$ in a randomized way.

Following the same procedure, the connection matrix of uncoupled BCCs and the base matrices of SC-BCCs are obtained as

The fraction in is taken to represent that $\left(1 - \frac{1}{m+1}\right) \cdot 2N$ number of variable nodes of an SC-SCC graph at time t are connected with the trellises of the neighboring graphs in a SC chain in a randomized way. Likewise, the connection matrix of uncoupled BCC and sub-matrices of SC-BCC are described as

The compact graph representation and encoding method of the considered TCs are provided in [6] and [7]. For these ensembles we provide the base matrices as follows

$$\boldsymbol{P}_{\mathsf{BCC}} = \begin{array}{c} \text{node} \\ T^{\mathrm{U}} \\ T^{\mathrm{L}} \end{array} \begin{array}{c} \begin{array}{ccc} \boldsymbol{u} & \boldsymbol{v}^{\mathrm{U}} & \boldsymbol{v}^{\mathrm{L}} \end{array} \\ \begin{bmatrix} \boldsymbol{I}_N & \boldsymbol{I}_N & \boldsymbol{P}_N^{\mathsf{U}} \\ \boldsymbol{P}_N & \boldsymbol{P}_N^{\mathsf{L}} & \boldsymbol{I}_N \end{bmatrix} \end{array}, \quad \boldsymbol{B}_0 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{B}_{i>0} = \begin{bmatrix} 0 & 0 & \frac{1}{m} \\ 0 & \frac{1}{m} & 0 \end{bmatrix} .$$

and for PCCs as

$$\boldsymbol{P}_{\mathsf{PCC}} = \begin{array}{c} \text{node} \\ T^{\mathrm{U}} \\ T^{\mathrm{L}} \end{array} \begin{array}{c} \begin{array}{ccc} \boldsymbol{u} & \boldsymbol{v}^{\mathrm{U}} & \boldsymbol{v}^{\mathrm{L}} \end{array} \\ \begin{bmatrix} \boldsymbol{I}_N & \boldsymbol{I}_N & \boldsymbol{0} \\ \boldsymbol{P}_N & \boldsymbol{0} & \boldsymbol{I}_N \end{bmatrix} \end{array}, \quad \boldsymbol{B}_0 = \begin{bmatrix} \frac{1}{m} & 1 & 0 \\ \frac{1}{m} & 0 & 1 \end{bmatrix}, \quad \boldsymbol{B}_{i>0} = \begin{bmatrix} \frac{1}{m} & 0 & 0 \\ \frac{1}{m} & 0 & 0 \end{bmatrix} .$$

The connection matrix of the uncoupled HCC ensemble is

$$\boldsymbol{P}_{\mathsf{HCC}} = \begin{array}{c} \text{node} \\ T^{\mathrm{U}} \\ T^{\mathrm{L}} \\ T^{\mathrm{I}} \end{array} \begin{array}{c} \begin{array}{cccc} \boldsymbol{u} & \boldsymbol{v}^{\mathrm{U}} & \boldsymbol{v}^{\mathrm{L}} & \boldsymbol{v}^{\mathrm{I}} \end{array} \\ \left[\begin{array}{c|c|c|c} \boldsymbol{I}_N & \boldsymbol{I}_N & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{P}_N & \boldsymbol{0} & \boldsymbol{I}_N & \boldsymbol{0} \\ \hline \boldsymbol{0} & \multicolumn{2}{c|}{\boldsymbol{P}_{2N}^{\mathrm{I}}} & \boldsymbol{I}_{2N} \end{array}\right] \end{array} .$$

In [7], two spatially coupled ensembles of HCCs, referred to as Type-I SC-HCCs and Type-II SC-HCCs, were introduced. For Type-I, the base matrices are

$$\boldsymbol{B}_0 = \begin{bmatrix} 1 & 1 & \circ & 0 & 0 \\ 1 & 0 & \circ & 1 & 0 \\ 0 & \circ & \frac{1}{m+1} & \circ & 1 \end{bmatrix}, \quad \boldsymbol{B}_{i>0} = \begin{bmatrix} 0 & 0 & \circ & 0 & 0 \\ 0 & 0 & \circ & 0 & 0 \\ 0 & \circ & \frac{1}{m+1} & \circ & 0 \end{bmatrix},$$

whereas for Type-II are obtained as

$$\boldsymbol{B}_0 = \begin{bmatrix} \frac{1}{m+1} & 1 & \circ & 0 & 0 \\ \frac{1}{m+1} & 0 & \circ & 1 & 0 \\ 0 & \circ & \frac{1}{m+1} & \circ & 1 \end{bmatrix}, \quad \boldsymbol{B}_{i>0} = \begin{bmatrix} \frac{1}{m+1} & 0 & \circ & 0 & 0 \\ \frac{1}{m+1} & 0 & \circ & 0 & 0 \\ 0 & \circ & \frac{1}{m+1} & \circ & 0 \end{bmatrix}.$$

Throughout this paper we consider convolutional encoders with four-state trellises in all threshold computations and finite length simulations. In particular, rate-2/3 encoders with generator matrix

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 1/7 \\ 0 & 1 & 5/7 \end{bmatrix}$$

are used for BCCs and rate-1/2 encoders with generator matrix $\boldsymbol{G} = [1\ 5/7]$ are used for all other ensembles.

2.3 Sliding Window Decoding

In this work, for the threshold analysis we assume SC-TCs of coupling length $L = \infty$ and coupling memory $m = 1$ under sliding window decoding with on-demand symbol node updating schedule [9] of window size W. In this schedule, the constraint nodes within the window are updated sequentially by receiving the most recent updated messages from the neighboring nodes. Note that a larger window size is required for the computation of the thresholds of coupled ensembles with larger m, as the window size needs to be large enough so that the decoding wave is formed. All our threshold computations are carried out by considering $W = 20$, which is observed to be enough for a reliable estimate of the decoding thresholds for the considered ensembles with $m = \{1, 3\}$, yet allowing an efficient computation. In our finite length simulations we use $W = 8$.

3 Threshold Computation via Monte Carlo Density Evolution

The BP thresholds can be computed by performing DE. For codes for which the transfer functions of the component codes are not available, MC-DE can be used. In this section, we describe two MC-DE methods and discuss their advantages and shortcomings. The described MC-DE methods are computationally demanding. We hence also discuss an efficient method that provides an approximation of the BP threshold, and compare the thresholds obtained applying the three approaches for uncoupled SCCs.

3.1 Monte-Carlo Density Evolution

MC-DE comprises three key steps [8], which are performed a number of iterations:

1. Variable node update: A variable node generates a sequence of extrinsic log-likelihood ratios (LLRs) by properly combining the sequence of Gaussian-distributed channel LLRs and the sequences of extrinsic LLRs—generated according an appropriate distribution—received from the neighboring constraint nodes. In particular, the output LLRs are obtained as the sum of the channel LLR and the LLRs from neighboring constraint nodes.

2. Constraint node update: A constraint node performs BCJR decoding on the sequences of extrinsic LLRs—used as a-priori information—received from the neighboring variable nodes. The BCJR decoder generates sequences of extrinsic LLRs that are passed to the corresponding neighboring variable nodes.

3. Density estimation and re-sampling: A sequence of channel LLRs, $\boldsymbol{L}_{\mathsf{ch}}$, and the sequences of extrinsic LLRs, $\boldsymbol{L}_{\mathsf{ext}}$ (one for each code sequence), are created from the corresponding probability densities $f(L_{\mathsf{ch}})$ and $f(L_{\mathsf{ext}})$. These sequences are used in the next MC-DE iteration.

MC-DE tracks the evolution of $f(L_{\mathsf{ext}})$ through the iterative decoding process.

3.2 Monte-Carlo Density Evolution with Gaussian Approximation

In MC-DE with Gaussian approximation (MC-DE-GA), the densities $f(L_{\mathsf{ext}})$ are approximated by a Gaussian distribution, which can be characterized by its mean m_G and standard deviation σ. Since the extrinsic LLRs are symmetric and consistent, the Gaussian distribution is completely characterized by the single parameter σ, which is related to the mean m_G as

$$m_{\mathrm{G}} = \frac{\sigma^2}{2}. \tag{2}$$

The standard deviation σ is computed from the mutual information I_{E} between an extrinsic LLR sequence $\boldsymbol{L}_{\mathsf{ext}}$ and the corresponding binary code sequence [4, 5],

$$\sigma = C_{\mathrm{G}}^{-1}\left(I_{\mathrm{E}}\right) \ . \tag{3}$$

Here $C_{\mathrm{G}}(\sigma)$ denotes the AWGN channel capacity for a given channel parameter σ, which can be computed efficiently using the following series expansion [13, Chapter 4]:

$$C_{\mathrm{G}}(\sigma) = 1 + \frac{1}{\ln 2}\cdot \left(\left(\frac{2}{\sigma^2} - 1\right) Q\left(\frac{1}{\sigma}\right) - \sqrt{\frac{2}{\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}} + \sum_{i=1}^{\infty} \frac{(-1)^i}{i(i+1)} e^{\frac{2i(i+1)}{\sigma^2}} Q\left(\frac{1+2i}{\sigma}\right)\right) \tag{4}$$

with $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{y^2}{2}} dy$. The mutual information I_{E} is computed from [5]

$$I_{\mathrm{E}} \approx 1 - \frac{1}{N'} \sum_{n=1}^{N'} \log_2(1 + e^{-L_{\mathsf{ext,n}}}) \ , \tag{5}$$

where $L_{\mathsf{ext,n}}, n = 1, \ldots, N'$, are the elements of $\boldsymbol{L}_{\mathsf{ext}}$.

Note that MC-DE-GA is equivalent to an EXIT function analysis. While a threshold computation via the Gaussian assumption becomes highly efficient for LDPC codes, MC-DE-GA has only a minor computational advantage for TCs. Furthermore, for several ensembles, in particular multi-edge type ensembles such as BCCs, SCCs, HCCs, and SC-TCs, the true distributions of the messages may significantly deviate from a Gaussian distribution, leading to inexact decoding thresholds.

3.3 MC-DE with Histogram

A more accurate BP threshold can be obtained by estimating the true densities, which can be performed by means of histograms. We refer to this method as MC-DE-H, which we describe in detail in the following. For ease of notation, let L be the random variable corresponding to an extrinsic LLR. From the consistency property of the LLRs, we have that $f_L(l) = e^l \cdot f_L(-l)$ [5]. The consistency and symmetry properties of L allow us to determine $f_L(l)$ from the distribution of $|L|$ from [13]

$$f_L(l) = \mathbb{I}_{\{l \geq 0\}} \frac{1}{1 + e^{-l}} f_{|L|}(l) + \mathbb{I}_{\{l \leq 0\}} \frac{e^l}{1 + e^l} f_{|L|}(-l) \ . \tag{6}$$

Taking advantage of this symmetry drastically improves the speed of converge of this method.

Without loss of generality, we consider the transmission of the all-zero codeword and approximate the density $f_L(l)$ in (6) with a histogram of M bins and obtain an approximated probability mass function $\hat{P}(l)$ (For the threshold calculations within this paper we use a fixed number of $M = 2001$ bins, divided uniformly between $-L_{\max}$ and $+L_{\max}$, where $L_{\max} = \max_n |L_n|$ denotes the maximum magnitude among the elements L_n of the measured sequence. An odd value of M is recommended to represent erasures accurately. The length of the sequence is chosen adaptively to achieve a desired accuracy). The bit error rate (BER) can be computed from $\hat{P}(l)$ as

$$BER \approx \sum_{\{l<0\}} \hat{P}(l) \ . \tag{7}$$

We denote by $\hat{f}_L(l)$ the approximation of $f_L(l)$. A sequence of extrinsic LLRs distributed according to $\hat{f}_L(l)$ can be obtained from $\hat{P}(l)$ by using the probability integral transform. The probability integral transform method states that for a uniformly distributed random variable $U \in [0, 1]$, and a strictly increasing cumulative distribution function $\hat{F}_L(l)$, we have $U = \hat{F}_L(l) \approx \hat{P}(L \leq l)$. Samples are generated from $\hat{F}_L(l)$ by applying the inversion $\hat{F}_L^{-1}(U)$.

3.4 Erasure Channel Prediction

Since MC-DE is time consuming, we are interested in exploring some faster alternatives to predict the BP thresholds of TCs over the AWGN channel. In [3], the erasure channel prediction (ECP) method was proposed to efficiently predict BP thresholds of codes over the AWGN channel from their corresponding BEC thresholds ϵ^*. For a given code, the AWGN channel BP threshold σ^* can be obtained from the corresponding ϵ^* as

$$\sigma^* \approx C_{\mathrm{G}}^{-1}\left(C_{\mathrm{E}}(\varepsilon^*)\right) = C_{\mathrm{G}}^{-1}\left(1-\varepsilon^*\right) \ . \tag{8}$$

where $C_{\mathrm{E}}(\epsilon^*) = 1 - \epsilon^*$ is the capacity of the BEC.

3.5 Discussion

In Table 1 we give the BP thresholds of the uncoupled SCCs computed via MC-DE-GA, MC-DE-H, and ECP, for several code rates. We observe that both MC-DE-GA and MC-DE-H yield similar thresholds. However, we remark that the computational complexity of MC-DE-GA and MC-DE-H is similar, as it is primarily dominated by that of the BCJR decoder. Hence, one may resort to MC-DE-H, which does not rely on a Gaussian assumption and gives a more accurate estimation of the threshold provided that the quantization resolution is chosen sufficiently high. The thresholds predicted by the ECP method differ noticeably from those predicted by the MC-DE methods.

Thresholds	**Rate**			
E_b/N_0 (dB)	1/4	1/3	1/2	2/3
MC-DE-GA	0.11	0.50	1.46	2.95
MC-DE-H	0.12	0.51	1.50	3.05
ECP	0.37	0.76	1.74	3.25

Table 1: BP thresholds of uncoupled serially concatenated codes (SCCs) obtained by Monte-Carlo density evolution with Gaussian approximation (MC-DE-GA), Monte-Carlo density evolution with histogram (MC-DE-H), and erasure channel prediction (ECP).

4 Efficient AWGN Channel Threshold Predictions of Randomly Punctured TCs

Following the ideas in [11], efficient methods for predicting the thresholds of randomly punctured BCCs were investigated in [8], namely the θ_{E} prediction, the θ_{G} prediction, and the mixed prediction (MP) method. In this section, we re-visit these methods by analyzing the MC-DE-H and predicted thresholds of the SCC ensemble as an example.

4.1 θ_{E}-Predictions

Consider a code ensemble of rate $R(\alpha)$ obtained by randomly puncturing a mother code of rate $R = R/(1-\alpha)$, where $0 \leq \alpha < 1$ is the puncturing fraction,

i.e., the fraction of bits that are punctured. For the BEC, the BP threshold of the punctured code ensemble, $\epsilon^*(\alpha)$, can be obtained as

$$\epsilon^*(\alpha) = 1 - \theta_{\mathrm{E}} R(\alpha) \ , \tag{9}$$

where

$$\theta_{\mathrm{E}} = \frac{1-\epsilon^*}{R} \ , \tag{10}$$

with ϵ^* being the BP threshold of the mother code.

The BP threshold $\sigma^*(\alpha)$ of a randomly punctured ensemble over the AWGN channel can be predicted from the BEC threshold of the corresponding mother code by combining (9) with the ECP in (8) to

$$h_{\mathrm{G}}(\sigma^*(\alpha)) \approx h_{\mathrm{E}}(\varepsilon^*(\alpha)) = \epsilon^*(\alpha) = 1 - \theta_{\mathrm{E}} R(\alpha). \tag{11}$$

Here, $h_{\mathrm{G}}(\sigma) = 1 - C_{\mathrm{G}}(\sigma)$ and $h_{\mathrm{E}}(\varepsilon) = 1 - C_{\mathrm{E}}(\varepsilon) = \varepsilon$ denote the conditional entropy of the AWGN channel and the BEC, respectively. We refer to this method as θ_{E}-prediction.

The θ_{E}-predictions and the MC-DE-H thresholds of SCCs and SC-SCCs with $m = 1$ and different code rates are shown in Table 2, where ϵ and ϵ_{SC} denote the BEC thresholds of SCCs and SC-SCCs with $m = 1$, respectively. It is observed that with coupling, the thresholds computed using the θ_{E}-prediction are similar to those obtained via MC-DE-H, i.e., the θ_{E}-prediction yields accurate thresholds. For SCCs, however, the thresholds differ noticeably. We remark that punctured bits can be equivalently seen as erasures. Hence, the behavior of the ensembles over the AWGN channel becomes closer to their behavior over the BEC for increasing puncturing fraction. This explains that the relative difference between the θ_{E}-prediction and MC-DE-H thresholds is larger for lower rates. We conjecture that the accurateness of the θ_{E}-prediction for the SC-SCC ensemble is due to the universality of this ensemble. Indeed, it was shown in [6] that for large-enough coupling memory, SC-SCCs approach capacity.

		SCC			**SC-SCC**	
Rate	ϵ	θ_{E} (E_b/N_0)	MC-DE-H	ϵ_{SC}	θ_{E} (E_b/N_0)	MC-DE-H
1/4	0.6896	0.37	0.12	0.7379	-0.54	-0.59
1/3	0.5861	0.76	0.51	0.6505	-0.22	-0.29
1/2	0.3792	1.74	1.50	0.4758	0.51	0.43
2/3	0.1723	3.25	3.05	0.3011	1.48	1.39
3/4	0.0688	4.70	-	0.2137	2.13	2.05

Table 2: Erasure-, θ_{E} predicted- and MC-DE-H thresholds of SCC ensembles.

4.2 θ_{G}-Predictions

The gap between the MC-DE-H threshold and the θ_{E}-prediction for low rates can be reduced by using the θ_{G}-prediction [8]. This prediction method uses the AWGN channel threshold of the mother code to determine the BP threshold

of the punctured code. Using the θ_{G}-prediction, the AWGN channel threshold $h_{\mathrm{G}}(\sigma^*)$ is obtained as

$$h_{\mathrm{G}}(\sigma^*(\alpha)) \approx 1 - \theta_{\mathrm{G}} R(\alpha) \ , \tag{12}$$

where

$$\theta_{\mathrm{G}} = \frac{1 - h_{\mathrm{G}}(\sigma^*)}{R} \ . \tag{13}$$

The θ_{G}-predictions for SCCs are shown in Table 3. For low rates, the θ_{G}-predictions are close to the MC-DE-H thresholds. However, the θ_{G}-prediction fails to accurately predict the thresholds for higher rates, and a gap to the MC-DE-H thresholds is observed.

Thresholds	**Rate**				
E_b/N_0 (dB)	1/4	1/3	1/2	2/3	3/4
θ_{E} Predicted	0.37	0.76	1.74	3.25	4.70
θ_{G} Predicted	0.12	0.50	1.40	2.72	3.82
MC-DE-H	0.12	0.51	1.50	3.05	-

Table 3: Comparison of θ_{G} and θ_{E} predicted thresholds of SCCs.

4.3 Mixed Predictions

A mixed prediction (MP) method is proposed in [8] to overcome the discrepancies observed in the θ_{E} and the θ_{G} predictions. The idea of the MP stems in [11], where the AWGN channel thresholds of randomly punctured LDPC codes were observed to lie on a straight line in the entropy perspective. The MP method uses both θ_{E} and $h_{\mathrm{G}}(\sigma^*)$ in accurately predicting the thresholds of randomly punctured LDPC codes at all rates. For randomly punctured TCs, as shown for randomly punctured SCCs in Figure 2a, we can observe that the AWGN channel thresholds tend to follow a straight line as well. From θ_{E} and $h_{\mathrm{G}}(\sigma^*)$, the predicted thresholds of the punctured TCs via the MP method [8] are obtained by using

$$h_{\mathrm{G}}(\sigma_{BP}(\alpha)) \approx h_{\mathrm{G}}(\sigma^*) - \theta_{\mathrm{MP}} \left(R(\alpha) - R \right) , \tag{14}$$

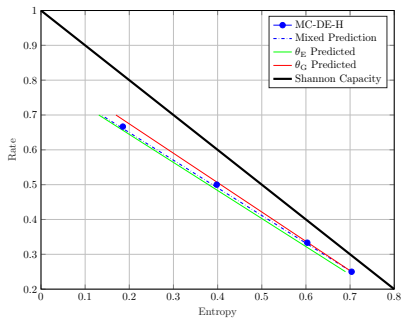
where θ_{MP} is

$$\theta_{\mathrm{MP}} = \frac{\theta_{\mathrm{E}} + h_{\mathrm{G}}(\sigma^*) - 1}{1 - R} \ . \tag{15}$$

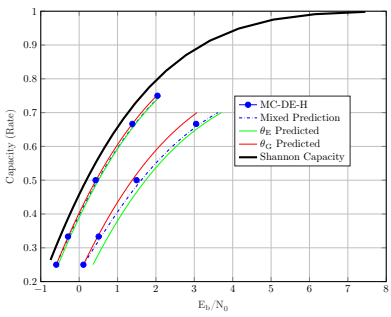
The MP thresholds are shown as a dashed line in Figure 2a. It is observed that the SCC MP thresholds deviate slightly from the MC-DE-H thresholds at medium rates, unlike the more accurate MP thresholds for the LDPC codes in [11]. In fact, even for LDPC codes it is still an open problem to prove the conjecture that AWGN channel thresholds follow a straight line with random puncturing. For TC ensembles with more complicated component codes this is even harder to prove and may be wrong. On the other hand, the deviations we observe are small enough to use the MP thresholds as an efficient approximation.

The MC-DE-H and predicted thresholds of the uncoupled and coupled SCCs over the AWGN channel are shown in Figure 2b, where the MP method is

observed to be a more suitable representative of the MC-DE-H thresholds for all the considered rates.



(a) SCC Entropy vs Capacity



(b) SCC and $m = 1$ SC-SCC E_b/N_0 vs Capacity

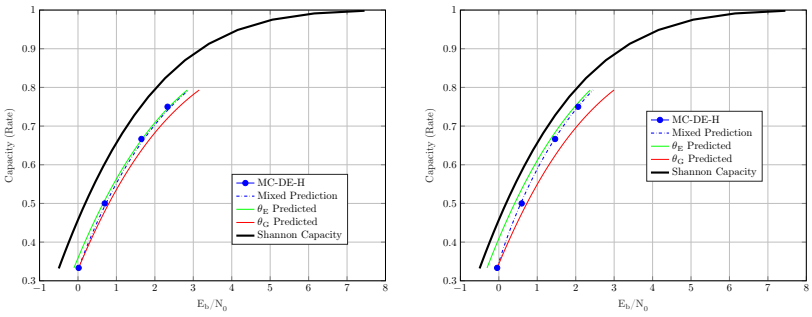
Figure 2: AWGN channel thresholds of uncoupled and coupled SCCs.

5 Threshold Comparison for Different TC Ensembles

Table 4 presents the MC-DE-H and θ_E predicted thresholds of the coupled TCs. Unlike uncoupled TCs, the predictions are observed to be relatively close to the MC-DE-H thresholds. For this reason, we do not list the θ_G- or MP thresholds of SC-TCs in the table. For the uncoupled ensembles, however, the MP method provides better predicted thresholds than the θ_E method. We further observe that, in general, the θ_G predictions are closer to the Shannon capacity than the θ_E predictions. The exception to this are PCCs, where θ_E predictions are closer to the Shannon capacity than the θ_G predictions, and the gap increases with coupling, as shown in Figure 3.

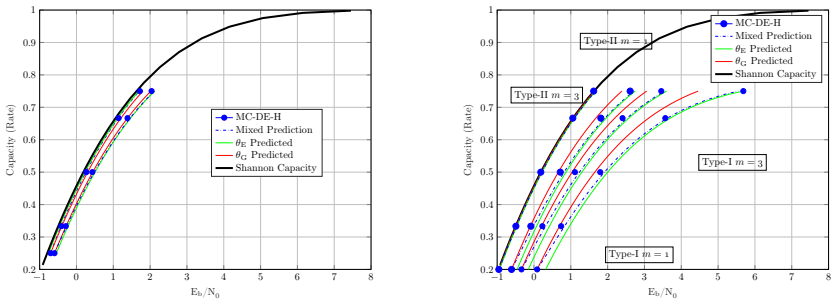
Ensemble	m	**Thresholds**			**Rate**			
		E_b/N_0 (dB)	1/5	1/4	1/3	1/2	2/3	3/4
Shannon Capacity			-0.9637	-0.7942	-0.4952	0.1872	1.0597	1.6262
SC-SCC	1	θ_{E} Predicted	-	-0.54	-0.22	0.51	1.48	2.13
SC-SCC	1	MC-DE-H	-	-0.59	-0.29	0.43	1.39	2.05
SC-SCC	3	θ_{E} Predicted	-	-0.75	-0.45	0.24	1.12	1.70
SC-SCC	3	MC-DE-H	-	-0.70	-0.41	0.27	1.15	1.73
SC-PCC	1	θ_{E} Predicted	-	-	-0.30	0.42	1.35	1.98
SC-PCC	1	MC-DE-H	-	-	-0.04	0.60	1.47	2.07
SC-HCC-II type-II	1	θ_{E} Predicted	-0.45	-	0.08	0.87	1.97	2.75
SC-HCC-II type-II	1	MC-DE-H	-0.60	-	-0.08	0.72	1.82	2.62
SC-HCC-II type-II	3	θ_{E} Predicted	-0.93	-	-0.46	0.23	1.11	1.68
SC-HCC-II type-II	3	MC-DE-H	-0.95	-	-0.49	0.19	1.06	1.63
SC-BCC	1	θ_{E} Predicted	-	-	-0.39	0.31	1.21	1.81
SC-BCC	1	MC-DE-H			-0.39	0.30	1.19	1.78
SC-BCC	3	θ_{E} Predicted	-	-	-0.45	0.24	1.12	1.70
SC-BCC	3	MC-DE-H			-0.43	0.25	1.13	1.70

Table 4: θ_{E} predicted and MC-DE-H thresholds of coupled turbo-like codes (TCs).



(a) E_b/N_0 vs Rate of PCC (b) E_b/N_0 vs Rate of $m = 1$ SC-PCC

Figure 3: Thresholds of PCCs over the AWGN channel

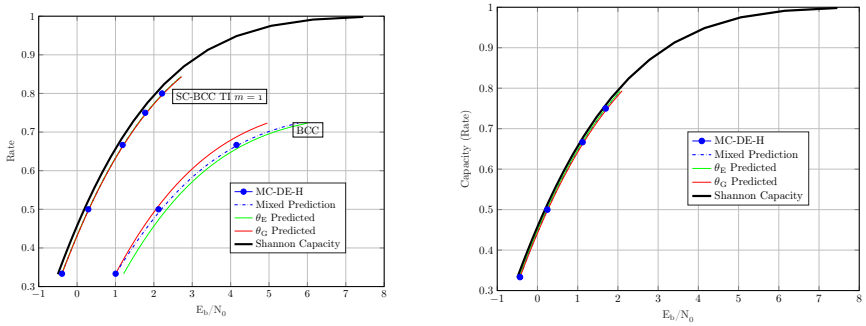


(a) E_b/N_0 vs Rate of SC-SCCs with $m = 1$ and $m = 3$ (b) E_b/N_0 vs Rate of type 1 and type 2 SC-HCCs-II with $m = 1$ and $m = 3$

Figure 4: Thresholds of SC-SCCs and SC-HCCs over the AWGN channel

From the thresholds of different TC ensembles, we also observe that the similarity of the MC-DE-H and the predicted thresholds depends on an ensemble's strength. For stronger TCs, the similarity between the predicted and the MC-DE-H thresholds is clearly observed, as shown for $m = \{1, 3\}$ SC-SCCs in Figure 4a, $m = \{1, 3\}$ SC-BCCs in Figure 5, and $m = 3$ SC-HCCs-II type-II in Figure 4b. We further observe that the predicted and the MC-DE-H thresholds are not alike 1) for PCCs in Figure 3, as these have BP thresholds close to the MAP thresholds but relatively poor MAP thresholds 2) for other uncoupled TCs in general and SC-HCCs-II type-I in Figure 4b, as these have strong MAP but poor BP thresholds and 3) for the SC-HCCs-II type-II with $m = 1$ in Figure 4b, which have a good MAP threshold but require a larger coupling memory.

SC-HCCs-II offer an interesting insight regarding the similarity of predicted and MC-DE-H thresholds of an ensemble. HCCs-II have the strongest MAP thresholds on the BEC among all considered TC ensembles, and we expect the threshold predictions to show strong similarity with spatial coupling. From the thresholds of the SC-HCCs-II in Figure 4b, however, we observe that this strong similarity is only visible for the SC-HCC-II type-II ensemble with $m = 3$. The SC-HCC-II type-I ensemble, which uses a different type of coupling than the type-II, shows a weaker BP performance and lower similarity than the type-II, even at a larger coupling memory $m = 3$. This suggests that in addition to



(a) E_b/N_0 vs Rate of BCCs and SC-BCCs with $m = 1$ (b) E_b/N_0 vs Rate of SC-BCCs with $m = 3$

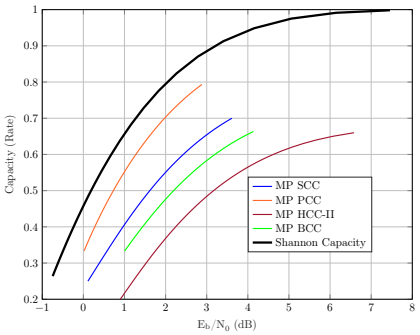
Figure 5: Thresholds of BCCs over the AWGN channel

a strong MAP threshold, the capability of an ensemble to exhibit very similar predicted and MC-DE-H thresholds is also linked to its ability of achieving threshold saturation [14]. For those SC-TCs that demonstrate threshold saturation, we have additionally computed the binary symmetric channel (BSC) thresholds and compare the entropy of the thresholds for the BSC, BEC and AWGN channel in Table 5. A similar entropy h at the thresholds of the selected SC-TCs over all three channels confirms our conjecture that a strong similarity between predicted and MC-DE-H thresholds of an ensemble is associated with its capability of achieving threshold saturation.

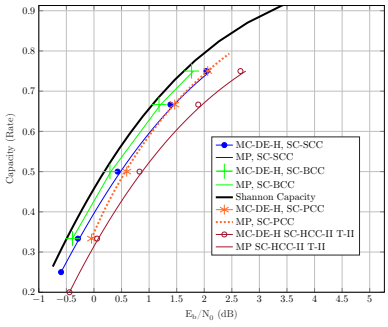
Ensemble	Rate	h_{BEC}	ϵ_{BSC}	h_{BSC}	h_{AWGN}
SC-BCC $m = 3$	$1/3$	0.6644	0.1718	0.6618	0.6630
SC-SCC $m = 3$	$1/4$	0.7483	0.2114	0.7442	0.7456
SC-SCC $m = 3$	$1/3$	0.6644	0.1708	0.6595	0.6616
SC-HCCII type-II $m = 3$	$1/5$	0.7990	0.2427	0.7995	0.7996
SC-HCCII type-II $m = 3$	$1/3$	0.6650	0.1738	0.6663	0.6664

Table 5: Entropy h of TCs over the binary erasure channel (BEC), binary symmetric channel (BSC) and additive white Gaussian noise (AWGN) channel.

In order to provide an overview over the different ensembles, the MC-DE-H and MP thresholds of uncoupled and coupled TCs with $m = 1$ are plotted in Figure 6. For coupled TCs with $m = 1$, BCCs perform best among the considered ensembles, whereas PCCs are the first among the uncoupled ensembles. Interestingly, SC-PCCs approach SC-HCCs at lower rates and SC-SCCs at larger rates. Note that the performance of SC-HCCs-II-TII can be improved by increasing the coupling memory and it is observed that they then outperform SC-BCCs with $m = \{1, 3\}$.



(a) E_b/N_0 vs Rate of uncoupled TCs



(b) E_b/N_0 vs Rate of SC-TCs with $m = 1$

Figure 6: Comparison of TC ensembles in terms of the AWGN channel Thresholds

The threshold observations are further validated by performing some finite length BER performance simulations for SC-HCC-II type-II codes with $m = 3$, and SC-SCC and SC-BCC codes with $m = 1$ for equal rate $R = 1/3$ and equal decoding latency. The SC-HCC-II type-II code with $m = 3$ is chosen because the $m = 1$ ensemble has a poor BP performance on both the BEC and the AWGN channel compared to SC-BCCs and SC-SCCs with $m = 1$. We use sliding window decoding with a window size $W = 8$, coupling length $L = 100$, and 20 decoding iterations at each window position. The input block length is $N = 16\,384$ for all ensembles, resulting in an overall structural decoding latency of $3\,N\,W = 393\,216$ code symbols. The BER performance results are shown in Figure 7. It is observed that the SC-HCC-II type-II code with $m = 3$ has the best performance followed by the SC-BCC code and SC-SCC code with $m = 1$ respectively. The simulations are observed to be consistent with the BP thresholds. However, the gain of the HCC ensemble in terms of the threshold is larger is than in terms of the simulated waterfall performance. This partially can be prescribed to the limited window size.

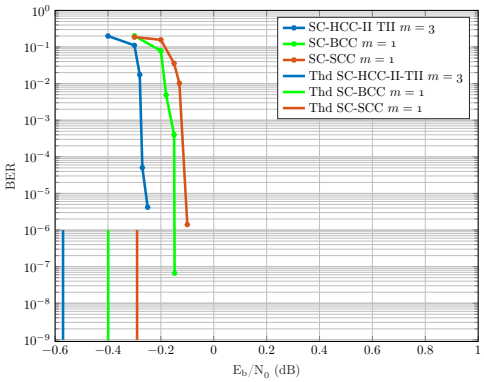


Figure 7: Finite block length performance of rate 1/3 ensembles with equal latency.

In Table 6, we list the BEC and the AWGN channel thresholds of the TCs along with the parameters θ_{E}, θ_{G} and θ_{MP}. By using these values together with the prediction methods described in Section 4, it is possible to immediately reproduce all the continuous threshold curves in Figures 2–6. The computation of MC-DE-H thresholds, which are shown as blue dots in these figures, is very time consuming and provided only for validation of the prediction methods. Consider the MP thresholds of rate 1/3 SCCs as an example to show how to calculate the predicted thresholds using Table 6. First, we obtain the noise threshold $\sigma^* = 1.3963$ of rate $R = 1/4$ SCCs from its MC-DE-H threshold $E_b/N_0(\mathsf{dB}) = 0.1109$. Next, we apply the entropy at the noise threshold $h_{\mathrm{G}}(1.3963) = 1 - C_{\mathrm{G}}(1.3963) = 0.7035$, and $\theta_{\mathrm{MP}} = 1.2601$ from the Table 6 to (14) for $R(\alpha = 1/4) = 1/3$. This gives us $h_{\mathrm{G}}(\sigma_{BP}(\alpha = 1/4)) = 0.5985$. From the AWGN channel capacity $C_{\mathrm{G}}(\sigma) = 1 - 0.5985 = 0.4015$, we obtain $\sigma = C_{\mathrm{G}}^{-1}(0.4015) = 1.1461$. Lastly, by using that $\sigma^2 = 1/\left(2 \cdot E_b/N_0 \cdot R(\alpha)\right)$ we obtain the MP for rate $R = 1/3$ SCCs in terms of $E_b/N_0(\mathsf{dB}) = 0.5761$.

Ensemble	Rate	MC-DE-H $E_b/N_0(dB)$	ϵ	**Parameters**		
				θ_{E}	θ_{G}	θ_{MP}
PCC	$1/3$	0.02	0.6428	1.0716	1.0953	1.0597
SC-PCC, $m = 1$	$1/3$	-0.04	0.6553	1.0341	1.0839	1.0092
SCC	$1/4$	0.12	0.6896	1.2416	1.1880	1.2595
SC-SCC, $m = 1$	$1/4$	-0.59	0.7379	1.0484	1.0398	1.0513
SC-SCC, $m = 3$	$1/4$	-0.70	0.7483	1.0068	1.0182	1.0030
HCC	$1/5$	0.90	0.7044	1.4780	1.4339	1.4890
SC-HCCII-TII, $m = 1$	$1/5$	-0.60	0.7790	1.1050	1.0748	1.1125
SC-HCCII-TII, $m = 3$	$1/5$	-0.95	0.7990	1.0050	1.0027	1.0056
BCC	$1/3$	1.01	0.5541	1.3377	1.2943	1.3594
SC-BCC, $m = 1$	$1/3$	-0.39	0.6609	1.0173	1.0190	1.0165
SC-BCC, $m = 3$	$1/3$	-0.43	0.6644	1.0068	1.0117	1.0043

Table 6: θ parameter of various prediction methods.

The prediction methods provide a convenient way of comparing the thresholds of mother code ensembles with different rates. A randomly punctured code ensemble is characterized by the parameter $\theta \geq 1$, where $\theta = 1$ corresponds to a capacity achieving ensemble [11]. An ensemble with a smaller θ_{E} and θ_{G} will outperform an ensemble with larger θ_{E} and θ_{G} at all achievable rates.

6 Conclusions

In this paper, we have performed a BP decoding threshold analysis of SC-TCs on the AWGN channel and demonstrated that the prediction methods presented in [8, 11] can be used to approximate the thresholds efficiently. The prediction methods approximate the AWGN channel thresholds of the considered ensembles in a computationally efficient manner by using their BEC thresholds. Conventionally, MC-DE or EXIT function analysis are applied to analyze thresholds of TCs over the AWGN channel. Although these methods can provide a very accurate estimate of the BP thresholds, they are computationally expensive, especially for spatially coupled ensembles. Our results show that the predicted thresholds are very close to the MC-DE thresholds for strong spatially coupled ensembles such as SC-SCCs, SC-BCCs and SC-HCCs-II. It is further conjectured that the similarity between the predictions and MC-DE is associated with the strength of an ensemble and its threshold saturation capability. For strong coupled ensembles, universality is observed from the entropy of their thresholds over the BEC, AWGN channel and BSC. For uncoupled ensembles with random puncturing, the predictions are improved with help of both the AWGN channel and BEC threshold of the mother code ensembles.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings of ICC '93 - IEEE International Conference on Communications*, vol. 2, pp. 1064–1070 vol.2, 1993.

[2] R. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[3] S.-Y. Chung, *On the construction of some capacity-approaching coding schemes*. PhD thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, Sept. 2000.

[4] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, 2001.

[5] J. Hagenauer, "The exit chart - introduction to extrinsic information transfer in iterative processing," in *2004 12th European Signal Processing Conference*, pp. 1541–1548, 2004.

[6] S. Moloudi, M. Lentmaier, and A. G. i Amat, "Spatially coupled turbo-like codes," *IEEE Trans. Inf. Theory*, vol. 63, pp. 6199–6215, Oct. 2017.

[7] S. Moloudi, M. Lentmaier, and A. G. i Amat, "Spatially coupled hybrid concatenated codes," in *Proc. 11th Int. ITG Conf. Systems, Commun. and Coding (SCC)*, (Hamburg, Germany), 2017.

[8] M. U. Farooq, S. Moloudi, and M. Lentmaier, "Thresholds of braided convolutional codes on the awgn channel," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1375–1379, 2018.

[9] M. Lentmaier, A. Sridharan, D.J. Costello, Jr., and K.Sh. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 56, pp. 5274–5289, Oct. 2010.

[10] S. Kudekar, T. Richardson, and R. L. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 7761–7813, 2013.

[11] D. G. M. Mitchell, M. Lentmaier, A. E. Pusane, and D. J. Costello, Jr., "Randomly punctured LDPC codes," *IEEE Journal on Selected Areas in Communications*, vol. 34, pp. 408–421, Feb. 2016.

[12] D. Mitchell, M. Lentmaier, and D. Costello, Jr., "Spatially coupled LDPC codes constructed from protographs," *IEEE Trans. Inf. Theory*, vol. 61, pp. 4866–4889, Sep. 2015.

[13] T. Richardson and R. Urbanke, *Modern Coding Theory*. USA: Cambridge University Press, 2008.

[14] S. Kudekar, T. Richardson, and R. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. Inf. Theory*, vol. 57, pp. 803 –834, Feb. 2011.

Paper III

Spatially-Coupled Serially Concatenated Codes with Periodic Convolutional Permutors

Spatially-coupled serially concatenated codes (SC-SCCs) are a class of turbo-like codes constructed by interconnecting a sequence of SCCs using a set of block permutors. At short block lengths, however, the bit-error-rate (BER) performance of SC-SCCs constructed by independent block permutors exhibits a high error floor. In this paper, we propose an alternative method for constructing SC-SCCs to mitigate this problem. Particularly, we use a family of periodically time-varying blockwise convolutional permutors with flexible block length. We derive these convolutional permutors from a block permutor of an optimized spread by applying an unwrapping procedure. We prove that for any chosen block length, the unwrapping procedure preserves the spread of the original block permutor. We further present an efficient implementation method for the blockwise convolutional permutor that derives the permutation indices directly from those of the underlying block permutor. Considering both s-random permutors and quadratic permutation polynomial (QPP) permutors, we perform BER simulations for SC-SCCs with decoding latencies 4096 and 16384. Numerical results show that SC-SCCs based on the proposed convolutional permutors have no visible error floor, which is especially notable at short block lengths.

1 Introduction

Spatially-coupled turbo-like codes [1] with simple 4-state and 8-state component encoders achieve close-to-capacity thresholds thanks to the threshold saturation phenomenon—the belief propagation threshold converges to the optimal maximum-a-posteriori (MAP) threshold. Among them, spatially-coupled serially concatenated codes (SC-SCCs) are particularly enticing since they can also achieve very low error floors due to the large minimum distance stemming from the serial concatenation.

In [1], SC-SCCs were analyzed in the asymptotic limit of infinitely large block length. Their performance in the error floor region was investigated in [2], where it was shown that SC-SCCs can achieve very low error floors by performing a weight enumerator analysis. Particularly, [2] derived conditions under which the spatial coupling either preserves or improves the minimum distance of the underlying code. A thorough evaluation of the performance of SC-SCCs for varying block length of the underlying SCCs and decoding window size—which determine the decoding latency—as well as coupling memory and number of iterations per decoded bit, was conducted in [3]. It was shown that for short block length and limited number of iterations per decoded bit, a relatively high error floor appears. This error floor is due to the use of individual permutors at each spatial position to spread the edges across spatial positions. Indeed, the original construction in [1] specifies two permutors at each spatial position. For short block length, i.e., short component permutors, a joint design is required to avoid the performance in the error floor being impaired by the poor code properties induced by the short permutors—which may entail a poor minimum distance—, as observed in [3]. Unfortunately, a joint design of the permutors is a formidable task.

In this paper, we address this shortcoming by proposing a different construction of the permutors which allows to realize low error floors even for short block length and moderate decoding latency. The key idea is to construct a periodic blockwise convolutional permutor that spans several spatial positions by unwrapping [4] a block permutor of size equal to the desired decoding latency. In other words, rather than jointly designing short permutors to realize a strong longer permutor, we directly design the longer permutor. More specifically, we design convolutional permutors from s-random and quadratic permutation polynomial (QPP) block permutors. Due to the unwrapping technique, the obtained permutors inherit the properties of the block permutor, resulting in a better weight enumerator of the SC-SCC compared to the case where component permutors are designed independently. We show via Monte-Carlo simulations that the proposed construction yields SC-SCCs with low error floor even when the underlying SCCs have an information block length as short as 64 bits.

2 Spatially-Coupled Serially Concatenated Codes

2.1 Coupling at the compact graph level

We briefly describe the SC-SCCs introduced in [1]. The compact graph of an SC-SCC of chain length L is constructed by placing L copies of a serially concatenated code (SCC) in L spatial positions in the set $\mathcal{L} = \{1, \ldots, L\}$ and inter-

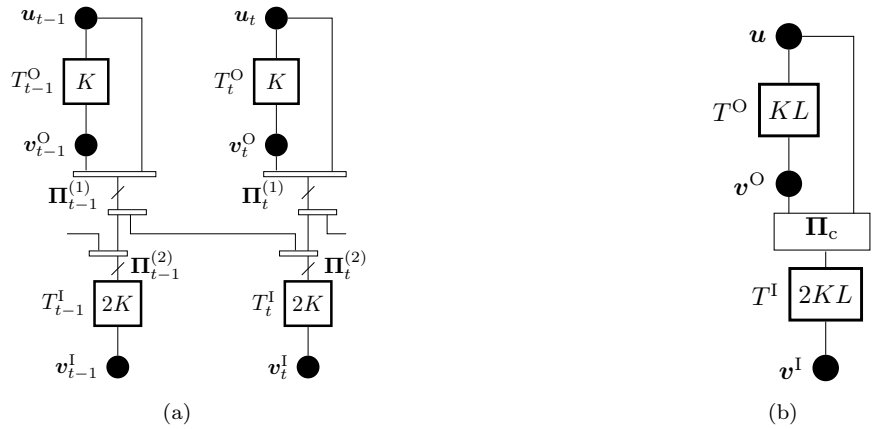


Figure 1: (a) Compact graph representation of the SC-SCC ensemble in [1] (b) a convolutional permutor perspective of SC-SCCs.

connecting them appropriately. Let $\boldsymbol{u}_t = (u_{t,1}, ..., u_{t,K})$ be the information sequence of length K at spatial position (time instance) t, and $\boldsymbol{v}^{\mathrm{O}}_t = (v^{\mathrm{O}}_{t,1}, ..., v^{\mathrm{O}}_{t,K})$ and $\boldsymbol{v}^{\mathrm{I}}_t = (v^{\mathrm{I}}_{t,1}, ..., v^{\mathrm{I}}_{t,2K})$ the parity sequence at the output of the outer and inner encoder, respectively, at spatial position t. The compact graph of an SC-SCC of coupling memory $m = 1$ with underlying SCC of information block length K and rate $1/2$ is depicted in Fig. 1(a) for two spatial positions. In the figure, the variable nodes, represented by circles, correspond to the bit sequences $\boldsymbol{u}_t$, $\boldsymbol{v}^{\mathrm{O}}_t$, and $\boldsymbol{v}^{\mathrm{I}}_t$, and the constraint nodes, represented by squares, correspond to trellises. Specifically, the constraint nodes T^{O}_t and T^{I}_t represent the trellis of the outer and inner encoder of the underlying SCC, respectively. Each constraint node is labeled by the length of the corresponding trellis, i.e., K and $2K$ for the upper and lower trellis, respectively. The sequence $(\boldsymbol{u}_t, \boldsymbol{v}^{\mathrm{O}}_t)$ is reordered by a permutor $\boldsymbol{\Pi}^{(1)}_t$ (a permutor is indicated in the figure by a line crossing an edge) into the sequence $\tilde{\boldsymbol{v}}^{\mathrm{O}}_t$, which is divided into $m+1$ sequences of equal length, denoted by $\tilde{\boldsymbol{v}}^{\mathrm{O}}_{t,j}$, $j = 0, \ldots, m$. At spatial position t, the input to the inner encoder is $(\tilde{\boldsymbol{v}}^{\mathrm{O}}_{t-j,0}, \tilde{\boldsymbol{v}}^{\mathrm{O}}_{t-1,1}, \ldots, \tilde{\boldsymbol{v}}^{\mathrm{O}}_{t-m,m})$, properly reordered by a permutor $\boldsymbol{\Pi}^{(2)}_t$. The input constraint length is given by $c = K(m+1)$, which is equal to the number of information bits that contribute to the encoding of the parity bits at a given position t. Note that the construction in Fig. 1(a) [1], uses two individual permutors at each spatial position, which should be jointly optimized to yield an SC-SCC code with good distance properties.

2.2 A convolutional permutor perspective of coupling

The ensemble defined above can equivalently be described by the compact graph in Fig. 1(b), in which the variable nodes represent the combined sequences $\boldsymbol{u} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_L)$, $\boldsymbol{v}^{\mathrm{O}} = (\boldsymbol{v}^{\mathrm{O}}_1, \ldots, \boldsymbol{v}^{\mathrm{O}}_L)$, and $\boldsymbol{v}^{\mathrm{I}} = (\boldsymbol{v}^{\mathrm{I}}_1, \ldots, \boldsymbol{v}^{\mathrm{I}}_L)$ and the constraint nodes represent the combined trellises. The individual permutors in Fig. 1(a) can then be combined to a blockwise convolutional permutor $\boldsymbol{\Pi}_{\mathrm{c}}$, as illustrated in Fig. 1(b). The properties of $\boldsymbol{\Pi}_{\mathrm{c}}$ depend on the collection of permutors $\boldsymbol{\Pi}^{(1)}_t$ and $\boldsymbol{\Pi}^{(2)}_t$, $t = 1, \ldots, L$. If the same permutors $\boldsymbol{\Pi}^{(1)}_t = \boldsymbol{\Pi}^{(1)}$ and $\boldsymbol{\Pi}^{(2)}_t = \boldsymbol{\Pi}^{(2)}$ are used at each position t, then $\boldsymbol{\Pi}_{\mathrm{c}}$ is time-invariant. Unfortunately, such a design

may lead to poor distance properties.

2.3 Sliding-window decoding

To avoid a large decoding latency, SC-SCCs are decoded using sliding window decoding [5]. Following [3], the forward and backward recursions of the BCJR decoder span over all blocks within the window and the iteration number per window position is adapted to achieve a constant complexity for all considered block lengths. For a window size W, the decoding latency, denoted by Λ, is $\Lambda = WK$ information bits.

3 Deriving Periodic Blockwise Convolutional Permutors from a Block Permutor

In this paper, we propose a family of periodically time-varying convolutional permutors $\mathbf{\Pi}_\mathrm{c}$ with flexible block length as an alternative to a joint design of the individual permutors $\{\mathbf{\Pi}_t^{(1)}\}$ and $\{\mathbf{\Pi}_t^{(2)}\}$. In order to design an SC-SCC code with information block length K and coupling memory m, we start from a block permutor of size $M' = M(m+1)$, where $M = 2K$ denotes the length of the individual permutors in Fig. 1(a). As shown in [6–8], a bitwise convolutional permutor can be derived from a block permutor by an unwrapping procedure, similarly to the construction of LDPC convolutional codes [4]. We generalize this procedure to blockwise convolutional permutors of any even block length M, such that M divides M', resulting in a family of SC-SCC codes with fixed input constraint length $c = M'/2$ and varying block length $K = M/2$ and memory $m = M'/M - 1$.

We illustrate the construction of a convolutional permutor $\mathbf{\Pi}_\mathrm{c}$ with an example. Consider a block permutor $\mathbf{\Pi}$ of length $M' = 16$, as given in Fig. 2(a), and choose $M = 4$ and $m = 3$. First, the permutor $\mathbf{\Pi}$ is divided into square matrices of size M. Along these submatrices, we then divide the permutor below the blockwise main diagonal, as highlighted with a bold line in the figure. The lower-left part below the diagonal is then unwrapped to the right, as shown in Fig. 2(b), and the resulting matrix is replicated diagonally, as shown in Fig. 2(c). As a result, we obtain a periodically time-varying convolutional permutor $\mathbf{\Pi}_\mathrm{c}$ with a period of $m+1$ blocks, i.e., M' bits. A single period of $\mathbf{\Pi}_\mathrm{c}$ is highlighted by a bold line in Fig. 2(c). As indicated by the different colors, it can be divided into rectangular submatrices $\mathbf{\Pi}_\mathrm{c}^{(i)}$, $i = 0, \ldots, m$, of size $M' \times M$, containing one 1 in each column. Furthermore, it follows from the construction that, after m initial blocks, the convolutional permutor $\mathbf{\Pi}_\mathrm{c}$ has one 1 in each row and one 1 in each column.

Note that the same block permutor $\mathbf{\Pi}$ of length $M' = 16$ could also be used to derive convolutional permutors $\mathbf{\Pi}_\mathrm{c}$ with parameters $M = 2$ and $m = 7$ or $M = 8$ and $m = 1$. We will show below that for all possible unwrappings, the spread S of the original permutor $\mathbf{\Pi}$ is preserved.

Theorem 1. *Consider a convolutional permutor $\mathbf{\Pi}_c$ derived from a block permutor $\mathbf{\Pi}$ of length $M' = M(m+1)$ using the unwrapping procedure described above. Assume that $\mathbf{\Pi}_c$ has diagonal spread $S(\mathbf{\Pi}_c)$ and $\mathbf{\Pi}$ has circular diagonal*

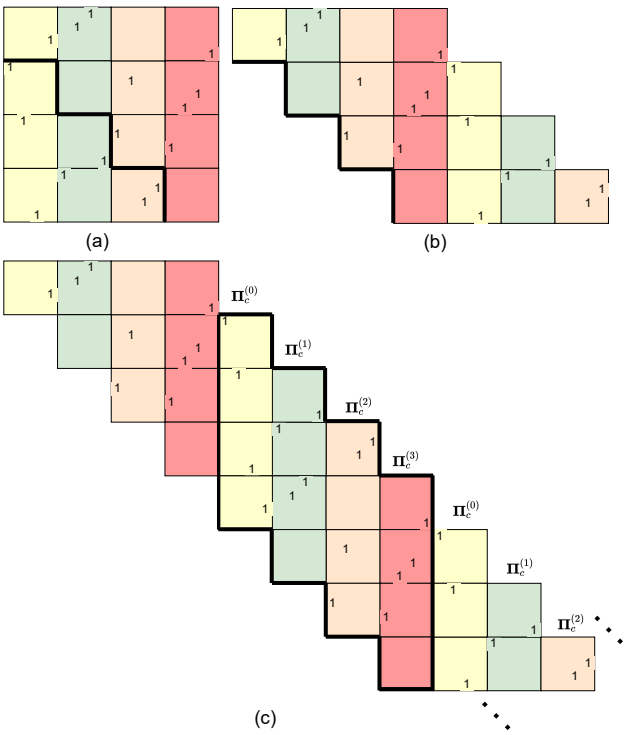


Figure 2: Construction of a convolutional permutor with block length $M = 4$ and memory $m = 3$ (a) block permutor $\mathbf{\Pi}$ of length $M' = 16$ (b) unwrapped block permutor (c) periodic convolutional permutor $\mathbf{\Pi}_\mathrm{c}$.

spread $S_L(\mathbf{\Pi})$ *(based on the Lee metric). Then*

$$S(\mathbf{\Pi}_c) \geq S_L(\mathbf{\Pi})$$

for any valid parameters M *and* $m+1$.

The proof of this theorem, together with the definitions of $S(\cdot)$ and $S_L(\cdot)$, can be found in the appendix.

Remark: In [2], it was shown for the ensemble in Fig. 1(a) that the minimum distance of an uncoupled code can be preserved if the individual permutors in Fig. 1(a) fulfill some special conditions with respect to the permutor of the uncoupled code. This result, however, was limited to time-invariant codes and derived from a block code with permutor length M. Hence it could not capture the gains expected with time-varying permutors and larger coupling memories $m > 1$. In this paper, we derive a convolutional permutor from a block permutor of length $M' = M(m+1)$ and show that the spread of this large permutor can be preserved.

4 Efficient Implementation of a Convolutional Permutor

4.1 A blockwise matrix representation of $\mathbf{\Pi}_\mathrm{c}$

Let $\tilde{\boldsymbol{u}}^\mathrm{I}$ and $\boldsymbol{u}^\mathrm{I}$ denote the sequences at the input and the output of the convolutional permutor $\mathbf{\Pi}_\mathrm{c}$, respectively. The mapping between these sequences can be performed blockwise by means of the $M' \times M$ submatrices $\mathbf{\Pi}_\mathrm{c}^{(i)}$. In particular, with $\tilde{\boldsymbol{u}}_t^\mathrm{I} = (u_{t,1}, v_{t,1}^\mathrm{O}, \ldots, u_{t,K}, v_{t,K}^\mathrm{O})$ the input of the inner encoder at position t can be expressed as

$$\boldsymbol{u}_t^\mathrm{I} = (\tilde{\boldsymbol{u}}_{t-m}^\mathrm{I}, \ldots, \tilde{\boldsymbol{u}}_t^\mathrm{I}) \cdot \mathbf{\Pi}_\mathrm{c}^{(i)} \ , \quad i = (t-1) \bmod (m+1) \ . \tag{1}$$

While (1) assumes that $\mathbf{\Pi}_\mathrm{c}$ is time-varying with period $m+1$, as constructed in Section 3, this can easily be adapted to general blockwise convolutional permutors. In particular, the encoding of the original SC-SCC ensemble described in Section 2 can equivalently be performed according to (1) for a corresponding set of submatrices $\mathbf{\Pi}_\mathrm{c}^{(i)}$, $i = 0, \ldots, L-1$.

4.2 Efficient indexing based on the block permutor $\mathbf{\Pi}$

Let us now present a method to implement the complete family of convolutional permutors $\mathbf{\Pi}_\mathrm{c}$ as introduced in Section 3 directly, using the block permutor $\mathbf{\Pi}$ they are derived from. The block length M can be chosen flexibly and no extra storage of the matrices $\mathbf{\Pi}_\mathrm{c}^{(i)}$ is required.

A general convolutional permutor $\mathbf{\Pi}_\mathrm{c}$ can be interpreted as a time-varying delay, which stores each symbol at the input for a specific amount of time units before releasing it to the output. All delays are measured with respect to the main diagonal. Let $\delta \geq 0$ and $\Delta < M'$ denote the minimal and maximal delay, respectively [8]. The number of delays required to represent a permutation is equal to its period in bits. We denote these M' delays by $d_{i,k}$, where $i = 0, \ldots, m$ and $k = 1, \ldots, M$. For a given i and k, let j denote the row in the matrix $\mathbf{\Pi}_\mathrm{c}^{(i)}$ that is connected to column k by a 1, i.e., $\{\mathbf{\Pi}_\mathrm{c}^{(i)}\}_{j,k} = 1$. Then $d_{i,k} = M' - j$. A block permutor $\mathbf{\Pi}$ corresponds to the special case of $m = 0$ and $M' = M$, and it can be represented by the delays d_k, $k = 1, \ldots, M'$.

Assume now that $\mathbf{\Pi}_\mathrm{c}$, defined by the delays $d_{i,k}$, is derived from $\mathbf{\Pi}$, defined by the delays d_k, by the unwrapping procedure for a given set of parameters m and M. Then the delays of $\mathbf{\Pi}_\mathrm{c}$ can be obtained as

$$d_{i,k} = d_{k+iM} + (i+1)M \bmod M' \ , \tag{2}$$

where $i = 0, \ldots, m$ and $k = 1, \ldots, M$.

Example 4.1. *Consider the permutor illustrated in Fig. 2(a). The first eight values d_k can be identified as* 11, 7, 0, 13, 3, 14, 15, *and* 5 *(yellow and green blocks). According to* (2)*, for $M = 4$ and $M' = 16$, these delays will be mapped to a convolutional permutor with delays $d_{0,k} =$* 15, 11, 4, 1 *and $d_{1,k} =$* 11, 6, 7, 12*. These values can be verified in Fig. 2(c).*

The delays $d_{i,k}$ of the convolutional permutor can now be used as alternative to the matrix representation in (1). More precisely, the k-th element of the input

$\Lambda = 4096$ bits			$\Lambda = 16384$ bits		
L	K	m	L	K	m
100	1024	1	100	4096	1
200	512	3	200	2048	3
400	256	7	400	1024	7
1600	64	31	1600	256	31

Table 1: SC-SCCs Configurations

sequence $\boldsymbol{u}_t^{\mathrm{I}}$ of the inner encoder can be identified as the j-th element of the sequence $(\tilde{\boldsymbol{u}}_{t-m}^{\mathrm{I}}, \ldots, \tilde{\boldsymbol{u}}_t^{\mathrm{I}})$ at the input of the permutor submatrix $\boldsymbol{\Pi}_{\mathrm{c}}$, where $j = M' - d_{i,k}$.

The storage requirements can be reduced further if the underlying permutor can be expressed explicitly by means of an equation. For example, for the QPP permutors used in the 3GPP LTE standard [9], the output index k for an input index j can be expressed as

$$k = (f_1 j + f_2 j^2) \bmod M' \tag{3}$$

for some suitable pair of parameters f_1 and f_2 that have to be chosen for a given length M'. In Section 5, we consider a QPP permutor of length $M' = 4096$ with $f_1 = 31$ and $f_2 = 64$ and compare its performance with that of an s-random permutor. Note that (3) cannot directly be combined with (2) for an on-the-fly implementation since the delays $d_{i,k}$ are expressed by the inverse permutation. However, if an explicit equation is not available for the inverse of $\boldsymbol{\Pi}$, then the storage of M' delays d_k can be avoided if the inverse permutation of $\boldsymbol{\Pi}_{\mathrm{c}}$ is implemented in the encoder instead.

5 Numerical Results

We investigate the performance of SC-SCCs with the proposed periodic blockwise convolutional permutors for fixed decoding latency and varying block length K and coupling memory m for transmission over the additive white Gaussian noise channel. In particular, we consider rate-1/3 SC-SCCs with constraint length 2048 and 8192 built from the concatenation of two rate-1/2, 4-state component encoders with generator matrix $\left(1, \frac{1+\mathsf{D}^2}{1+\mathsf{D}+\mathsf{D}^2}\right)$. To achieve rate 1/3, we puncture every other parity bit of the inner code as in [3]. The considered chain lengths L, block length K, and coupling memory m are listed in Table 1.

For the simulation results, we use sliding window decoding with window size $2(m+1)$, corresponding to a decoding latency $\Lambda = 2(m+1)K$ information bits. Further, we assume 80 decoding iterations per decoded bit and consider both s-random permutors with optimized circular diagonal spread and QPP permutors to obtain the convolutional permutors. As shown in Table 2, the spread values are equal for all m.

In Fig. 3, we show the bit error rate (BER), and block error rate (BLER) performance for SC-SCCs with periodic blockwise convolutional permutors obtained from a block s-random permutor for $\Lambda = 4096$ bits (solid curves) and

Size	Type	Spread $S_L(\mathbf{\Pi})$	Spread $S(\mathbf{\Pi}_c)$	
4096	s-random	21	$m = 1$	21
	s-random		$m = 31$	21
	QPP	32	$m = 1$	32
	QPP		$m = 31$	32
16384	s-random	60	$m = 1$	60
	s-random		$m = 255$	60

Table 2: Spread of the considered permutors

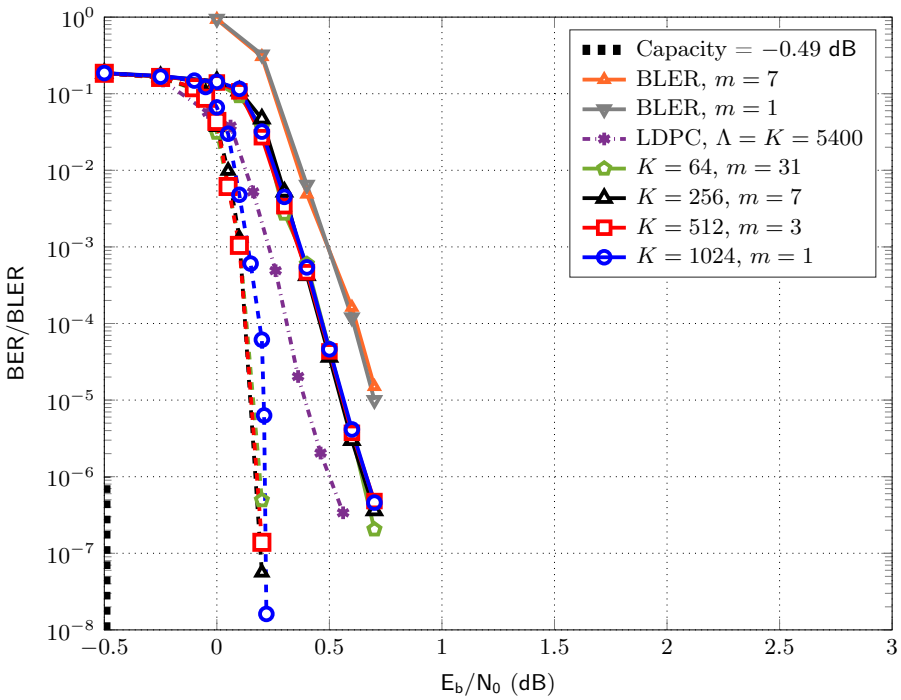


Figure 3: BER performance of SC-SCCs with periodic blockwise convolutinal permutors obtained from s-random permutors for $\Lambda = 4096$ (solid curves), $\Lambda = 16384$ (dashed curves).

$\Lambda = 16834$ bits (dashed curves) and varying K and m. For comparison, we also show the BER of an LDPC code from the DVB-S2 standard (short), taken from [10]. Interestingly, for a fixed decoding latency, the performance is almost identical by varying the block length K, even for K as small as 64 bits. Note that if K is halved, to achieve the same decoding latency, the coupling memory needs to be increased to $2m + 1$. In other words, the impact of a weaker underlying SCC is perfectly compensated by a larger coupling memory. A similar phenomenon is observed in Fig. 4 for blockwise convolutional permutors obtained from a QPP permutor. Further, the performance is almost identical to that of SC-SCCs with underlying s-random permutor, underscoring the fact that SC-SCCs with permutors that can be expressed by means of an equation—and

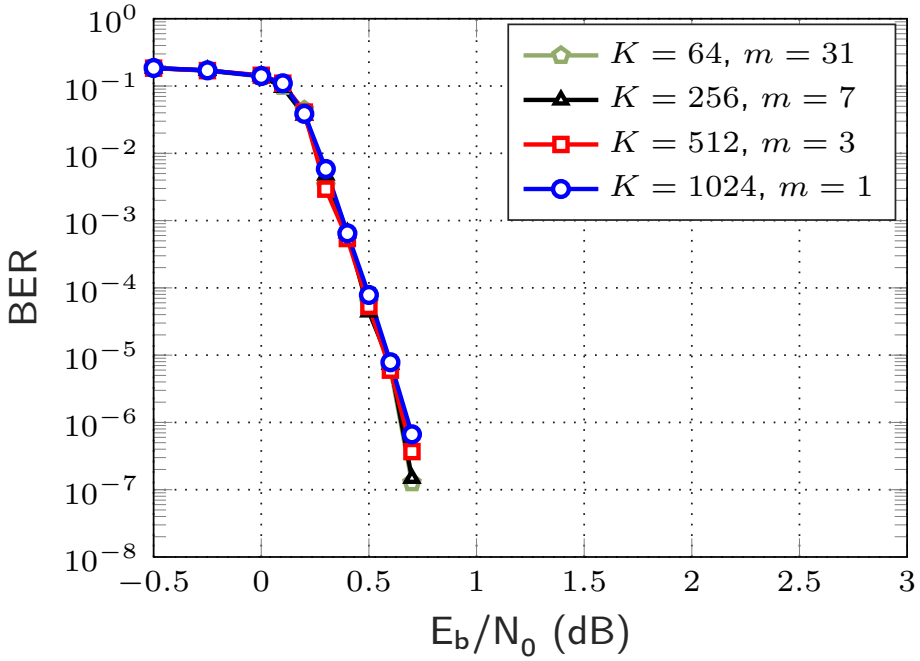


Figure 4: BER performance of SC-SCCs with periodic blockwise convolutinal permutors obtained from a QPP permutor for $\Lambda = 4096$.

hence are implementation-friendly—yield good performance.

In Fig. 5, we compare the performance of SC-SCCs with the proposed convolutional-based permutors (blue curves with circle markers) with that of SC-SCCs with independently-optimized individual permutors [1] (red curves with square markers) for $\Lambda = 4096$, and varying K (and accordingly m). The curves for the SC-SCCs with individual permutors are borrowed from [3]. Notably, the SC-SCCs with the proposed periodic blockwise permutors yield lower error floors than those using independently-designed individual permutors. This is especially noticeable for the shortest block length, $K = 64$, for which the code with individual permutors shows a very high error floor, as already noticed in [3], while no visible error floor is visible for the proposed scheme below 10^{-6}. A similar behavior is observed in Fig. 6 for $\Lambda = 16384$.

6 Conclusion

We introduced a family of blockwise periodically time-varying convolutional permutors for designing SC-SCCs with flexible block length and coupling memory. The permutors can easily be derived from a given block permutor with a procedure that preserves the spread. We demonstrated via simulations that the proposed convolutional permutors make it possible to vary the component code block length at a fixed decoding latency and complexity without any noticeable performance loss, yielding almost identical BER performance.

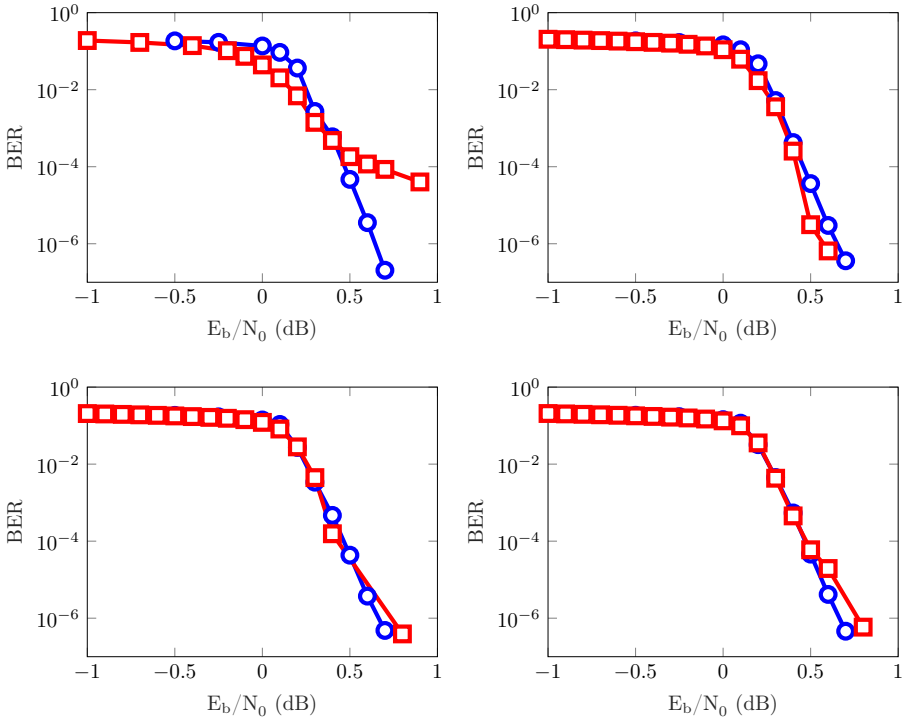


Figure 5: BER comparison of SC-SCCs with convolutional permutors vs SC-SCCs with independent permutors for a decoding latency of $\Lambda = 4096$.

Appendix: Proof of Theorem 1

For a given permutor $\mathbf{\Pi}$, let j_1, j_2 be a pair of distinct positions in the input sequence. The corresponding positions in the output sequence, k_1, k_2, are then defined by the condition $(\mathbf{\Pi})_{j_1,k_1} = 1$ and $(\mathbf{\Pi})_{j_2,k_2} = 1$. Following the notation in [11], the sequence j_1, k_1, j_2, k_2, j_1 forms a cycle of length 2 in the permutor. The diagonal spread of a permutor cycle is defined as $S(j_1, j_2) = |j_1 - j_2| + |k_1 - k_2|$ [12]. For a block permutor of length M', we can also define the circular diagonal spread of a cycle as $S_L(j_1, j_2) = |j_1 - j_2|_L + |k_1 - k_2|_L$, where the distances are now defined according to the Lee metric, i.e., $|a - b|_L = \min(|a - b|, M' - |a - b|)$. The spread of a permutor can now be defined as the minimum spread over all possible cycles, i.e.,

$$S(\mathbf{\Pi}) = \min_{j_1,j_2} S(j_1, j_2) \quad \text{and } S_L(\mathbf{\Pi}) = \min_{j_1,j_2} S_L(j_1, j_2) \ . \tag{4}$$

Assume now that $\mathbf{\Pi_c}$ is a convolutional permutor derived from $\mathbf{\Pi}$ by the unwrapping procedure. Consider a minimal cycle j_1, k_1, j_2, k_2, j_1 in $\mathbf{\Pi_c}$ such that $S(j_1, j_2) = S(\mathbf{\Pi_c})$. From the construction it follows that this cycle in $\mathbf{\Pi_c}$ implies the existence of a cycle $j'_1, k'_1, j'_2, k'_2, j'_1$ in $\mathbf{\Pi}$ with

$$j'_x = (j_x - 1) \bmod (M') + 1 \ , \ \ k'_x = (k_x - 1) \bmod (M') + 1 \ ,$$

where $x = 1, 2$ and $j'_x, k'_x \in \{1, \ldots . M'\}$. With this, it can be verified that $|j'_1 - j'_2|_L \leq |j_1 - j_2|$ and $|k'_1 - k'_2|_L \leq |k_1 - k_2|$. Hence, $S_L(j'_1, j'_2) \leq S(j_1, j_2)$

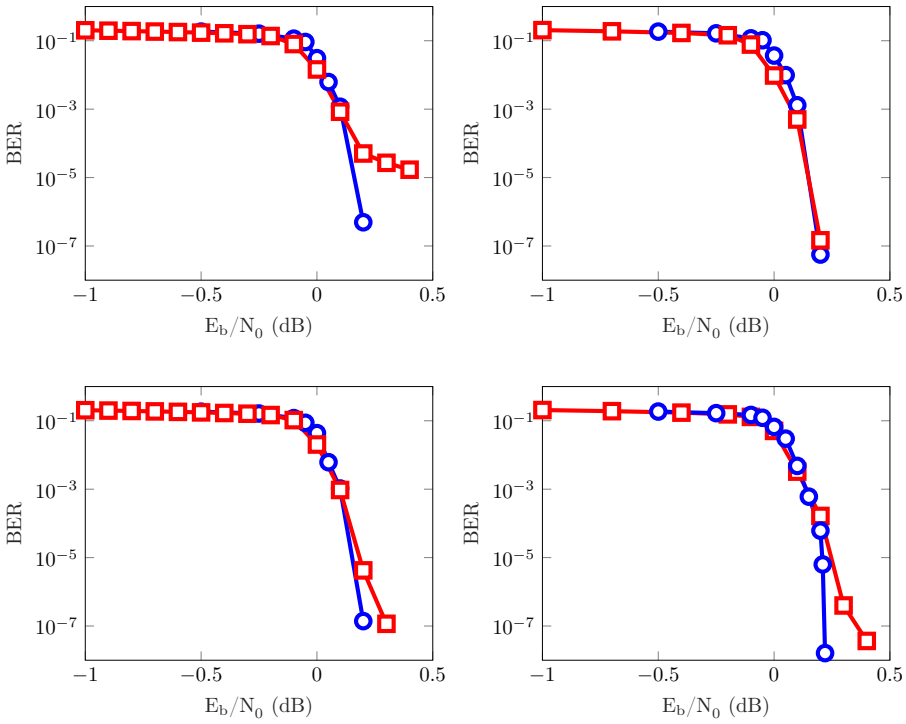


Figure 6: BER comparison of SC-SCCs with convolutional permutors vs SC-SCCs with independent permutors for a decoding latency of $\Lambda = 16384$.

and it follows that $S_L(\mathbf{\Pi})$ cannot be larger than $S(\mathbf{\Pi}_{\boldsymbol{c}})$, which completes the proof.

References

[1] S. Moloudi, M. Lentmaier, and A. G. i Amat, "Spatially coupled turbo-like codes," *IEEE Trans. Inf. Theory*, vol. 63, pp. 6199–6215, Oct. 2017.

[2] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Spatially coupled turbo-like codes: A new trade-off between waterfall and error floor," *IEEE Trans. Commun.*, vol. 67, pp. 3114–3123, May 2019.

[3] M. Mahdavi and M.U. Farooq and L. Liu and O. Edfors and V.Ö. Viktor and M. Lentmaier, "The Effect of Coupling Memory and Block Length on Spatially Coupled Serially Concatenated Codes," in *Proc. IEEE Veh. Technol. Conf. (VTC)*, Apr. 2021.

[4] A. Jiménez Feltström and K.Sh. Zigangirov, "Periodic time-varying convolutional codes with low-density parity-check matrices," *IEEE Trans. Inf. Theory*, vol. 45, pp. 2181–2190, Sep. 1999.

[5] A. Iyengar, M. Papaleo, P. Siegel, J. Wolf, A. Vanelli-Coralli, and G. Corazza, "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels," *IEEE Trans. Inf. Theory*, vol. 58, pp. 2303–2320, Apr. 2012.

[6] A. Feltström, D. Truhachev, M. Lentmaier, and K. Zigangirov, "Braided block codes," *IEEE Trans. Inf. Theory*, vol. 55, pp. 2640–2658, June 2009.

[7] W. Zhang, M. Lentmaier, K.Sh. Zigangirov, and D.J. Costello, Jr., "Braided convolutional codes: a new class of turbo-like codes," *IEEE Trans. Inf. Theory*, vol. 56, pp. 316–331, Jan. 2010.

[8] R. Johannesson and K.Sh. Zigangirov, *Fundamentals of Convolutional Coding*. Piscataway, NJ: IEEE Press, 2nd ed., 2015.

[9] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding," Technical Specification (TS) 36.212, 3rd Generation Partnership Project (3GPP), Sep 2019. Version 15.3.0.

[10] G. Liva and F. Steiner, "pretty-good-codes.org: Online library of good channel codes," Aug. 2021.

[11] D. V. Truhachev, M. Lentmaier, and K. Sh. Zigangirov, "Some results concerning the design and decoding of turbo-codes," *Problems of Information Transmission*, vol. 37, pp. 190–205, July 2001.

[12] P. Popovski, L. Kocarev, and A. Risteski, "Design of flexible-length s-random interleaver for turbo codes," *IEEE Commun. Lett.*, vol. 8, pp. 461–463, July 2004.

Paper IV

Generalized LDPC Codes with Convolutional Code Constraints

Braided convolutional codes (BCCs) are a class of spatially coupled turbo-like codes that can be described by a $(2, 3)$-regular compact graph. In this paper, we introduce a family of (d_v, d_c)-regular GLDPC codes with convolutional code constraints (CC-GLDPC codes), which form an extension of classical BCCs to arbitrary regular graphs. In order to characterize the performance in the waterfall and error floor regions, we perform an analysis of the density evolution thresholds as well as the finite-length ensemble weight enumerators and minimum distances of the ensembles. In particular, we consider various ensembles of overall rate $R = 1/3$ and $R = 1/2$ and study the trade-off between variable node degree and strength of the component codes. We also compare the results to corresponding classical LDPC codes with equal degrees and rates. It is observed that for the considered LDPC codes with variable node degree $d_v > 2$, we can find a CC-GLDPC code with smaller d_v that offers similar or better performance in terms of BP and MAP thresholds at the expense of a negligible loss in the minimum distance.

M. U. Farooq, S. Moloudi and M. Lentmaier,
"Generalized LDPC Codes with Convolutional Code Constraints", *2020 IEEE International Symposium on Information Theory (ISIT)*, LA, USA, 2020.

1 Introduction

Turbo codes and low-density parity-check (LDPC) codes are widely used forward error correction techniques in many communication applications. For LDPC convolutional codes [1, 2], also known as spatially coupled LDPC (SC-LDPC) codes, it has been proved that the threshold of efficient belief propagation (BP) decoding saturates to the threshold of an optimal maximum a-posteriori probability (MAP) decoder [3, 4]. Spatially coupled turbo-like codes were introduced in [5], where it was proved that threshold saturation also occurs for this class of codes. It was observed that turbo-like codes with good BP thresholds tend to have weaker MAP thresholds and minimum distance [5, 6]. Braided convolutional codes (BCCs) [7], which are characterized by $(2, 3)$-regular graphs, have better MAP thresholds and distances than parallel concatenated convolutional codes that suffer from degree-one variable nodes. In combination with spatial coupling, ensembles with good MAP thresholds and low error floors are able to simultaneously approach capacity and achieve very low error floor thanks to the threshold saturation phenomenon [6].

In principle, it is possible to improve the threshold and minimum distance of an SC-LDPC ensemble by increasing the variable node degree. For finite block lengths, however, ensembles with stronger component codes can have advantages [8], since larger variable node degrees increase the number of short cycles in the factor graph, which negatively impacts the performance of a BP decoder. Furthermore, it has been observed in [6] that, due to the stronger component codes at the constraint nodes, spatially coupled turbo-like ensembles can achieve excellent decoding thresholds and minimum distances with low variable node degrees.

In this work, our aim is to gain a better understanding of the general trade-off between increasing the variable node degree or the strength of the component codes. For this purpose, we introduce a family of (d_v, d_c)-regular generalized LDPC codes with convolutional code constraints (CC-GLDPC codes), which form an extension of classical BCCs to arbitrary regular graphs and allow for a one-to-one comparison with the corresponding (d_v, d_c)-regular LDPC code ensembles. As examples we consider $(2, 3)$, $(4, 6)$ and $(6, 9)$ graphs of rate $R = 1/3$ as well as $(2, 4)$, $(3, 6)$ and $(4, 8)$ graphs of rate $R = 1/2$, based on component code trellises with 2, 4 and 8 states. For these ensembles we determine the BP thresholds (with and without spatial coupling), MAP thresholds and minimum distances and compare them with the corresponding LDPC code ensembles.

2 Code Ensembles

2.1 An Ensemble of (d_v, d_c)-regular GLDPC Codes with Convolutional Code Constraints

Braided convolutional codes can be viewed as a class of turbo-like codes with parity-feedback between the component encoders, as illustrated in Fig. 1(a). Since the parity symbols enter the other encoder after a delay of one block of N symbols, BCCs are inherently spatially coupled. An uncoupled version of BCCs can be obtained by removing this delay. Fig. 1(b) shows a compact graph representation of the uncoupled BCCs, in which the variable nodes represent the

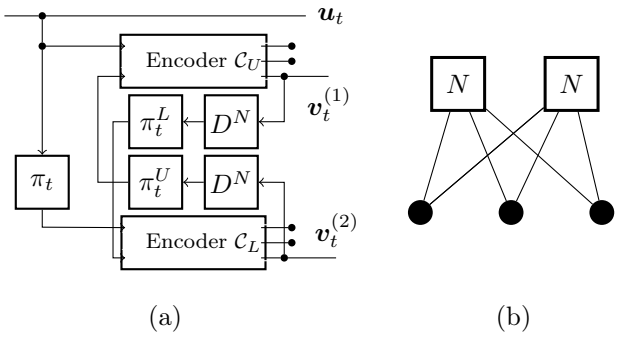


Figure 1: Classical BCCs as (2,3)-regular ensemble: (a) BCC encoder (b) compact graph (uncoupled).

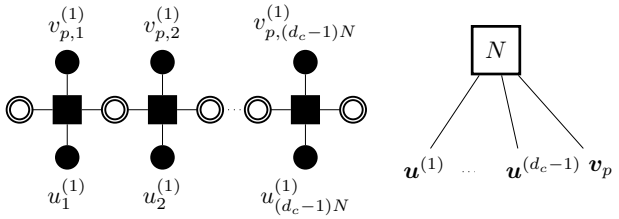
different blocks of code symbols and the constraint nodes represent the length N component encoder trellises of rate 2/3. The permutations occur along the edges of the compact graph.

Observe that the compact graph of the original BCCs is a fully connected $(2, 3)$-regular graph, analogous to the protograph of a $(2, 3)$-regular LDPC code. In order to generalize BCCs to larger variable node degrees, we can increase the number of component encoders as well as their number of inputs, resulting in $(3, 4)$-regular graphs with overall rate $R = 1/4$, $(4, 5)$-regular graphs with rate $R = 1/5$, and so on. These ensembles, however, are hard to compare due to their different rates. Alternatively, we can add edges to the $(2, 3)$-regular graph, obtaining $(4, 6)$ or $(6, 9)$-regular graphs without changing the original rate $R = 1/3$. In general, using d_v component encoders of rate $(d_c - 1)/d_c$ we can construct arbitrary (d_v, d_c)-regular CC-GLDPC codes. Moreover, these codes can be directly compared to the corresponding (d_v, d_c)-regular LDPC codes, which have the same overall design rate $R = 1 - d_v/d_c$ and the same length if their lifting factor is set equal to the number of sections in the trellises.

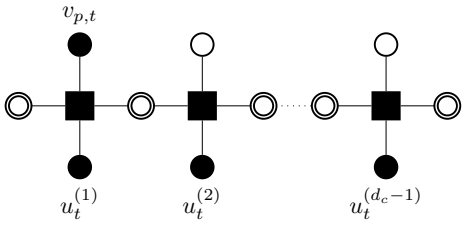
2.2 Punctured Trellises for Degree d_c Constraint Nodes

Consider a degree d_c constraint node in the compact graph. Each edge represents a length N sequence of code symbols that are represented by the connected variable node. One of these sequences will correspond to the parity sequence $\boldsymbol{v}_p = (v_{p,1}, \ldots, v_{p,N})$ of the component convolutional code and the other $d_c - 1$ to the information sequences $\boldsymbol{u}^{(i)} = (u_1^{(i)}, \ldots, u_N^{(i)})$, $i = 1, \ldots, d_c - 1$.

In order to construct a component code of rate $(d_c - 1)/d_c$, for any $d_c \geq 2$ we can use a rate-1/2 mother code with a trellis of $(d_c - 1)N$ sections. A factor graph of such a trellis is shown in Fig. 2. The desired code rate is achieved by puncturing $d_c - 2$ of the parity bits in each segment of $d_c - 1$ trellis sections, as shown by white circles in the factor graph. For example, the degree $d_c = 3$ constraint node of the classical BCC graph in Fig. 1(b) can be implemented by a trellis of length $2N$ in which every second parity bit is punctured to achieve a rate-2/3 component encoder. In general, the puncturing patterns in different segments of the trellis can be time-varying, and finding patterns that optimize the thresholds or the distance spectrum of the resulting codes is an open problem. In our threshold analysis, we will assume



(a) Rate 1/2 trellis (left) in a constraint node (right).



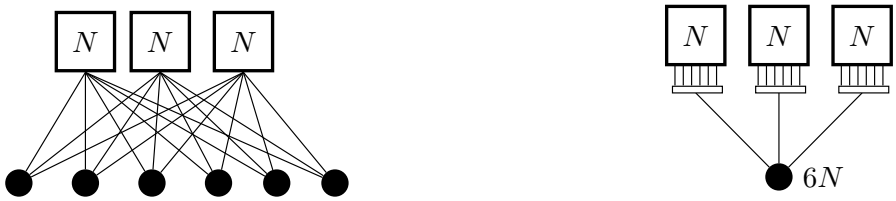
(b) Trellis section at time t after puncturing to rate $(d_c - 1)/d_c$.

Figure 2: Factor graph representation of a constraint node trellis.

uniform random puncturing within each segment, such that a parity bit remains unpunctured with probability $1/(d_c - 1)$.

The strength of the constraint nodes can be flexibly changed without altering their degree by simply increasing the number of states in the component code trellis. In this work, we consider recursive systematic convolutional encoders with generator polynomials $(1, 1/3)$, $(1, 5/7)$ and $(1, 13/15)$ in octal notation, with 2, 4 and 8 trellis states, respectively.

2.3 Single-Edge Type Ensembles



(a) Structured

(b) Single-edge type

Figure 3: Graph represenations of uncoupled (3,6)-regular ensembles.

The compact graph of a $(3, 6)$-regular ensemble is shown in Fig. 3(a). Since the edges define a clear assignment from outputs of the constraint nodes to the variable nodes, this is an example of a structured graph. In order to simplify analysis, when computing ensemble weight enumerators and thresholds, we will instead consider single-edge type ensembles like shown in Fig. 3(b).

3 Finite-Length Ensemble Weight Enumerators

A weight enumerator analysis for different ensembles of turbo-like codes, including uncoupled BCCs, was carried out in [6]. Considering the $(2,3)$-regular ensemble in Fig. 1(b), let $A^{(j)}_{i_1,i_1,p}$ denote the number of code sequences of input weights i_1, i_2 and parity weight p for the length N convolutional code at constraint node j. Assuming uniform random permutations, it is then possible to compute the average number of codewords $\bar{A}^{BCC}_{i,p}$ over all codes in the ensemble as follows:

$$\bar{A}^{BCC}_{i,p} = \sum_{p_1} \frac{A^{(1)}_{i,p_1,p-p_1} \cdot A^{(2)}_{i,p-p_1,p_1}}{\binom{N}{i}\binom{N}{p_1}\binom{N}{p-p_1}} \ . \tag{1}$$

In principle, a generalization to general structured (d_v, d_c) ensembles, like illustrated in Fig. 3(a), is possible[1]. Unfortunately, this approach becomes numerically infeasible for a given N when the variable node degree increases. In this work, we consider unstructured single-edge type ensembles, like illustrated in Fig. 3(b), and generalize Gallager's weight enumerator analysis for LDPC codes [10, 11] to our ensembles:

$$\bar{A}^{(d_v,d_c)}_{w} = \frac{\left(A^{(j)}_{w}\right)^{d_v}}{\binom{d_c \cdot N}{w}^{d_v-1}} \ , \tag{2}$$

where w is the total weight of information and parity bits.

A bound on the minimum distance d_{min} of codes in an ensemble can be obtained by computing the largest positive integer $\hat{d}$ that satisfies the expression

$$\sum_{w=1}^{\hat{d}-1} \bar{A}^{(d_v,d_c)}_{w} < 1 - \alpha \tag{3}$$

for a given $\alpha < 1$. Then a fraction α of all codes in the ensemble must have a minimum distance $d_{min} \geq \hat{d}$.

4 Convergence Thresholds for the BEC

We assume that the BP decoder of a CC-GLDPC code is based on optimal bitwise a-posteriori probability (APP) decoding at the constraint nodes[2]. The MAP decoding threshold, on the other hand, refers to the optimal bitwise decoding of the overall code, which is computationally infeasible. For the BEC it is possible to find analytical expressions for the input/output transfer functions of the component decoders [13]. By means of these it is possible to derive exact DE equations for the ensembles introduced in Section II, which capture the evolution of erasure probabilities of messages being passed back and forth along the edges in the graph.

[1] As pointed out in [6], the weight enumerator expression in (1) is equivalent to protograph-based GLDPC code ensembles analyzed in [9].

[2] This is the equivalent to the classical turbo decoder [12].

4.1 Density Evolution Equations for Uncoupled Ensembles

Consider a (d_v, d_c)-regular graph and let $e_{j,k}$ denote the edge connecting variable node j to constraint node k, where $j \in \{1, \ldots, d_v\}$ and $k \in \{1, \ldots, d_c\}$. The DE update at a constraint node in iteration i can be expressed as

$$p^{(i)}(e_{j,k}) = f_k \left(q^{(i-1)}(e_{j,1}), \ldots, q^{(i-1)}(e_{j,d_c}) \right) , \tag{4}$$

where $q^{(i)}(e_{j,k})$ and $p^{(i)}(e_{j,k})$ denote the probabilities that messages passed from variable to check nodes and from check nodes to variable nodes are erased, respectively. f_k denotes the (extrinsic) transfer function of the constraint node, corresponding to the trellis output message type associated with edge $e_{j,k}$. For conventional LDPC codes this transfer function is independent of k and reduces to the well-known expression

$$p^{(i)}(e_{j,k}) = 1 - \prod_{k' \setminus k} \left(1 - q^{(i-1)}(e_{j,k'}) \right) \tag{5}$$

$$= 1 - \left(1 - q^{(i-1)} \right)^{d_c - 1} . \tag{6}$$

Before the first iteration $i = 1$, all input erasure probabilities are initialized to $q^{(0)}(e_{j,k}) = \epsilon$, which is the erasure probability of the BEC. At a variable node, the DE update can be written as

$$q^{(i)}(e_{j,k}) = \epsilon \cdot \prod_{j' \setminus j} p^{(i)}(e_{j',k}) = \epsilon \cdot \left(p^{(i)} \right)^{d_v - 1} . \tag{7}$$

In this work we consider single-edge type regular graphs, as illustrated in Fig. 3(b), for which the trellis outputs of the constraint nodes are distributed uniformly over all code symbols of the variable node. In this case $p^{(i)}$ and $q^{(i-1)}$ are equal along all edges of the graph.

4.2 Transfer Functions for Punctured Trellises

In order to compute the transfer functions of the constraint nodes of the graph, a rate-1/2 trellis is punctured to match the constraint node degree of the graph. The mother code transfer functions for the considered generator polynomials can be derived as shown in [5]. Let f_s and f_p denote these transfer functions for systematic and parity bits, respectively. Then we can write

$$p_\mathrm{s}^{(i)} = f_\mathrm{s} \left(q_\mathrm{s}^{(i)}, q_\mathrm{p}^{(i)} \right) , \tag{8}$$

$$p_\mathrm{p}^{(i)} = f_\mathrm{p} \left(q_\mathrm{s}^{(i)}, q_\mathrm{p}^{(i)} \right) , \tag{9}$$

where $p_\mathrm{s}^{(i)}$ and $p_\mathrm{p}^{(i)}$ denote the extrinsic output erasure probabilities, and $q_\mathrm{s}^{(i)}$ and $q_\mathrm{p}^{(i)}$ the erasure probabilities of incoming messages to the constraint node. Assuming that random puncturing of parity bits is used for achieving a target rate $(d_c - 1)/d_c$, these input erasure probabilities are given by

$$q_\mathrm{s}^{(i)} = q^{(i-1)} , \tag{10}$$

$$q_\mathrm{p}^{(i)} = \frac{d_c - 2}{d_c - 1} \cdot 1 + \frac{1}{d_c - 1} \cdot q^{(i-1)} = \frac{q^{(i-1)} + d_c - 2}{d_c - 1} . \tag{11}$$

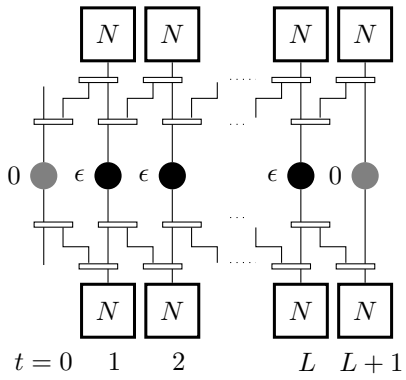


Figure 4: Spatially coupled $(2, 3)$-regular ensemble: single-edge type representation with coupling memory $m = 1$.

The average erasure probability of messages sent from the constraint nodes to the variable node is equal to

$$p^{(i)} = \frac{(d_c - 1)}{d_c} \cdot p_{\mathrm{s}}^{(i)} + \frac{1}{d_c} \cdot p_{\mathrm{p}}^{(i)} \ . \tag{12}$$

DE iteration i is then completed by a variable note update according to (7), resulting in $q^{(i)}$.

4.3 Density Evolution Equations for Coupled Ensembles

For spatially coupled ensembles, as illustrated in Fig. 4, we have a sequence of L graphs whose constraint nodes and variable nodes are placed at time instants $t = 1, \ldots, L$. We consider ensembles with uniform coupling, i.e., every edge from a variable node at time t is connected to a constraint node at time $t' \in \{t, t+1, \ldots, t+m\}$ with probability $1/(m+1)$, where m is called the coupling memory.

For conventional SC-LDPC codes a constraint node represents a rate $(d_c - 1)/d_c$ single parity-check code and the update equation becomes

$$p_t^{(i)} = 1 - \left(1 - \frac{1}{m+1} \sum_{\ell=0}^{m} q_{t-\ell}^{(i-1)}\right)^{d_c - 1} \ . \tag{13}$$

For CC-GLDPC codes with punctured component code trellises, we update the transfer functions (8)–(9) with the input erasure probabilities

$$q_{\mathrm{s},t}^{(i)} = \frac{1}{m+1} \sum_{\ell=0}^{m} q_{t-\ell}^{(i-1)} \ , \tag{14}$$

$$q_{\mathrm{p},t}^{(i)} = \frac{q_{\mathrm{s},t}^{(i)} + d_c - 2}{d_c - 1} \tag{15}$$

and obtain $p_t^{(i)}$ analogously to (12) for each t. Before the first iteration $i = 1$, the input erasure probabilities are initialized to $q_t^{(0)} = \epsilon$ for $t \in \{1, \ldots, L\}$. For all other t, the code symbols are known to be zero by definition and $q_t^{(0)} = 0$.

At the variable nodes, the DE update can be written as

$$q_t^{(i)} = \epsilon \cdot \left(\frac{1}{m+1} \sum_{\ell=0}^{m} p_{t+\ell}^{(i)} \right)^{d_v - 1} , \quad t = 1, \ldots, L \ . \tag{16}$$

4.4 BP and MAP Thresholds

The BP threshold ϵ_{BP} is defined as the largest channel erasure probability ϵ for which the a-posteriori erasure probabilities $p_a^{(i)}$ at the output of the BP decoder converge to zero for all variable nodes as the number of iterations i tends to infinity. The probabilities $p_a^{(i)} = \epsilon \cdot \left(p^{(i)}\right)^{d_v}$ can be computed by repeated use of the density evolution equations for different ϵ. The (bitwise) MAP threshold ϵ_{MAP} can obtained by applying the area theorem [13,14], which makes it possible to connect the performance under BP decoding to that of MAP decoding. Let $\bar{p}_e(\epsilon) = \lim_{i\to\infty} \bar{p}_a(\epsilon)^{(i)}/\epsilon$ denote the average extrinsic probability of erasure. An upper bound on the MAP threshold can be computed by the equation

$$\int_{\epsilon_{MAP}}^{1} \bar{p}_e(\epsilon) d\epsilon = R \ , \tag{17}$$

where R is the rate of the considered code.

5 Results and Discussion

5.1 Minimum Distance Bounds

The minimum distance bounds, computed using (3) with $\alpha = 0.5$ for the ensembles of rate $R = 1/3$ and $R = 1/2$, are shown in Fig. 5(a) and (b), respectively. It follows from the bound that half of the codes in an ensemble must have minimum distance $d_{min} \geq \hat{d}$. From the figure, it is observed that in general the minimum distance improves when the component code gets stronger. Furthermore, for a given component code the distance improves if the variable node degree is increased. Interestingly, for $R = 1/3$, the weakest CC-GLDPC codes with 2-state components appear to have better minimum distance than classical LDPC codes. The results show that we indeed can reduce the variable node degree if we increase the number of states of the component encoder. For example, codes from the $(3, 6)$ ensemble with 4 states have better minimum distance than those of the $(4, 8)$ ensemble with 2 states and the $(4, 8)$ LDPC ensemble. As expected [10], the minimum distances of LDPC ensembles with variable node degree 2 are very poor, which is also observed for the 2-state CC-GLDPC ensembles.

5.2 Thresholds

Table 1 shows the BP thresholds and MAP thresholds for the uncoupled ensembles of rate $R = 1/3$ and $R = 1/2$. It is observed that BP thresholds tend to *decrease* with increasing variable node degree and increasing number of trellis states. However, MAP thresholds tend to *increase* with increasing variable node

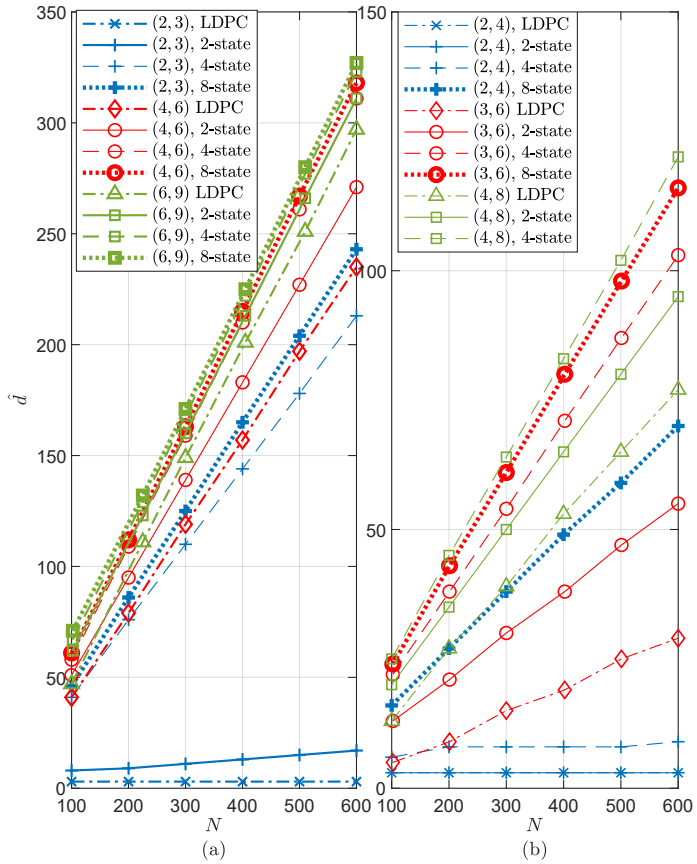


Figure 5: Bound on the minimum distance: a fraction $\alpha = 1/2$ of codes in an ensemble have $d_{min} \geq \hat{d}$: (a) $R = 1/3$ (b) $R = 1/2$.

degree and increasing number of trellis states. An exception from this behavior is observed for 2-state ensembles and LDPC ensembles with variable node degree 2 at rate $R = 1/2$.[3]

In order to understand the trade-off between variable node degree and number of trellis states, we compare the minimum distances, BP thresholds and MAP thresholds of the $(2, 3)$ ensemble with 4 states and 8 states, the $(4, 6)$ ensemble with 4 states, and the $(6, 9)$ ensemble with 2 states. The $(2, 3)$ ensemble with 8 states has better BP and MAP thresholds than the $(6, 9)$ ensemble with 2 states, but the minimum distance is clearly worse. If a comparable minimum distance is a requirement, but it is desired to keep the variable node degree as low as possible, then the $(4, 6)$ ensemble with 4 states can be used instead. In terms of the BP decoding performance, this ensemble is not as good as the $(2, 3)$ ensemble with 8 states or the $(6, 9)$ ensemble with 2 states, but it has the best MAP threshold among these three ensembles. The strong MAP threshold of the $(4, 6)$ ensemble with 4 states makes it a compelling candidate for spatial

[3]These ensembles are poor and not of practical interest. As shown in [15], regular GLDPC ensembles with $d_v < 3$ require component codes with minimum distance $d_{min} > 2$ in order to guarantee that the block error probability tends to zero at the BP threshold.

Thresholds	**States**	**Graph rate 1/3**		
		$(2,3)$	$(4,6)$	$(6,9)$
ϵ_{BP}	2	0.6086	0.5339	0.4698
ϵ_{MAP}	2	0.6213	0.6564	0.6610
ϵ_{BP}	4	0.5618	0.4464	0.3853
ϵ_{MAP}	4	0.6647	0.6662	0.6664
ϵ_{BP}	8	0.5352	0.4041	0.3401
ϵ_{MAP}	8	0.6659	0.6665	0.6666
ϵ_{BP}	LDPC	0.2570	0.5061	0.4034
ϵ_{MAP}	LDPC	0.5089	0.6658	0.6667
		Graph rate 1/2		
		$(2,4)$	$(3,6)$	$(4,8)$
ϵ_{BP}	2	0.3234	0.4110	0.3916
ϵ_{MAP}	2	0.3444	0.4557	0.4737
ϵ_{BP}	4	0.4426	0.3929	0.3555
ϵ_{MAP}	4	0.4890	0.4958	0.4976
ϵ_{BP}	8	0.4249	0.3638	0.3225
ϵ_{MAP}	8	0.4955	0.4985	0.4991
ϵ_{BP}	LDPC	0.1725	0.4294	0.3834
ϵ_{MAP}	LDPC	0.4002	0.4883	0.4978

Table 1: Thresholds of uncoupled ensembles.

coupling.

The BP thresholds of the spatially coupled ensembles are shown in Table 2 for different coupling memories m, until saturation to the MAP threshold occurs. Due to the threshold saturation phenomenon, the BP thresholds approach the MAP thresholds as m increases and the $(4,6)$ ensemble with 4 states from our example above has now a better BP threshold than the other considered ensembles.

6 Conclusion

We have introduced a family of GLDPC codes with convolutional code constraints, which allows a one-by-one comparison with corresponding LDPC code ensembles of arbitrary variable node and check node degrees. Although we have focused in this work on regular graphs only, the ensembles can easily be extended to irregular codes by removing some edges in the graphs. Furthermore, it is possible to use component codes of lower rate at the constraint nodes, but then the rate of the resulting ensembles will be different from the LDPC code ensembles defined by the same graphs. An advantage of using convolutional codes at the constraint nodes is that the strength of the component codes can be altered without changing the node degrees in the graph.

The considered ensembles permit us to study the trade-off between variable node degree and component code strength in terms of their minimum distance, BP decoding thresholds and MAP decoding thresholds.[4] A larger number of

[4]Note that the computational complexity increases with the number of trellis states and

States/SC	Graph					
memory	$(2,3)$	$(4,6)$	$(6,9)$	$(2,4)$	$(3,6)$	$(4,8)$
2/1	0.6212	0.6532	0.6294	0.3345	0.4556	0.4715
2/2	-	0.6563	0.6586	-	0.4557	0.4736
2/3	-	0.6563	0.6608	-	-	-
2/4	-	0.6564	0.6609	-	-	-
4/1	0.6581	0.6351	0.5822	0.4885	0.4911	0.4829
4/2	0.6645	0.6639	0.6510	0.4890	0.4956	0.4967
4/3	0.6647	0.6661	0.6643	-	0.4957	0.4975
4/4	-	0.6662	0.6662	-	-	-
4/5	-	-	0.6663	-	-	-
8/1	0.6479	0.6029	0.5364	0.4917	0.4825	0.4644
8/2	0.6643	0.6560	0.6271	0.4953	0.4974	0.4947
8/3	0.6658	0.6651	0.6570	0.4954	0.4983	0.4987
8/4	0.6659	0.6664	0.6645	-	0.4984	0.4991
8/5	-	-	0.6662	-	-	-
8/6	-	-	0.6664	-	-	-
8/7	-	-	0.6665	-	-	-
LDPC/1	0.5014	0.6611	0.6118	0.3348	0.4880	0.4943
LDPC/2	-	0.6655	0.6622	-	0.4881	0.4977
LDPC/3	-	0.6655	0.6664	-	-	-
LDPC/4	-	0.6655	0.6665	-	-	-
LDPC/5	-	0.6656	-	-	-	-

Table 2: Thresholds of SC ensembles.

trellis states is shown to yield better minimum distances and MAP thresholds but degraded BP thresholds. This degraded BP performance is avoided by applying spatial coupling to the underlying uncoupled ensembles. It can also be seen from the threshold results that for a regular LDPC ensemble with $d_v > 2$, there is an alternative CC-GLDPC ensemble, having a lower variable node degree than the LDPC ensemble, that has a better BP threshold and a similar or better MAP threshold than the LDPC ensemble.

with the node degrees. A complete analysis of the complexity/performance trade-off is beyond the scope of this work.

References

[1] A. Jiménez Feltström and K.Sh. Zigangirov, "Periodic time-varying convolutional codes with low-density parity-check matrices," *IEEE Trans. Inf. Theory*, vol. 45, pp. 2181–2190, Sep. 1999.

[2] M. Lentmaier, A. Sridharan, D.J. Costello, Jr., and K.Sh. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 56, pp. 5274–5289, Oct. 2010.

[3] S. Kudekar, T. Richardson, and R. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. Inf. Theory*, vol. 57, pp. 803 –834, Feb. 2011.

[4] A. Yedla, Y.-Y. Jian, P. Nguyen, and H. Pfister, "A simple proof of Maxwell saturation for coupled scalar recursions," *IEEE Trans. Inf. Theory*, vol. 60, pp. 6943–6965, Nov. 2014.

[5] S. Moloudi, M. Lentmaier, and A. G. i Amat, "Spatially coupled turbo-like codes," *IEEE Trans. Inf. Theory*, vol. 63, pp. 6199–6215, Oct. 2017.

[6] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Spatially coupled turbo-like codes: A new trade-off between waterfall and error floor," *IEEE Trans. Commun.*, vol. 67, pp. 3114–3123, May 2019.

[7] D. J. Costello, Jr., M. Lentmaier, and D. G. M. Mitchell, "New perspectives on braided convolutional codes," in *Proc. 9th Int. Symp. Turbo Codes and Iterative Inf. Processing (ISTC)*, pp. 400–405, Sept. 2016.

[8] Y. Liu, P. M. Olmos, and D. G. M. Mitchell, "Generalized LDPC codes for ultra reliable low latency communication in 5G and beyond," *IEEE Access*, vol. 6, pp. 72002–72014, 2018.

[9] S. Abu-Surra, D. Divsalar, and W. E. Ryan, "Enumerators for protograph-based ensembles of LDPC and generalized LDPC codes," *IEEE Trans. Inf. Theory*, vol. 57, pp. 858–886, Feb. 2011.

[10] R. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[11] I. E. Bocharova, B. D. Kudryashov, V. Skachek, and Y. Yakimenka, "Average spectra for ensembles of LDPC codes and applications," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, pp. 361–365, June 2017.

[12] R. J. McEliece, D. J. C. MacKay, and Jung-Fu Cheng, "Turbo decoding as an instance of Pearl's "belief propagation" algorithm," *IEEE J. Sel. Areas Commun.*, vol. 16, pp. 140–152, Feb. 1998.

[13] A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions: model and erasure channel properties," *IEEE Trans. Inf. Theory*, vol. 50, pp. 2657–2673, Nov 2004.

[14] C. Measson, A. Montanari, T. Richardson, and R. Urbanke, "The generalized area theorem and some of its consequences," *IEEE Trans. Inf. Theory*, vol. 55, pp. 4793–4821, Nov. 2009.

[15] M. Lentmaier, D.V. Truhachev, K.Sh. Zigangirov, and D.J. Costello, Jr., "An analysis of the block error probability performance of iterative decoding," *IEEE Trans. Inf. Theory*, vol. 51, pp. 3834–3855, Nov. 2005.

Paper V

Improving the Thresholds of Generalized LDPC Codes with Convolutional Code Constraints

We propose a generalized framework for the design of convolutional code (CC) based GLDPC codes. A novel construction method of class of CC-GLDPC codes is presented, which extends the construction method of CC-GLDPC codes discussed in [1]. We leverage the BP EXIT function technique, that was used for the LDPCs in [2], in the thresholds analysis to the proposed class of codes, and perform an exhaustive search using these thresholds in finding the CC-GLDPC codes with competitive thresholds. Results show that for low-rate CC-GLDPC codes, a trade-off between the BP and the MAP optimized CC-GLDPC codes offer a moderate BP decoding performance, with steep slopes in the waterfall region and no visible error-floor. For high-rate CC-GLDPC codes, the MAP optimized CC-GLDPC codes seem to be the preferred design choice. MAP optimized CC-GLDPC codes are in general observed to have structure matching to a regular graph, whereas BP optimized CC-GLDPC codes are composed of an irregular graph.

M. U. Farooq, A. G. i. Amat, and M. Lentmaier,
"Improving the Thresholds of Generalized LDPC Codes with Convolutional Code Constraints", *To be submitted at IEEE Communication Letters* in Nov 2022.

1 Introduction

A family of regular GLDPC codes with convolutional code (CC) constraints was introduced in [1]. These CC-GLDPC codes can be viewed as turbo-like codes with regular graph structure [3] and hence as an extension of braided convolutional codes to arbitrary regular graphs. The aim was to identify the performance trade-off between CC-GLDPC codes with strong component codes and lower variable node degree to conventional LDPC codes with larger variable node degree. The decoding thresholds and the minimum distances of CC-GLDPC codes and LDPC codes were compared. It was observed that sparser CC-GLDPC code graphs—graphs with lower variable node degrees—with a suitable trellis strength can perform better than the LDPC codes with similar graph structure. Furthermore, their performance was comparable to LDPC codes with denser graph structures in some cases.

It was observed that both the minimum distance of CC-GLDPC codes and their MAP thresholds improve with either the strength of the component code trellis or increasing the variable node degree of the graph at the expense of degraded BP thresholds. This type of performance behavior make CC-GLDPC codes suitable for spatial coupling (SC). Indeed, the SC-CC-GLDPC codes were observed numerically to exhibit threshold saturation phenomenon. Some practical matters regarding CC-GLDPC codes such as encoding, optimization of graph structures were still an open question.

In this paper, we extend the concept of CC-GLDPC codes to a much broader class of codes containing both irregular and regular graph structures for a given design rate. Some strategies to numerically search the CC-GLDPC codes optimized for BP and MAP performance are discussed, where the search is restricted to a sub-class of a broader class of codes. The search results show the potential of irregularity in the CC-GLDPC codes that are optimized for BP configurations. A search strategy tailored to leverage both the optimized BP and MAP performances is further explored, and its findings are discussed as well. Finally, the threshold analysis is validated through bit-error-rate (BER) simulations.

2 An Ensemble of Irregular GLDPC Codes with Convolutional Code Constraints

2.1 Compact Graph Representation

CC-GLDPC codes, analogously to protograph-based LDPC codes, can be described by means of a compact graph representation, as shown in Fig. 1(a) for a $(2, 4)$-regular ensemble [1]. In general, each variable node corresponds to a block of code symbols and each constraint node to a convolutional code $\mathcal{C}_j$ of rate $r_j = k_j/n_j$ and memory m_j, represented by a trellis of length N_j. The degree $d_{\mathrm{c},j}$ of a constraint node is equal to the number of code symbols per trellis section, i.e., $d_{\mathrm{c},j} = n_j$.

The value N_j is equivalent to the lifting factor of a protograph node, and the number of edges $n_{\mathrm{e},j} = d_{\mathrm{c},j} \cdot N_j$ in the lifted graph are equal to the total number of code symbols in the corresponding trellis. The permutations occur along the edges of the compact graph. A component code $\mathcal{C}_j$ of rate $r_j = k_j/n_j \geq 1/2$ can be constructed by puncturing a rate $r_{\mathrm{m}} = 1/2$ mother code trellis with $k_j N_j$

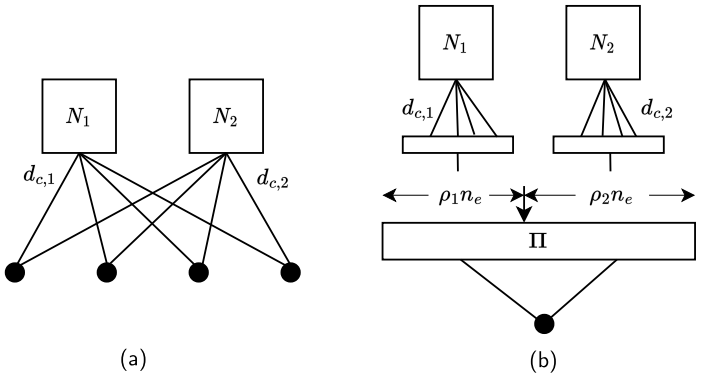


Figure 1: Compact graph: (a) (2,4) regular ensemble (b) proposed ensemble with adjustable constraint nodes.

sections. The factor graph of the mother code trellis is illustrated in Fig. 2(a), together with the corresponding constraint node. The desired rate is achieved by combining k_j sections to a rate $k_j/(2k_j)$ code and puncturing $2k_j - n_j$ of the parity bits. The punctured bits are shown as non-shaded variable nodes in Fig. 2(b).

Within this paper we assume that the number of constraint nodes is equal to two and all variable nodes have degree $d_{\mathrm{v}} = 2$, i.e., both information and parity symbols are protected twice by a trellis. The resulting overall codes have length $n = (n_{\mathrm{e},1} + n_{\mathrm{e},2})/2$. In order to improve the decoding thresholds we consider different types of irregularity at the constraint node side while keeping the variable nodes regular.

2.2 Irregular Component Code Memory

Under a preserved graph structure, in [1] the strength of the code ensemble was changed by varying the component code strength. Component code trellises with generator polynomials having trellises memories of one, two and three, respectively were used.

An irregular CC-GLDPC w.r.t. the component code trellis strength is obtained when $m_1 \neq m_2$, whereas a regular CC-GLDPC w.r.t. the component code trellis strength is obtained when $m_1 = m_2$. In this work, we have utilized the trellises with memories $m_j = 1, 2, 3$ having generator polynomials $(1, 1/3)$, $(1, 5/7)$, and $(1, 13/15)$. A strong feature of trellis based GLDPC codes is that the strength of the component codes can be changed without altering the graph structure, or equivalently the design rate r_D of the graph.

2.3 Irregular Distribution of Edges

Now consider the case that $d_{\mathrm{c}} = d_{\mathrm{c},1} = d_{\mathrm{c},2}$, $N_1 \neq N_2$, and $m_1 \neq m_2$. In this case, both the strengths and the proportions of the component codes $\mathcal{C}_j$ become irregular, but their rates are kept regular. Let us introduce ρ_j, which represents the fraction of the total number of edges that are connected to $\mathcal{C}_j$, which can

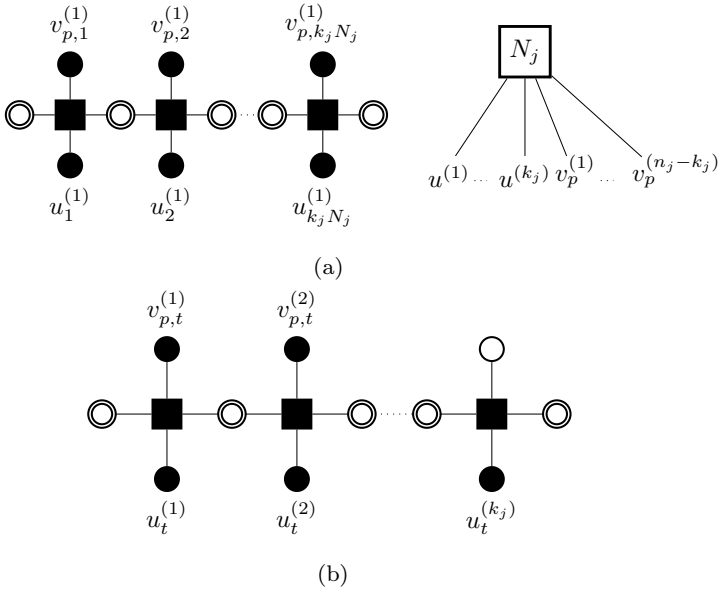


Figure 2: Factor graph representation of a constraint node trellis (a) Rate 1/2 trellis (left) in a constraint node (right). (b) Trellis section at time t after puncturing to rate k_j/n_j.

be expressed as

$$\rho_1 = \rho = \frac{d_{\mathrm{c},1} \cdot N_1}{d_{\mathrm{c},1} \cdot N_1 + d_{\mathrm{c},2} \cdot N_2} \ , \quad \rho_2 = 1 - \rho \ . \tag{1}$$

A varying ρ produces a family of codes with a varying proportion of $\mathcal{C}_j$, under a fixed $R = 1 - 2/d_{\mathrm{c}}$, and fixed component code rates $r = r_1 = r_2$.

2.4 Irregular Component Code Rates

For a given design rate R, an irregularity w.r.t. the component code rates pair $\boldsymbol{r} = (r_1, r_2)$ is achieved if $r_1 \neq r_2$, whereas regularity w.r.t. the rates is achieved if $r_1 = r_2$. Regular or irregular component code rates r_j can be combined with regular or irregular memories m_j, and regular or irregular lifting factors N_j under the constraint that the design rate R remains fixed. The design rate in terms of component code rates pair $\boldsymbol{r} = (r_1, r_2)$, is expressed as

$$\begin{aligned} R(\rho, \boldsymbol{r}) &= 1 - d_v(1 - r_c(\rho, \boldsymbol{r})) \\ &= 2r_c(\rho, \boldsymbol{r}) - 1 \ , \end{aligned} \tag{2}$$

where r_c denotes the average component code rate

$$r_c(\rho, \boldsymbol{r}) = \rho\, r_1 + (1 - \rho)\, r_2 \ . \tag{3}$$

For a given r_1 and a fixed design rate R we can then express r_2 as a function of ρ, namely

$$r_2(\rho) = \frac{r_c - \rho \cdot r_1}{(1 - \rho)} = \frac{(R+1)/2 - \rho \cdot r_1}{(1 - \rho)} \ . \tag{4}$$

We remark that both r_1 and r_2 are rational fractions, and lie within the interval $[r_m, 1)$, where $r_m = 1/2$ is the rate of the mother code trellis used in this work. For a given design rate R or an averaged component code rate r_c, a family of CC-GLDPC codes with irregular component codes is obtained for $\rho \in [0, 1]$.

3 Threshold Analysis and Optimization

In this section, we discuss the exhaustive grid search employed in the CC-GLDPC code design space to determine the CC-GLDPC codes with competitive decoding thresholds. The search uses the DE and MAP decoding thresholds of the considered code space on the BEC. A more efficient and reliable BP and MAP thresholds computation method than the DE analysis method that was used in [1] is discussed. Furthermore, the derivation of the decoder transfer function of a punctured trellis is also described. In the end, the findings of the exhaustive search are discussed.

3.1 Density Evolution Equations

The mother code transfer functions of considered generator polynomials can be derived by following the method explained in [3]. Suppose f_s and f_p denote the decoder transfer functions of systematic and parity bits of the mother code, then

$$p_\mathrm{s}^{(i)} = f_\mathrm{s}(q_\mathrm{s}^{(i)}, q_\mathrm{p}^{(i)}), \tag{5}$$

$$p_\mathrm{p}^{(i)} = f_\mathrm{p}(q_\mathrm{s}^{(i)}, q_\mathrm{p}^{(i)}), \tag{6}$$

where $p_\mathrm{s}^{(i)}$ and $p_\mathrm{p}^{(i)}$ are the erasure probabilities of the outgoing extrinsic message from the CN, and the $q_\mathrm{s}^{(i)}$ and $q_\mathrm{p}^{(i)}$ are the erasure probabilities of the incoming extrinsic messages to the CN. If the parity bits of the mother code trellis are punctured randomly with a uniform distribution to achieve the desired component code rate, then the incoming erasure probabilities to the transfer functions become

$$q_\mathrm{s}^{(i)} = q^{(i-1)}, \tag{7}$$

$$q_\mathrm{p}^{(i)} = \frac{2k_j - n_j}{k_j} \cdot 1 + \frac{n_j - k_j}{k_j} q^{(i-1)} \ . \tag{8}$$

After the decoding, the erasure probability of the averaged extrinsic message from $\mathcal{C}_j$ is computed using

$$p_j^{(i)} = \frac{k_j}{n_j} \cdot p_\mathrm{s}^{(i)} + \frac{n_j - k_j}{n_j} \cdot p_\mathrm{p}^{(i)} \ . \tag{9}$$

Combining the contributions of both decoders we get

$$p^{(i)} = \rho_1 p_1^{(i)} + \rho_2 p_2^{(i)} = f_\mathrm{T}(q^{(i-1)}) \ , \tag{10}$$

where the transfer function $f_\mathrm{T}(\cdot)$ follows from applying equations (5)–(9). Finally, including the variable node update the DE recursion can be expressed as

$$q^{(i)} = \varepsilon \cdot f_\mathrm{T}(q^{(i-1)}) \ . \tag{11}$$

The transfer function of the punctured trellis in (9) is applicable for a rate range of $\big(1/n^{(j)}, \cdots, (n^{(j)} - 1)/n^{(j)}\big)$.

3.2 Computation of BP and MAP Thresholds

We use the BP-EXIT function $h^{\mathrm{BP}}(\varepsilon)$ [4] to determine the BP and MAP thresholds of CC-GLDPC codes, which in parameterized form is expressed as

$$h^{\mathrm{BP}}(\varepsilon) = \begin{cases} (\varepsilon, 0) & \varepsilon \in [0, \varepsilon^{\mathrm{BP}}) \\ (\varepsilon(x), f_{\mathrm{T}}^2(x)) & \varepsilon \in (\varepsilon^{\mathrm{BP}}, 1] \leftrightarrow x \in (x^{\mathrm{BP}}, 1] \end{cases}$$

since we restrict the variable node degree to $d_{\mathrm{v}} = 2$ in this work. Note that x^{BP} corresponds to value of x for which $\varepsilon(x)$ is minimum.

The MAP threshold is determined from the BP-EXIT function as

$$R(\rho, \boldsymbol{r}) = \int_{\varepsilon^{\mathrm{MAP}}}^{1} h^{\mathrm{BP}}(\varepsilon)\, d\varepsilon \ .$$

3.3 Threshold Optimization

A CC-GLDPC code design space is constructed by choosing the component codes: rates pair $\boldsymbol{r}$, trellis memory pair $\boldsymbol{m}$, degrees pair $\boldsymbol{d}_c$ or equivalently the fraction pair $(\rho, 1-\rho)$. The resulting code space is infinite dimensional along component code rates pair $\boldsymbol{r}$, and fractions pair $(\rho, 1-\rho)$. We remark that the possible choices of component code rates, and the fraction ρ are interlinked via (3), which must be satisfied for the codes belonging to the code space.

The restriction to a code graph with two CNs and degree two VNs results in a symmetry in the component code rates. This symmetry allows us to reduce the dimensions of the code space to r_1, ρ and $\boldsymbol{m}$ instead. Note that r_2 is computed from (4) for a given $\rho \in [0, 1]$ and $r_1 \in [r_c, 1)$. The graph symmetry in component code rate implies that for a fixed ρ, when r_2 lies within $[r_m, r_c]$, then the resultant r_1 lies within $[r_c, 1)$. This holds even when the rate ranges described above are swapped.

A significant reduction of the code space comes from the discretization. The discretization of the code space is linked to the length of the sequence $\rho = (2/n, \cdots, (n-2)/n)$. For $n \to \infty$, the length of the sequence, and possible choices of component code rates approaches infinity. This corresponds to infinite dimensional code space. A discretized code space is obtained by limiting the number of edges n in a code graph.

A reduced code space—characterized as $r_1(\rho, \boldsymbol{m}, r_2)$ for $\rho = (2/n, \cdots, (n-2)/n)$ with a finite n, $r_2 \in [r_m, r_c]$, and all trellises strengths pairs $\boldsymbol{m}$ —become feasible for an exhaustive grid search using the BP and MAP thresholds on the BEC. With a fine enough resolution in reduced space, the findings of the exhaustive search may be regarded as reliable.

3.4 Results and Discussion

Thresholds of CC-GLDPC codes having regular graph structure $d_{\mathrm{c},1} = d_{\mathrm{c},2}$, $N_1 = N_2$, and both regular and irregular trellis strengths are shown in Table 1 (left). It can be observed that irregularity w.r.t. trellis strength alone, while keeping the component codes degrees, and rates regular, doesn't offer performance improvement compared to the fully regular CC-GLDPC codes.

Figure 3 shows how an irregular distribution of edges, characterized by the parameter ρ, influences the thresholds. Choosing the best parameter $\rho = \rho^*$

(m_1, m_2),	ε^{BP}	ε^{MAP}	ρ^*	ε^{BP}
$(1,1)$	0.3334	0.3361	-	-
$(2,1)$	0.4423	0.4570	103/128	0.4453
$(3,1)$	0.4396	0.4673	56/128	0.4399
$(2,2)$	0.4429	0.4889	-	-
$(3,2)$	0.4339	0.4926	0	0.4429
$(3,3)$	0.4249	0.4955	-	-

Table 1: Thresholds of $R = 1/2$ CC-GLDPC codes with irregular trellis strength.

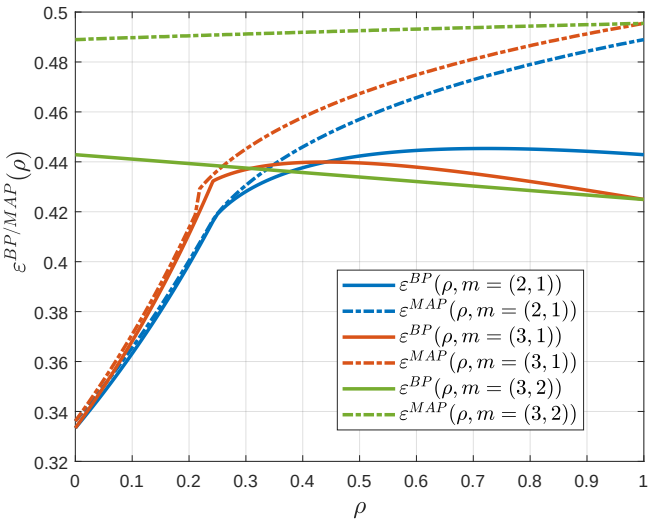


Figure 3: BP and MAP thresholds vs ρ of rate $R = 1/2$ CC-GLDPC codes with irregular trellis strength.

results in the values shown in Table 1 (right). Now the BP threshold can be improved compared to the fully regular case.

Now we compute BP and MAP thresholds of rate $R = 1/3, 1/2, 2/3$ CC-GLDPC codes over the complete search space, including irregular component code rates. For a given trellis memory pair $\boldsymbol{m}$, the exhaustive search produces a family of curves $\varepsilon^{\mathrm{BP}}$ as a function of $r_1(\rho)$ for each ρ. The design configuration with the best BP decoding performance then corresponds to the largest BP threshold in the family of curves $\varepsilon^{\mathrm{BP}}(r_1(\rho, \boldsymbol{m}))$.

Suppose ρ^* denotes the fraction that produces the best BP threshold in the reduced code space. The plot of $\varepsilon^{\mathrm{BP}}$ as a function of $r_1(\rho = \rho^*, \boldsymbol{m})$ for $r_D = 1/2$ CC-GLDPC is shown in Fig. 4. The plot is obtained by projecting the curves $\varepsilon^{\mathrm{BP}}$ of all trellis memory combinations into a single dimension. It can be seen from the figure that $\rho^* = 0.5212$, $\boldsymbol{m} = (3,2)$, and $r_1 = 0.6554$ constitute the design parameters corresponding to the best BP threshold of rate $R = 1/2$ CC-

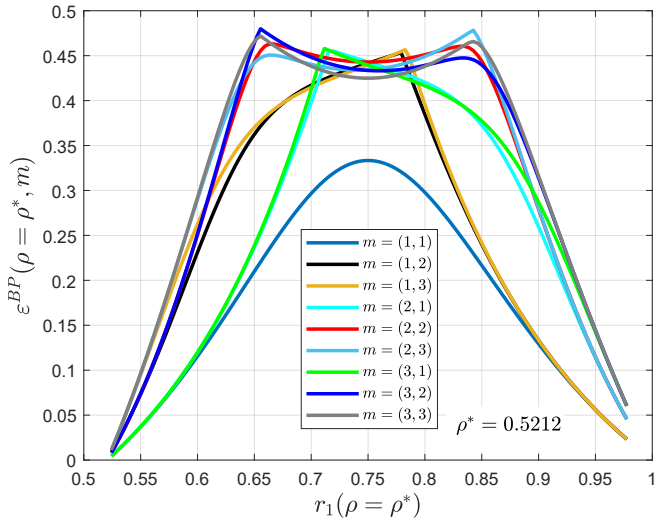


Figure 4: BP Threshold vs rate at $\rho*$ of Rate 1/2 CC-GLDPC codes with irregular component code rates.

GLDPC codes. Similarly, we obtain the design configuration with best MAP decoding performance.

The design parameters corresponding to best BP and best MAP thresholds for $R = 1/3, 1/2, 2/3$ are listed in Table 2. It is observed that the irregular CC-GLDPC graph structures yield the best BP and MAP decoding thresholds. However, the MAP decoding performance of the regular code ensembles is very similar to that of the irregular ones, also visible in Table 2.

4 Finite Length Performance of CC-GLDPC codes

We perform BER simulations of length $n = 102400$ CC-GLDPC codes with the design configurations corresponding to the best thresholds w.r.t. the BP and MAP decoding performance listed in Table 2. Some strategies leading to a trade-off between the BP thresholds and error floors are also discussed. The simulations are compared with AWGN channel thresholds. Table 3 shows AWGN channel thresholds computed by the erasure channel prediction method used in [5], while the thresholds in Table 4 are obtained by Monte Carlo DE based on histograms.

The threshold computation of CC-GLDPC codes in this work assumes an unstructured code graph. The simulation of such unstructured CC-GLDPC codes results in a BER performance with high error floor. This is circumvented by introducing a semi-structured graph. An edge type is distinguished by the CN to which an edge in the graph is attached to. In this way, a type-1 edge from a VN is connected to the first CN, and a type-2 edge is connected the second CN in the code graph. The number of variables nodes or block length $n = n_e/2$ in the graph. In the semi-structured ensembles construction, we first

Parameter	Best BP	MAP (r)	Best MAP (ir)
Rate-1/3 CC-GLDPC			
ε^{BP}	0.6441	0.5352	0.5356
ε^{MAP}	0.6603	0.6659	0.6660
$(\rho^*, 1-\rho^*)$	(0.4773,0.5227)	(0.5,0.5)	(0.3295,0.6705)
$(r^{(1)}, r^{(2)})$	(0.5159,0.8043)	(0.6667,0.6667)	(0.6552,0.6723)
$(m^{(1)}, m^{(2)})$	(2,2)	(3,3)	(3,3)
Rate-1/2 CC-GLDPC			
ε^{BP}	0.4799	0.4249	0.4250
ε^{MAP}	0.4858	0.4955	0.4955
$(\rho^*, 1-\rho^*)$	(0.5215,0.4785)	(0.5,0.5)	(0.2266,0.7734)
$(r^{(1)}, r^{(2)})$	(0.6554,0.8531)	(0.75,0.75)	(0.7414,0.7525)
$(m^{(1)}, m^{(2)})$	(3,2)	(3,3)	(3,3)
Rate-2/3 CC-GLDPC			
ε^{BP}	0.3041	0.2893	0.2893
ε^{MAP}	0.3093	0.3189	0.3189
$(\rho^*, 1-\rho^*)$	(0.8274,0.1726)	(0.5,0.5)	(0.1726,0.8274)
$(r^{(1)}, r^{(2)})$	(0.8082,0.9540)	(0.8333,0.8333)	(0.8276,0.8345)
$(m^{(1)}, m^{(2)})$	(3,3)	(3,3)	(3,3)

Table 2: Design Parameters of Best Thresholds

r_D	Parameter	Best BP	Regular	Best MAP (ir)
1/3	ε^{BP}	-0.1261	1.4747	1.4692
1/3	ε^{MAP}	-0.3893	-0.4824	-0.4841
1/2	ε^{BP}	0.4511	1.1565	1.1548
1/2	ε^{MAP}	0.3740	0.2467	0.2467
2/3	ε^{BP}	1.4313	1.6216	1.6216
2/3	ε^{MAP}	1.3648	1.2426	1.2426

Table 3: Predicted Thresholds $E_b/N_0(\rho^*)$-dB of CC-GLDPC

combine the n symbols with their permuted version through a length-n pseudo-random permutor Π_s with spreading factor s. Next, the first $n_{\mathrm{e},1}$ symbols of the combined sequence are connected to $\mathcal{C}_1$. The remaining $n_{\mathrm{e},1}$ symbols are connected to the second $\mathcal{C}_2$.

BER simulations of rates $R = 1/3, 1/2, 2/3$ semi-structured CC-GLDPC codes with design configurations in Table 2 are shown in Fig. 5. The AWGN channel thresholds shown in the figure correspond to Table 3. In general, the BP optimized CC-GLDPC codes BER plots of the considered ensembles show a larger gap between the waterfall performance and the capacity. Furthermore,

r_D	Parameter	Best BP	Best MAP (ir)
1/3	$(E_b/N_0)^{BP}$	-0.0046	1.2386
1/2	$(E_b/N_0)^{BP}$	0.9953	1.0616
2/3	$(E_b/N_0)^{BP}$	1.6037	1.6516

Table 4: MC-DE-H Thresholds $E_b/N_0(\rho^*)$-dB of CC-GLDPC

high-rate BP optimized CC-GLDPC codes exhibit an error floor in the performance, whereas no visible error floor is present for the rate $R = 1/3$ BP optimized CC-GLDPC codes down to a BER of 10^{-7}.

MAP optimized CC-GLDPC codes on the other hand reveal an excellent waterfall performance with steep slopes in the waterfall region. Additionally, no error floor is observed in the performance of these ensembles down to the BER of 10^{-7}. Low-rate MAP optimized CC-GLDPC codes, however, show a relatively poor BP decoding performance due to their weak BP thresholds. The BP threshold of MAP optimized CC-GLDPC codes is observed to improve with increased code rate. This threshold improvement is observed by measuring the gap in the sub-optimal decoding performance of the BP and MAP optimized CC-GLDPC codes. The gap is observed to be the largest for $R = 1/3$, whereas it is smallest for $R = 2/3$. Such an improvement is observed to be consistent in the BER performance as well.

For ensembles with trellis strengths $(3, 3)$ in Table 2, the decoding complexity can be interpreted in terms of total length of the trellises $k_1N_1 + k_2N_2$, and the number of decoding iterations. In this work, maximum number of iterations are restricted to 100 during the performance simulations. It is observed that the BP optimized CC-GLDPC codes require larger number of iterations compared to the MAP optimized CC-GLDPC codes. For instance, for MAP optimized ensembles, 10 decoding iterations were observed to be sufficient to achieve an adequate performance, whereas for the BP optimized ensembles the required number of iterations were observed to be fluctuating around 50 to 60.

4.1 Concluding Remarks

We have introduced different types of irregularity at the constraint nodes of CC-GLDPC codes in order to improve their BP and MAP decoding thresholds. The proposed ensemble can be flexibly tuned in terms of trellis memory, fraction of edges connected to the different constraint nodes, and the component code rates. An exhaustive grid search was then performed over these three dimensions to optimize the thresholds. Our results show that the BP thresholds can be improved compared to the turbo-like code ensembles considered in [3].

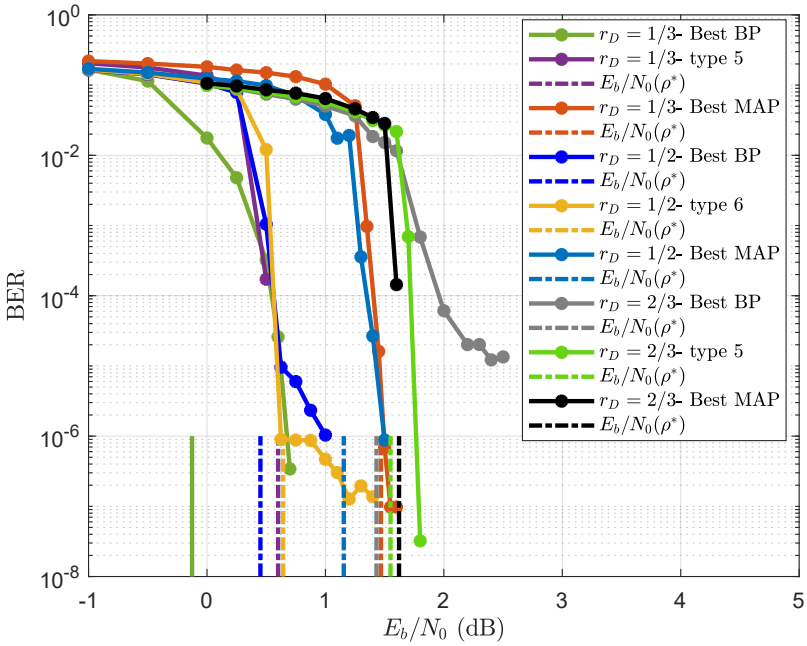


Figure 5: BER simulations.

References

[1] M. U. Farooq, S. Moloudi, and a. M. Lentmaier, "Generalized ldpc codes with convolutional code constraints," in *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 479–484, 2020.

[2] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, pp. 619–637, Feb. 2001.

[3] S. Moloudi, M. Lentmaier, and A. G. i Amat, "Spatially coupled turbo-like codes," *IEEE Trans. Inf. Theory*, vol. 63, pp. 6199–6215, Oct. 2017.

[4] T. Richardson and R. Urbanke, *Modern Coding Theory*. USA: Cambridge University Press, 2008.

[5] M. U. Farooq, S. Moloudi, and M. Lentmaier, "Thresholds of braided convolutional codes on the awgn channel," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1375–1379, 2018.