UNIVERSITY OF CALIFORNIA
Los Angeles

Efficient List Decoding for Short Blocklength Communication

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Electrical and Computer Engineering

by

Brendan Hisao Towell

2024

ABSTRACT OF THE THESIS

Efficient List Decoding for Short Blocklength Communication

by

Brendan Hisao Towell
Master of Science in Electrical and Computer Engineering
University of California, Los Angeles, 2024
Professor Richard D. Wesel, Chair

At short blocklengths, well designed zero-terminated (ZT) and tail-biting (TB) convolutional codes (CCs) concatenated with cyclic redundancy check (CRC) codes have been shown to closely approach the random coding union (RCU) bound for both low rate (rate-$1/n$) and high rate (rate-$(n-1)/n$) codes. The CRC acts as an outer error detection code, verifying that he codeword has been successfully received and decoded, while the CC acts as an inner error correction code, combating channel errors. Maximum likelihood (ML) decoding of such a code can be performed by the serial list Viterbi algorithm (S-LVA), which checks codewords of the inner CC in order of increasing distance from the received word, and returns the first inner codeword that also passes the CRC. Implementation of the S-LVA for low rate CCs can be done efficiently on the standard Viterbi trellis, while using the dual trellis for high rate CCs can offer significant performance improvements.

For some rates, it may be the case that no CRC is available. In that case, we consider a generalization called an expurgating linear function (ELF), which doesn't enforce the cyclic condition, but similarly serves the function of expurgating low weight codewords, improving the performance of the concatenated code. Both ELFs and CRCs that offer a good distance spectrum metric can be efficiently identified by the list decoding sieve method.

Sometimes, it may be desirable to sacrifice ML performance for improved decoding complexity. In such cases, it is possible to leverage the linear nature of CC-CRCs, and in particular, TBCC-CRCs, to reduce the decoding complexity with a small loss to frame error rate (FER) performance. Much of the cost associated with S-LVA of TBCC-CRCs is due to tracebacks

of paths that don't end up meeting the TB condition. Due to the linear nature of the code, we can precompute offsets from any trellis path with a particular ending state difference (ESD), which we can use to quickly search nearby codewords that are guaranteed to meet the TB condition. This has been shown empirically to significantly improve the expected list size required for decoding.

The thesis of Brendan Hisao Towell is approved.

Ian Roberts

Lara Dolecek

Richard D. Wesel, Committee Chair

University of California, Los Angeles

2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank Richard Wesel, for his continued guidance through my time at UCLA, both as my professor and PI. I would also like to thank Beryl Sui, for being an excellent collaborator along my research journey. This research was supported by National Science Foundation (NSF) grant CCF-2008918. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect views of NSF.

CHAPTER 1

Introduction

1.1 Organization

We will begin this thesis with a brief overview of relevant background information, following *Error Control Coding* by Lin and Costello [2]. We will draw primarily from chapters 1, 11, and 12. Thesis chapters 2, 3, 4 are each drawn from a published work I've contributed to, namely [3], [4], and [5], respectively. In chapter 2, we will look at list decoding for high rate convolutional codes (CCs) concatenated with cyclic redundancy check codes (CRCs). In chapter 3, we will look at extending the concept of using a CC concatenated with a CRC to a more general expurgating linear function (ELF), which may not be cyclic. In chapter 4, we will look at how the linear property of such concatenated codes allow for a reduction in decoding complexity with minimal loss of FER performance. Chapter 5 concludes this thesis.

1.2 My Work

Much of my research work has been done in collaboration with other researchers from my lab, including Beryl Sui and Professor Wesel. To ensure my contributions are fairly represented, I will be including a short note at the beginning of each chapter outlining my personal contributions.

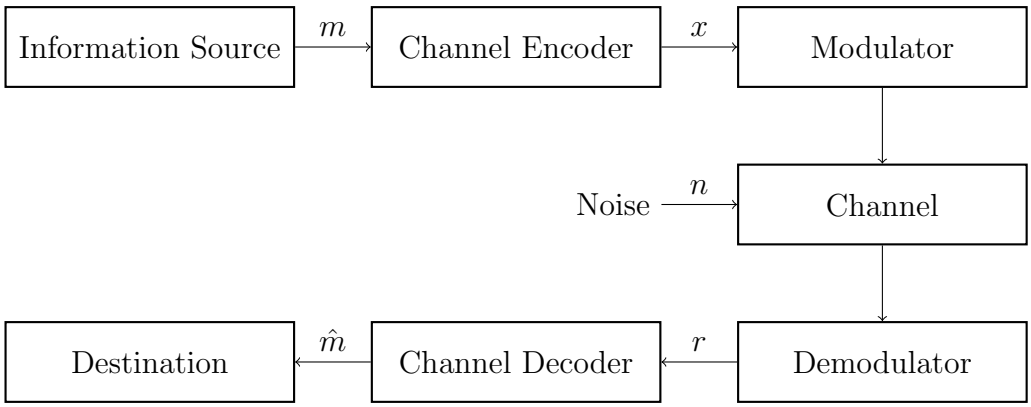

Figure 1.1: Basic transmission system block diagram

1.3 Channel Coding

Efficient and reliable data transmission over noisy channels has become increasingly important in the digital age. The components of a basic transmission system can be represented by a block diagram, as see in Fig. 1.1. The information source provides a binary message m, which is transformed by the encoder into an encoded sequence x, called a codeword. In our case, x will be binary. The modulator takes the codeword and converts it to a waveform to be transmitted over the channel, where the demodulator converts it back to a received codeword $r = x + n$, the sum of the original codeword and the noise introduced by the channel. Finally, the decoder resolves r to the estimated information sequence $\hat{m}$, which ideally is equivalent to the original m, and passes $\hat{m}$ to the destination, completing the transmission. In the case that $\hat{m} \neq m$, we say that a codeword error (equivalently, frame error) has occurred. A common metric for performance, and the one primarily used in this thesis, is the codeword error rate (CER) or equivalently, frame error rate (FER), the number of frame errors divided by the total number of transmissions. Throughout this thesis, we consider binary phase shift keying as the method of modulation, so the modulator takes $x_i \in \{0, 1\}$ and transmits $\pm E_b$, where E_b is the energy per bit. The noise is additive white Gaussian noise (AWGN), and is normally distributed.

There are several useful parameters to define in relation to Fig. 1.1. One is the block length n, which is the length of x, and another is the number of message bits k, which is

the length of m. These combine to give the code rate $R = k/n$. Since adding redundancy requires adding additional information, we are concerned with the cases where $k < n$, so $R < 1$. So, a code will consist of 2^m possible messages, corresponding to the same number of unique codewords, out of the 2^n possible n bit long messages.

When decoding, it makes intuitive sense that we would want to choose $\hat{m}$ such that $P(\hat{m} = m|r)$ is maximized. Making such a decision is known as maximum likelihood (ML) decoding. While this is generally true, it may be the case that ML decoding is more computationally intensive that desired, or even may be practically intractable. In such cases, sub-ML decoding can be a practical solution to reduce complexity, and depending on the algorithm, may not even substantially harm FER performance.

1.4 Convolutional Codes

1.4.1 Background and Encoding

Convolutional codes (CCs) are a class of codes that use memory elements in the encoding process to add redundancy. Typically, the number of memory elements is represented as v. A basic convolutional encoder can be seen in figure 1.2, where x_1 and x_2 are the output bits. The boxes represent the memory elements, and act as shift registers, and the circles are XORs. For the sake of example, consider we have a message string 10, and the memory elements start in the 0 state. Then the first input bit is a 1, so both output bits are 1, and the first shift register stores that 1. The second input bit is a 0, so considering the value of the memory elements, $x_1 = 1$ and $x_2 = 0$, and the 1 shifts to the second memory element, and the 0 is stored in the first memory element. Since we have two output bits for every input bit, this is a rate-1/2 encoder, and since we have two memory elements, $v = 2$.

Convolutional encoders can be viewed as state machines, where the state is given by the concatenation of the value stored in the memory elements, and edges corresponding to pairs of inputs and outputs, as illustrated in Fig. 1.3. The labels of the edges correspond to input/output pairs, and the states are the concatenation of the memory elements, with

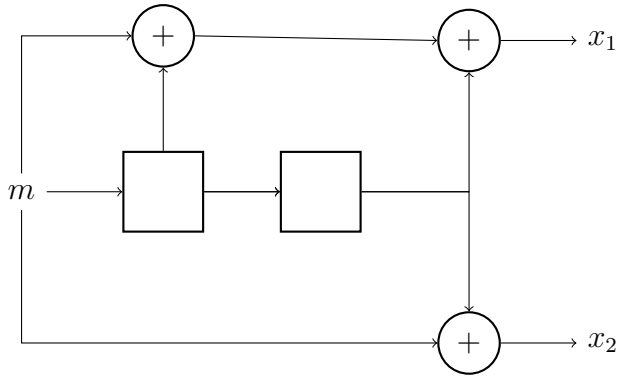

Figure 1.2: A basic convolutional encoder

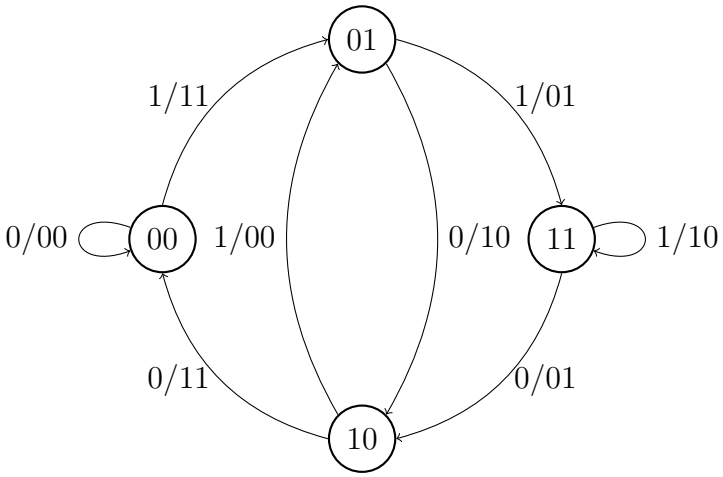

Figure 1.3: State machine view of the encoder in Fig. 1.2

the least significant bit corresponding to the first memory element. This view can be helpful to understand the encoding process, but it is more relevant to the discussion of decoding, which we will get to shortly.

Before we get to decoding, we'll briefly discuss some general properties of CCs, and the zero-terminating (ZT) and tail-biting (TB) conditions. First, it is worth noting that CCs sometimes use k and n to refer to the number of input and output bits instead of the message length and blocklength, so in this case, k would be 1 and n would be 2. The tuple (n, k, v) is often used to describe CCs, and would be $(2, 1, 2)$ in this case from 1.2. This definition of n allows us to split CCs into two classes based on rate- high rate codes, which have rate-$(n-1)/n$, and low rate codes, which have rate-$1/n$, which will be decoded slightly differently,

as will be shown in Ch. 2.

CCs generally offer strong performance at short blocklengths, since increasing the number of memory elements v improves performance, although it comes at the cost of decoding complexity. Many other types of codes require longer blocklengths before they offer substantial protection. However, they tend to perform worse at longer blocklengths, since the minimum distance d_{min} of convolutional codes doesn't change with increasing blocklength. This can be understood from the state machine view of CCs: any two paths through the state machine that diverge at one point and later merge are both valid codewords, which doesn't change as the blocklength changes. Since detours could be relatively small, at longer blocklengths this can significantly inhibit performance.

As our final note before we move onto decoding, we'll discuss the ZT and TB conditions. Naively encoding a message with a convolutional encoder with no termination condition results in uneven protection, namely for the ending bits. Using the ZT condition, the encoder starts in the all-zeros state, and after transmitting all the message bits, transmits additional bits until the encoder returns to the all-zeros state, fully protecting the last bits, however, this does incur a small rate penalty. Using the TB condition, the encoder chooses its starting state such that the starting state and ending state are the same, similarly smoothing protection, and without incurring a rate penalty, but at the cost of more complex decoding.

1.4.2 Decoding

As previously mentioned, decoding CCs is where the state machine view of CCs is most useful, and is extended to the concept of a trellis, as is shown in Fig. 1.4. Edges corresponding to a 0 input are shown as solid, and edges corresponding to a 1 input are dashed. In this diagram, we assumed that we start in the zero state, as would be the case for ZTCCs, and the trellis would eventually converge back to the zero state, but has been truncated for compactness. We can see directly how a path around the state machine corresponds to a path through the trellis.

While there are multiple decoding algorithms for CCs, we will be focusing on the Viterbi

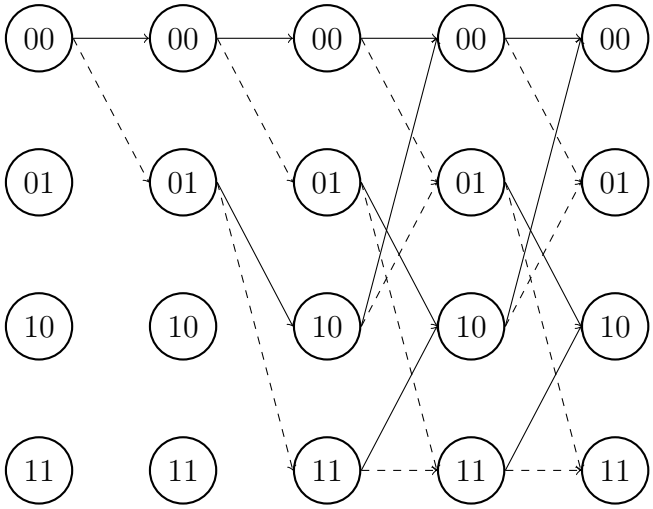

Figure 1.4: Trellis diagram for the encoder in Fig. 1.2

algorithm, a popular ML decoding algorithm. The first step is to do a forward pass. In the forward pass, we initialize the starting node to have a weight of zero. Then, we consider the paths going forward, in this case to the 00 and 01 state, and save in each of those nodes the weight associated with taking the branch plus the weight stored in the source node, and the state from which the branch originated. We continue this way until we have multiple incoming branches to a single node. At that point, we only have to save the best weight and source node, and can discard the other. This is the key insight that allows the Viterbi algorithm to decode CCs efficiently. It may not immediately be obvious why the other node can be discarded, but consider that the rest of the trellis, from that point forward, is identical, so both paths going forward will accrue the same additional weight. Thus, if one has a higher weight at this point, it can never be the minimal weight path, and cannot be the ML decision. This means we only ever have to keep track of a maximum of 2^v paths at any stage.

Once the forward pass it complete, decoding is simple. Since each node knows the source node from whence it came, we can simply start at the end of the trellis, and perform a traceback, taking the optimal source node at each turn, which will yield the optimal path through the trellis. This has a one to one correspondance with the optimal codeword. For TBCCs, ML decoding is slightly more complex, but the fundamental idea is the same, and

will be explored in Ch. 2. In the next section, we will consider cyclic redundancy check codes (CRCs), and concatenating codes.

1.5 CRCs and Concatenation

CRCs are a type of error detecting code that is often used to verify that a transmission has been received successfully. They do this by, on the encoding side, appending bits to ensure that the CRC-augmented message is divisible by the CRC, and on the receiver side, ensuring that the divisibility condition is still met. The degree of a CRC, which is the same as the number of bits that are appended, is often denoted m. CRCs cannot correct errors on their own in general, since receiving a codeword that is not divisible by the CRC offers no information about where the error might have occurred.

The coding scheme we explore throughout this thesis involves CCs concatenated with CRCs, as seen in Fig. 1.5. Unlike standard Viterbi decoding, since we're dealing with a CC that has many expurgated codewords, it is possible that the first codeword we find when decoding the CC does not pass the CRC. Using the serial list Viterbi algorithm (SLVA), we can explore CC codewords of increasing weight until we find one that passes the CRC, or until we exceed the maximum number of allowed iterations, at which point we declare a decoding failure. When concatenated with a CC, the the CRC expurgates codewords from the outer CC, and by choosing a CRC that expurgates low weight codewords, we can increase the minimum distance of the concatenated code, giving strong performance. The specifics of SLVA will be explored through the rest of this thesis.

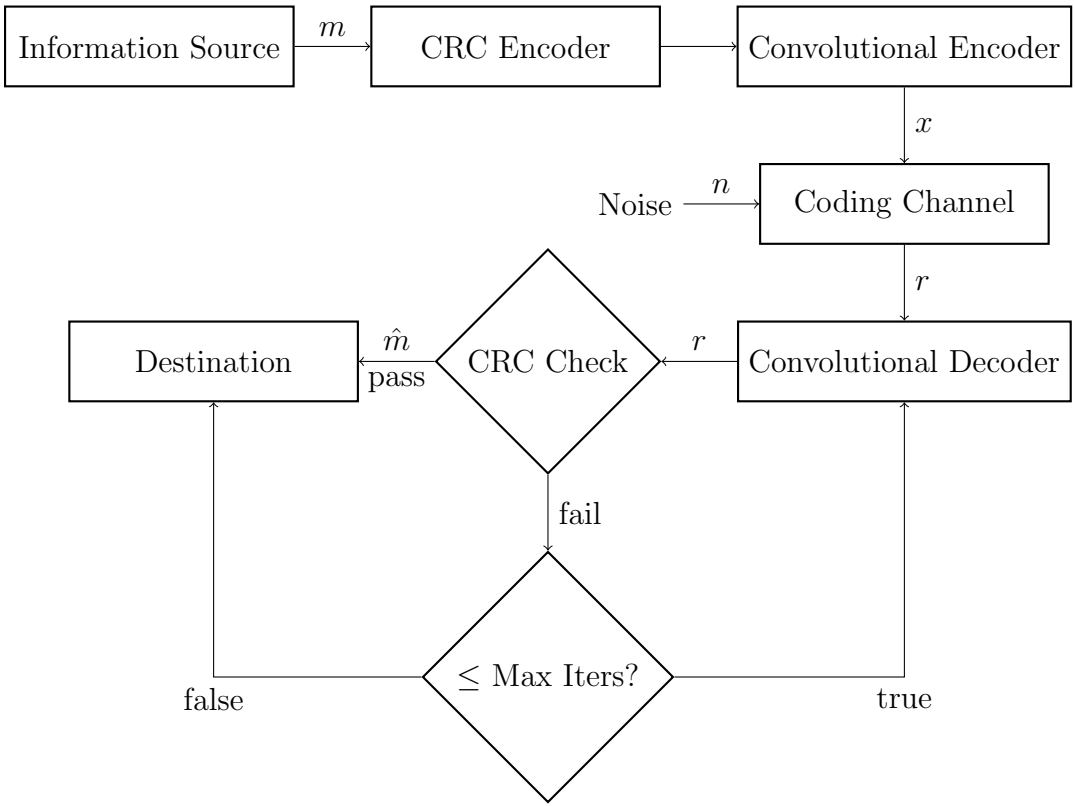

Figure 1.5: Block diagram of a transmission system using a CC concatenated with a CRC

CHAPTER 2

On CRC-Aided, Dual-Trellis, List Decoding for High-Rate Convolutional Codes with Short Blocklengths

2.1 Abstract

Recently, rate-$1/n$ zero-terminated and tail-biting convolutional codes (ZTCCs and TBCCs) with cyclic-redundancy-check (CRC)-aided list decoding have been shown to closely approach the random-coding union (RCU) bound for short blocklengths. This chapter designs CRCs for rate-$(n-1)/n$ CCs with short blocklengths, considering both the ZT and TB cases. The CRC design seeks to optimize the frame error rate (FER) performance of the code resulting from the concatenation of the CRC and the CC. Utilization of the dual trellis proposed by Yamada *et al.* [6] lowers the complexity of CRC-aided serial list Viterbi decoding (SLVD) of ZTCCs and TBCCs. CRC-aided SLVD of the TBCCs closely approaches the RCU bound at a blocklength of 128. This chapter also explores the complexity-performance trade-off for three decoders: a multi-trellis approach, a single-trellis approach, and a modified single trellis approach with pre-processing using the Wrap Around Viterbi Algorithm (WAVA).

2.2 Introduction

The structure of concatenating a convolutional code (CC) with a CRC code has been a popular paradigm since 1994 when it was proposed in the context of hybrid automatic repeat request (ARQ) [7]. It was subsequently adopted in the cellular communication standards of

both 3G [8] and 4G LTE [9]. In general, the CRC code serves as an outer error-detecting code that verifies if a codeword has been correctly received, whereas the CC serves as an inner error-correcting code to combat channel errors.

Recently, there has been a renewed interest in designing powerful short blocklength codes. This renewed interest is mainly driven by the development of finite blocklength information theory by Polyanskiy *et al.*, [10] and the stringent requirement of ultra-reliable low-latency communication (URLLC) for mission-critical IoT (Internet of Things) service [11]. In [10], Polyanskiy *et al.* developed a new achievability bound known as the RCU bound and a new converse bound, known as the meta-converse (MC) bound. Together, these two bounds characterize the error probability range for the best short blocklength code of length N with M codewords. The URLLC for mission-critical IoT requires that the time-to-transmit latency is within 500 μs while maintaining a block error rate no greater than 10^{-5}.

Several short blocklength code designs have been proposed in the literature. Important examples include the tail-biting (TB) convolutional codes decoded using the WAVA [12], extended BCH codes under ordered statistics decoding [13,14], non-binary low-density parity-check (LDPC) codes [15], non-binary turbo codes [16], and polar codes under CRC-aided successive-cancellation list decoding [17]. Recent advances also include the polarization adjusted convolutional codes by Arıkan [18]. As a comprehensive overview, Coşkun *et al.* [13] surveyed most of the contemporary short blocklength code designs in the recent decade. We refer the reader to [13] for additional information.

In [19], Yang *et al.* proposed the CRC-aided CCs as a powerful short blocklength code for binary-input (BI) additive white Gaussian noise (AWGN) channels. In [19], the convolutional encoder of interest has rate-$1/n$ and is either zero-terminated (ZT) or TB. In order to construct a good CRC-aided CC, Yang *et al.* selects a CC that maximizes its minimum distance and designs a *distance-spectrum optimal* (DSO) CRC generator polynomial for the given CC. The resulting concatenated code generated by the DSO CRC polynomial and the convolutional encoder is a good CRC-aided CC.

The nature of the concatenation naturally permits the use of the serial list Viterbi decod-

ing (SLVD), an efficient algorithm originally proposed by Seshadri and Sundberg [20]. Yang *et al.* showed that the expected list rank of SLVD of the CRC-aided CC is small at high SNR where the target error probability is low, thus achieving a low average decoding complexity at the operating point of interest. Yang *et al.* demonstrated that several concatenated codes generated by the DSO CRC polynomial and the TBCC, or in short, CRC-TBCCs, approach the RCU bound. In [21], Schiavone extended this line of work by looking at the parallel list Viterbi decoding with a bounded list size. In our precursor conference paper [3], this framework is extended to rate-$(n-1)/n$ CCs and the resulting concatenated code is able to approach the RCU bound with a low decoding complexity as well.

In this chapter, we present designs of good CRC-aided CCs for rate-$(n-1)/n$ CCs at short blocklengths for the BI-AWGN channel, where the CC is either ZT or TB. We consider systematic, rate-$(n-1)/n$ convolutional encoders. The resulting concatenated codes are respectively called a CRC-ZTCC and CRC-TBCC. We assume that SLVD has a sufficiently large list size such that no negative acknowledgement is produced. Thus, SLVD is an implementation of maximum-likelihood decoding. The frame error rate (FER) is in fact the undetected error probability. Simulations show that in the short-blocklength regime, our rate-$(n-1)/n$ CRC-TBCCs' performance is close to the RCU bound.

A work related to this line of research is that of Karimzadeh & Vu [1]. They considered designing the optimal CRC polynomial for multi-input CCs. In their framework, the information sequence is first divided into $(n-1)$ streams, one for each input rail, and they aim at designing optimal CRC polynomial for each rail. Unlike their architecture, in this chapter, the information sequence is first encoded with a single CRC polynomial and is then divided into $(n-1)$ streams.

For rate-$(n-1)/n$ CCs, SLVD on the primal trellis requires high decoding complexity due to the 2^{n-1} outgoing branches at each node. SLVD implementation becomes exponentially more complicated when there are more than two outgoing branches per state. In order to simplify SLVD implementation and reduce complexity, we utilize the *dual trellis* pioneered by Yamada *et al.* [6]. The dual trellis expands the length of the primal trellis by a factor of n, while reducing the number of outgoing branches at each node from 2^{n-1} to at most two.

We consider two architectures to enforce the TB condition for CRC-TBCCs. One approach uses a single trellis with all initial states possible. At low SNR, SLVD on the single trellis requires a large list size to identify the ML TB codeword, with a majority of trellis paths not satisfying the TB condition. Since only one trellis is constructed in the forward pass and all tracebacks are conducted on this single trellis, it's impractical to enforce the TB condition when finding a new path. Therefore, we propose a new multi-trellis approach, where multiple copies of the dual trellis are initialized. Each trellis corresponds to a unique starting and ending state pair, therefore it is guaranteed that all paths found will be TB paths. This approach trades off the decoding time complexity of potentially exploring a large list size against the upfront overhead and space complexity for creating and storing multiple trellises. It can provide a benefit over the single-trellis approach when noise level is high.

Introduced in [22], WAVA is a near-maximum likelihood decoding algorithm for TBCCs. To achieve the balance of decoding time and space complexities, we propose an approach that combines the wrap-around behavior of WAVA with SLVD for TBCCs. The decoding process is completed in two steps: the WAVA step with at most 2 trellis iterations, and the list decoding step with a sufficiently large list size such that there is no negative acknowledgment signals. Simulation results demonstrate that this decoding method reduces the average list size as compared with the single-trellis decoder without WAVA, but this reduction comes at a cost of degraded FER performance.

2.2.1 Contributions

As a primary contribution, this work extends our previous work on high-rate CRC-CC list decoding with the dual trellis [3] to further reduce the list size and decoding complexity. The original dual trellis approach, which uses the same tree-trellis algorithm to store the path metrics, suffers from a high average list rank at low SNRs. The novelty and contributions in this chapter are summarized as follows:

- *Complexity comparison of three list decoders for TBCCs.*
  - An ML multi-trellis decoder that includes only TB codewords in the list of poten-

tial codewords by maintaining a distinct trellis for each starting state. We refer to this decoding scheme as the multi-trellis approach.

  - An ML decoder that avoids the complexity of having 2^v distinct trellises by using a single trellis but faces a potentially significant increase in list size by including both TB and non-TB paths in the list of potential transmitted codewords.
  - A sub-optimal (non-ML) decoder that lowers the list size of the single-trellis approach by using the wrap-around Viterbi algorithm (WAVA) to set the initial state metrics of the traceback trellis.

- *DSO CRC design for high-rate ZTCCs and TBCCs.*

  - This chapter presents DSO CRC polynomial designs of various degrees for optimum rate-3/4 ZTCCs and TBCCs [2] in Table 2.1 and Table 2.2. Formulated on the DSO CRC design algorithm for rate-$1/n$ CCs [19], the design steps for high-rate include collecting the irreducible error events (IEEs), reconstructing all possible paths, and finally identifying the DSO CRC. These CRCs help reduce the gap between simulated SNR and the RCU bound.
  - This work compares the FER performance of a single DSO CRC and multiple shorter CRCs [1] for the same rate-$(n-1)/n$ ZTCCs, where the total CRC degrees of the two approaches are equal. The multi-CRC scheme assigns a different CRC for each of the $(n-1)$ input rails. Simulation results show that our framework can yield better FER performance than that of Karimzadeh & Vu.

2.2.2 My Work

My contributions to this chapter included the development of the software we used for running simulations, which was done in collaboration with Ava Asmani, and later Beryl Sui as well. I was also involved in the development of the multiple decoding algorithms we were experimenting with.

2.2.3 Organization

The remainder of this chapter is organized as follows. Sec. 2.3 reviews systematic encoding for $(n, n-1, v)$ convolutional codes, describes the dual trellis construction, and explains the tree-trellis algorithm for maintaining a list for trellis paths. Sec. 2.4 considers CRC-ZTCCs for rate-$(n-1)/n$ CCs. It addresses the zero-termination issue, presents DSO CRC design for high-rate ZTCCs, and shows CRC-ZTCC simulation results. Sec. 2.5 considers CRC-TBCCs for rate-$(n-1)/n$ CCs. It addresses how to find the tail-biting initial state over the dual trellis and describes DSO CRC design for TBCCs. Additionally, Sec. 2.5 introduces the multi-trellis decoder and WAVA decoder and analyzes the complexity and decoding performance of all three decoding schemes. Sec. 2.6 concludes the chapter.

2.3 Systematic Encoding and Dual Trellis

This section describes systematic encoding for $(n, n-1, v)$ convolutional codes and introduces the dual trellis proposed by Yamada *et al.* [6] for high-rate CCs generated with an $(n, n-1, v)$ convolutional encoder, where v represents the overall constraint length. This section also discusses the tree-trellis algorithm and its benefits.

2.3.1 Notation

Let K and N denote the information length and blocklength in bits. Let $R = K/N$ denote the rate of the CRC-aided CC. A degree-m CRC polynomial is of the form $p(x) = 1 + p_1 x + \cdots + p_{m-1} x^{m-1} + x^m$, where $p_i \in \{0, 1\}$, $i = 1, 2, \ldots, m-1$. For brevity, a CRC polynomial is represented in hexadecimal when its binary coefficients are written from the highest to lowest order. For instance, 0xD represents $x^3 + x^2 + 1$. The codewords are BPSK modulated. The SNR is defined as $\gamma_s \triangleq 10 \log_{10}(A^2)$ (dB), where A represents the BPSK amplitude and the noise is distributed as a standard normal.

2.3.2 Systematic Encoding

We briefly follow [2, Chapter 11] in describing a systematic $(n, n-1, v)$ convolutional encoder. A systematic $(n, n-1, v)$ convolutional encoder can be represented by its parity check matrix

$$H(D) = [h^{(n-1)}(D), h^{(n-2)}(D), \ldots, h^{(0)}(D)], \tag{2.1}$$

where each $h^{(i)}(D)$ is a polynomial of degree up to v in delay element D associated with the i-th code stream, i.e.,

$$h^{(i)}(D) = h_v^{(i)} D^v + h_{v-1}^{(i)} D^{v-1} + \cdots + h_0^{(i)}, \tag{2.2}$$

where $h_j^{(i)} \in \{0, 1\}$. For convenience, we represent each $h^{(i)}(D)$ in octal form. For instance, $H(D) = [D^3+D^2+D+1, D^3+D^2+1, D^3+D+1]$ can be concisely written as $H = (17, 15, 13)$. Let $\boldsymbol{h}^{(i)} \triangleq [h_v^{(i)}, h_{v-1}^{(i)}, \ldots, h_0^{(i)}]$, $i = 0, 1, \ldots, n-1$. The systematic encoding matrix $G(D)$ associated with $H(D)$ is given by

$$G(D) = \begin{bmatrix} \frac{h^{(1)}(D)}{h^{(0)}(D)} & 1 & 0 & \cdots & 0 \\ \frac{h^{(2)}(D)}{h^{(0)}(D)} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{h^{(n-1)}(D)}{h^{(0)}(D)} & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{2.3}$$

The first output bit is a coded bit and the remaining output bits are a direct copy of the corresponding input bits.

2.3.3 Dual Trellis

The primal trellis associated with a rate-$(n-1)/n$ ZTCC has 2^{n-1} outgoing branches per state. Performing SLVD over the primal trellis when $n > 2$ is highly complex. In [19], the low decoding complexity of SLVD for rate-$1/n$ convolutional codes relies on the fact that only 2 outgoing branches are associated with each state. In order to efficiently perform SLVD, we consider the dual trellis proposed by Yamada *et al.* [6].

We briefly explain the dual trellis construction for parity check matrix $H(D) = [h^{(n-1)}(D),$

$h^{(n-2)}(D), \ldots, h^{(0)}(D)]$. First, we define the maximum instant response order λ as

$$\lambda \triangleq \max\{j \in \{0, 1, \ldots, n-1\} : h_0^{(j)} = 1\}. \tag{2.4}$$

The state of the dual trellis is represented by the partial sums of $(v+1)$ adders in the observer canonical form of $H(D)$. At time index j, $j = 0, 1, \ldots, n-1$, the state is given by

$$\boldsymbol{s}^{(j)} = [s_v^{(j)}, s_{v-1}^{(j)}, \ldots, s_0^{(j)}]. \tag{2.5}$$

Next, we show how the state $\boldsymbol{s}^{(j)}$ evolves in terms of the output bits $\boldsymbol{y}_k = [y_k^{(0)}, y_k^{(1)}, \ldots, y_k^{(n-1)}]$, $k = 1, 2, \ldots, N/n$, so that a dual trellis can be established.

Dual trellis construction for $\boldsymbol{y}_k = [y_k^{(0)}, y_k^{(1)}, \ldots, y_k^{(n-1)}]$:

1) At time $j = 0$, $\boldsymbol{s}^{(0)} = [0, s_{v-1}^{(j)}, s_{v-2}^{(j)}, \ldots, s_0^{(j)}]$, where $s_i^{(0)} \in \{0, 1\}$. Namely, only 2^v states exist at $j = 0$.

2) At time j, $j < n-1$, draw branches from each state $\boldsymbol{s}^{(j)}$ to state $\boldsymbol{s}^{(j+1)}$ by

$$\boldsymbol{s}^{(j+1)} = \boldsymbol{s}^{(j)} + y_k^{(j)} \boldsymbol{h}^{(j)}, \quad y_k^{(j)} \in \{0, 1\}. \tag{2.6}$$

3) At time $j = n-1$, draw branches from each state $\boldsymbol{s}^{(n-1)}$ to state $\boldsymbol{s}^{(n)}$ by

$$\boldsymbol{s}^{(n)} = \left(\boldsymbol{s}^{(n-1)} + y_k^{(n-1)} \boldsymbol{h}^{(n-1)}\right)^r, \; y_k^{(n-1)} \in \{0, 1\}, \tag{2.7}$$

where $(a_v, a_{v-1}, \ldots, a_1, a_0)^r = (0, a_v, a_{v-1}, \ldots, a_1)$.

4) For time $j = \lambda$, draw a branch from each state $\boldsymbol{s}^{(\lambda)}$ according to (2.6) only for $y_k^{(\lambda)} = s_0^{(\lambda)}$.

After repeating the above construction for each $\boldsymbol{y}_k$, $k = 1, 2, \ldots, N/n$, we obtain the dual trellis associated with the $(n, n-1, v)$ convolutional code. Since the primal trellis is of length N/n, whereas the dual trellis is of length N, the dual trellis can be thought of as expanding the primal trellis length by a factor of n, while reducing the number of outgoing branches per state from 2^{n-1} to less than or equal to 2.

2.3.4 Tree-Trellis Algorithm

SLVD enumerates possible paths through the trellis, starting from the lowest weight path, stopping once the first path that satisfies both the CRC and the TB condition is reached. Thus recordings of previously investigated path metrics are required to find the next optimal path. Inserting or accessing an element in an unsorted list of path metrics can substantially increase the time complexity of an algorithm.

To efficiently perform SLVD on the dual trellis, we used the tree-trellis algorithm (TTA) proposed by Soong and Huang in [23]. The TTA maintains a sorted list of nodes which are indexed by path metric. These nodes either correspond to a previously unexplored ending state in the trellis, or to a previously explored path and a detour. This approach allows the efficient determination of the next path to be explored if the current one does not satisfy both the CRC and TB condition.

This algorithm maintains a sorted list, which can become expensive if not implemented with an efficient data structure. In [24], the authors used Red-Black tree [25] to maintain the sorted list of nodes. In this chapter, we use a Min Heap [26], which is easier to implement and has the same $\mathcal{O}(\log \ell)$ time complexity to maintain the properties of its structure. A Min Heap is a complete binary tree that often underpins practical priority queue implementations. It requires no space overhead over a standard array. It provides constant-time minimum element access and logarithmic-time deletion of the minimum element. Insertion of a new element is logarithmic-time in the worst case, but constant-time on average. Constant average insertion time helps control overall average time complexity because insertion is the dominant operation performed by the decoder. Every path searched requires only accessing and removing the minimum node, and all detours along that path are inserted into the heap as nodes.

Table 2.1: DSO CRC polynomials for rate-3/4 ZTCC at blocklength $N = 128$ generated by $H = (33, 25, 37, 31)$ with $v = 4$, by $H = (47, 73, 57, 75)$ With $v = 5$, and by $H = (107, 135, 133, 141)$ with $v = 6$. © 2022 IEEE

K	m	R	$v = 4$ CRC	$v = 5$ CRC	$v = 6$ CRC
87	3	0.680	0x9	0x9	0xB
86	4	0.672	0x1B	0x15	0x1D
85	5	0.664	0x25	0x25	0x25
84	6	0.656	0x4D	0x7B	0x6F
83	7	0.648	0xF3	0xED	0x97
82	8	0.641	0x1E9	0x1B7	0x1B5
81	9	0.633	0x31B	0x3F1	0x2F1
80	10	0.625	0x5C9	0x66F	0x59F
79	11	0.617	0xC2B	0xE8D	0xD2D

2.4 ZTCC with DSO CRC via Dual Trellis SLVD

This section considers CRC-ZTCCs for rate-$(n-1)/n$ CCs. Section 2.4.1 presents a zero termination method over the dual trellis. Section 2.4.2 describes our DSO CRC polynomial search procedure. Finally, Section 2.4.3 presents simulation results of the CRC-ZTCC compared with the RCU bound. As a case study, this chapter mainly focuses on the rate-3/4 systematic feedback convolutional codes in [2, Table 12.1(e)].

2.4.1 Zero Termination of Dual Trellis

For an $(n, n-1, v)$ CC, zero termination over the dual trellis requires at most $n\lceil v/(n-1)\rceil$ steps. In our implementation, a breadth-first search identifies the zero-termination input and output bit patterns that provide a trajectory from each possible state $\boldsymbol{s}$ to the zero state. The input and output bit patterns have lengths $(n-1)\lceil v/(n-1)\rceil$ and $n\lceil v/(n-1)\rceil$ respectively.

2.4.2 Design of DSO CRCs for High-Rate ZTCCs

In general, a DSO CRC polynomial provides the optimal distance spectrum which minimizes the union bound on the FER at a specified SNR [19]. In this chapter, we focus on the low FER regime. Thus, the DSO CRC polynomials identified in this chapter simply maximize the minimum distance of the concatenated code. Examples in [19] indicate that DSO CRC polynomials designed in this way can provide optimal or near-optimal performance for a wide range of SNRs.

The design procedure of the DSO CRC polynomial for high-rate ZTCCs essentially follows from the DSO CRC design algorithm for low target error probability in [19]. The first step is to collect the IEEs, which are ZT paths on the trellis that deviate from the zero state once and rejoin it once. IEEs with a very large output Hamming weight do not affect the choice of optimal CRCs. In order to reduce the runtime of the CRC optimization algorithm, IEEs with output Hamming weight larger than a threshold $\tilde{d}-1$ are not considered. Dynamic programming constructs all ZT paths of length equal to N/n and output weight less than $\tilde{d}$. Finally, we use the resulting set of ZT paths to identify the degree-m DSO CRC polynomial for the rate-$(n-1)/n$ CC.

Table 2.1 presents the DSO CRC polynomials for ZTCCs generated with $H = (33, 25, 37, 31)$, $H = (47, 73, 57, 75)$, and $H = (107, 135, 133, 141)$. The design assumes a fixed block-length $N = 128$ bits. Due to the overhead caused by the CRC bits and by zero termination, the rates of CRC-ZTCCs are less than 3/4. Specifically, for a given information length K, CRC degree m and an $(n, n-1, v)$ encoder, the blocklength N for a CRC-ZTCC is given by

$$N = \left(K + m + (n-1) \left\lceil \frac{v}{n-1} \right\rceil \right) \frac{n}{n-1}, \tag{2.8}$$

giving

$$R = \frac{K}{N} = \frac{n-1}{n} \frac{K}{K + m + (n-1)\lceil \frac{v}{n-1} \rceil}. \tag{2.9}$$

We see from (2.8) that the $(n, n-1, v)$ convolutional encoder can accept any CRC degree m as long as $K + m$ is divisible by $(n-1)$.

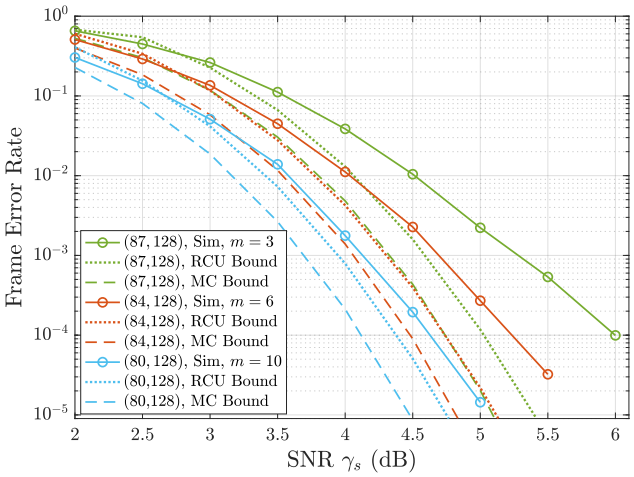

Figure 2.1: FER vs. SNR for various CRC-ZTCCs. The ZTCC is generated with the $(4, 3, 6)$ encoder $H = (107, 135, 133, 141)$. The DSO CRC polynomials of degrees $3, 6$, and 10 are 0xB, 0x6F, and 0x59F, respectively. Values in parenthesis denote information length K and blocklength N, respectively. © 2022 IEEE

2.4.3 Results and Comparison with RCU Bound

Fig. 2.1 shows the performance of CRC-ZTCCs with increasing CRC degrees $3, 6$ and 10 and a fixed blocklength $N = 128$ bits. We see that at the target FER of 10^{-4}, increasing the CRC degree reduces the gap to the RCU bound. With $m = 10$ and $v = 6$, the CRC-ZTCC approaches the RCU bound within 0.25 dB.

In [1], Karimzadeh *et al.* considered designing optimal CRC polynomials for each input rail of a multi-input CC. In their setup, an information sequence for an $(n, n-1, v)$ encoder needs to be split into $(n-1)$ subsequences before CRC encoding. In contrast, the entire information sequence in our framework is encoded with a single CRC polynomial. Then the resulting sequence is evenly divided into $(n-1)$ subsequences, one for each rail. To compare the performance between these two schemes, we design three degree-3 optimal CRC polynomials, one for each rail, for ZTCC with $H = (47, 73, 57, 75)$. The three CRC polynomials jointly maximize the minimum distance of the CRC-ZTCC. For the single-CRC design, we use the single degree-9 DSO CRC polynomial for the same encoder from Table 2.1. Both CRC-ZTCCs have an information length $K = 81$ and blocklength $N = 128$. Fig. 2.2

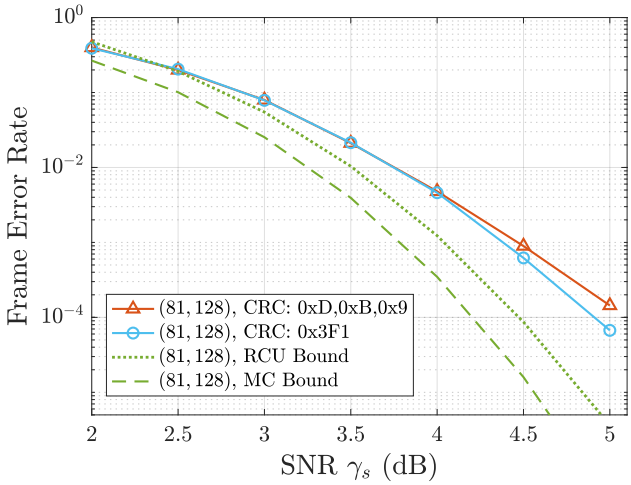

Figure 2.2: FER vs. SNR for $v = 5$ CRC-ZTCCs designed under Karimzadeh *et al.*'s scheme [1] and our scheme. Both CRC-ZTCCs have information length $K = 81$ and blocklength $N = 128$. © 2022 IEEE

shows the performance comparison between these two codes, showing that a single degree-9 DSO CRC polynomial outperforms three degree-3 DSO CRC polynomials, one for each rail. This suggests that a single DSO CRC polynomial may suffice to provide superior protection for each input rail. The decoding complexity is similar regardless of the CRC scheme.

2.5 TBCC with DSO CRC and Dual Trellis SLVD

The performance of CRC-aided list decoding of ZTCCs relative to the RCU bound is constrained by the termination bits appended to the end of the original message, which are required to bring the trellis back to the all-zero state. TBCCs avoid this overhead by replacing the zero termination condition with the TB condition, which states that the final state of the trellis is the same as the initial state of the trellis [27].

In this section, we apply SLVD to CRC-TBCCs over the dual trellis. Section 2.5.1 demonstrates designs of DSO CRCs for rate $(n-1)/n$ TB codes. In section 2.5.2, we will discuss how to determine the initial state for the TBCC to ensure that the TB condition is met on a single dual trellis. Detailed discussion about the construction of the multi-trellis

Table 2.2: DSO CRC polynomials for rate-3/4 TBCC at blocklength $N = 128$ generated by $H = (33, 25, 37, 31)$ with $v = 4$, by $H = (47, 73, 57, 75)$ with $v = 5$, and by $H = (107, 135, 133, 141)$ with $v = 6$. © 2022 IEEE

K	m	R	$v = 4$ CRC	$v = 5$ CRC	$v = 6$ CRC
93	3	0.727	0x9	0x9	0xB
92	4	0.719	0x1B	0x1D	0x17
91	5	0.711	0x25	0x3B	0x33
90	6	0.703	0x7D	0x4F	0x41
89	7	0.695	0xF9	0xD1	0xBD
88	8	0.688	0x1CF	0x173	0x111
87	9	0.680	0x38F	0x3BF	0x333
86	10	0.672	0x73F	0x697	0x723

decoder and WAVA decoder and their benefits are presented in sections 2.5.3 and 2.5.4, respectively. Finally, the decoding complexity and performance are analyzed in sections 2.5.5 and 2.5.6.

2.5.1 Design of DSO CRCs for High-Rate TBCCs

The design of DSO CRCs for high-rate TBCCs follows the two-phase design algorithm shown in [19]. This algorithm is briefly explained below.

Consider a TB trellis $T = (V, E, \mathcal{A})$ of length N, where $\mathcal{A}$ denotes the set of output alphabet, V denotes the set of states, and E denotes the set of edges described in an ordered triple (s, a, s') with $s, s' \in V$ and $a \in \mathcal{A}$ [28]. Assume $|V| = 2^v$ and let $V_0 = \{0, 1, \ldots, 2^v - 1\}$. Define the set of IEEs at state $\sigma \in V$ as

$$\mathrm{IEE}(\sigma) \triangleq \bigcup_{l=1,2,\ldots,N} \overline{\mathrm{IEE}}(\sigma, l), \tag{2.10}$$

where

$$\overline{\mathrm{IEE}}(\sigma, l) \triangleq \{(\boldsymbol{s}, \boldsymbol{a}) \in V_0^{l+1} \times \mathcal{A}^l : s_0 = s_l = \sigma; \\ \forall j, 0 < j < l,\ s_j \notin \{0, 1, \ldots, \sigma\}\}. \tag{2.11}$$

The IEEs at state σ can be thought of as "building blocks" for an arbitrarily long TB path that starts and ends at the same state σ.

The first phase is called the collection phase, during which the algorithm collects $\mathrm{IEE}(\sigma)$ with output Hamming weight less than the threshold $\tilde{d}$ over a sufficiently long TB trellis. The second phase is called the search phase, during which the algorithm first reconstructs all TB paths of length N/n and output weight less than $\tilde{d}$ via concatenation of the IEEs and circular shifting of the resulting path. Then, using these TB paths, the algorithm searches for the degree-m DSO CRC polynomial by maximizing the minimum distance of the undetected TB path.

Table 2.2 presents the DSO CRC polynomials for TBCCs generated with $H = (33, 25, 37, 31)$, $H = (47, 73, 57, 75)$, and $H = (107, 135, 133, 141)$. The design assumes a fixed blocklength $N = 128$. TB encoding avoids the rate loss caused by the overhead of the zero termination. Specifically, for a given information length K, CRC degree m and an $(n, n-1, v)$ encoder, the blocklength N for a CRC-TBCC is given by

$$N = (K + m)\frac{n}{n-1}, \tag{2.12}$$

giving

$$R = \frac{K}{N} = \frac{n-1}{n}\frac{K}{K+m}. \tag{2.13}$$

2.5.2 Single Trellis List Decoding for CRC-TBCC

There are two primary differences in the development and analysis of list decoding between ZTCCs, as described in Section 2.4, and TBCCs. One difference is that since the ZT condition is replaced with the TB condition, the encoder must determine the initial trellis state so that the TB condition is satisfied. The other difference is that SLVD on the dual trellis must be adapted to handle the TB condition.

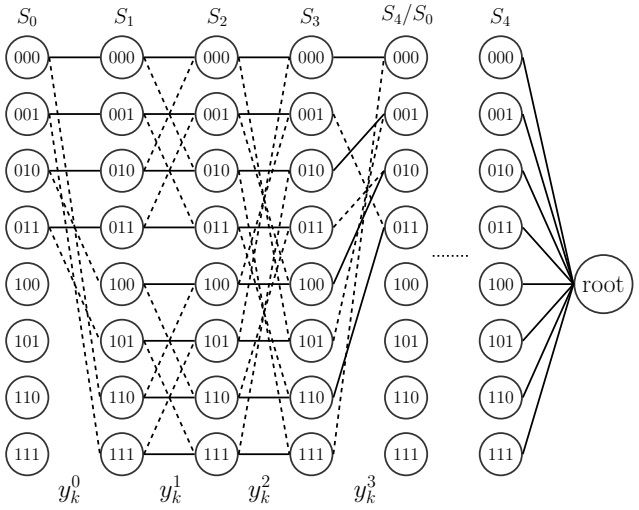

Figure 2.3: Dual trellis diagram for rate-3/4 TBCC with a root node at the end for encoder $H = (2, 5, 7, 6)$ with $v = 2$. Solid lines represent 0 paths and dashed lines represent 1 paths.

To satisfy the TB condition, encoding is attempted from every initial state to identify the initial state that satisfies the TB condition. This is required because our recursive encoder cannot simply achieve the TB condition by setting the initial encoder memory to be the final bits of the information sequence.

To adapt SLVD on the dual trellis to handle the TB condition, we propose an efficient way to keep track of the path metrics and find the next path with minimum metric through an additional root node as shown in Fig. 2.3. The root node connects to all terminating states after forward traversing the dual trellis. The Hamming distance of the branch metric for the branch connecting any state to this root node is zero. This additional root node allows the trellis to end in a single state, so that the basic SLVD approach for a ZTCC may be applied. During SLVD, if the current path does not pass either the CRC or TB check, the minimum value among all remaining path metrics will be selected as the next path to check.

2.5.3 Multi-Trellis List Decoding for CRC-TBCC

Decoding on a single dual trellis (single-trellis approach) leads to complexity issues, since a large sorted list is maintained to keep track of all possible traceback paths – but not all of them will satisfy the TB condition. Thus the decoder will go through a number of paths that pass neither the TB check nor CRC, resulting in high expected list ranks at low SNRs. To decode more efficiently, we propose a multi-trellis approach in this section.

As its name indicated, the multi-trellis approach requires construction of 2^v dual trellises, since only half of the states in a dual trellis are valid starting and ending states. As shown in Fig. 2.4, each multi-trellis follows the same structure as a dual trellis, but with only one starting and ending state to enforce the TB condition. For example, for the encoder $H = (2, 5, 7, 6)$ with $v = 2$, there are $2^2 = 4$ multi-trellises for this encoder. The final root node is also removed.

An obvious advantage of using the multi-trellis approach is that all the paths found will be tail-biting. This greatly reduces the expected list size compared to the single-trellis approach at low SNRs. However, at high SNRs, the multi-trellis approach has a substantially higher decoding complexity due to the additional upfront cost of constructing the dual trellises. In this case, the extra resources taken to initialize the multi-trellis approach brings down the overall decoder efficiency.

2.5.4 List Decoding with WAVA

As the constraint length of a TBCC increases, the number of states grows exponentially. The multi-trellis approach becomes impractical due to both time and memory for constructing the trellises. Thus, we adopt a non-ML decoder with WAVA to improve the decoding efficiency while maintaining a reasonable decoding complexity. Proposed in [22], the WAVA decodes TB trellises iteratively and is able to effectively reduce the expected list rank. An initial examination of the list size distributions for the three decoding schemes at SNR = 2 dB is presented in Fig. 2.5 on a $v = 4$ TBCC. While the multi-trellis approach outperforms the other approaches with an extremely small list size at all times, the WAVA approach still has

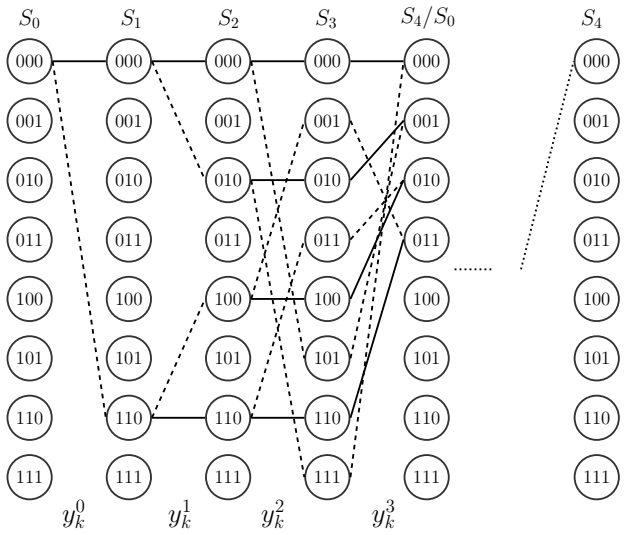

Figure 2.4: Multi-trellis construction for rate-3/4 TBCC dual trellis with encoder $H = (2, 5, 7, 6)$ with $v = 2$. The pictured trellis has starting and ending state of 0. Three similar trellises are also constructed for this encoder with starting and ending states of 1-3.

a substantially larger probability of a small list size compared to the single-trellis. Different distributions lead to different $\mathsf{E}[I]$ and $\mathsf{E}[L]$ values when evaluating the decoding complexity.

The non-ML decoder with WAVA proceeds in two steps. In the first step, the algorithm initializes each state of a single dual trellis with all zero metrics. It then performs two iterations of add-compare-select (ACS) along the trellis; each time the end of the trellis is encountered, the initial states of the trellis are initialized to the cumulative metrics in the final states. At the end of the first iteration, if the optimal path satisfies TB and CRC conditions, the algorithm outputs this path and stops decoding. In the second step, SLVD runs on the ending metrics of the second trellis iteration. This decoding algorithm improves the reliability of the final decision for the optimal traceback path and decreases the expected list rank while keeping the complexity low. However, this algorithm is non-optimal, thus it has a slightly worse decoding performance than the other two ML approaches.

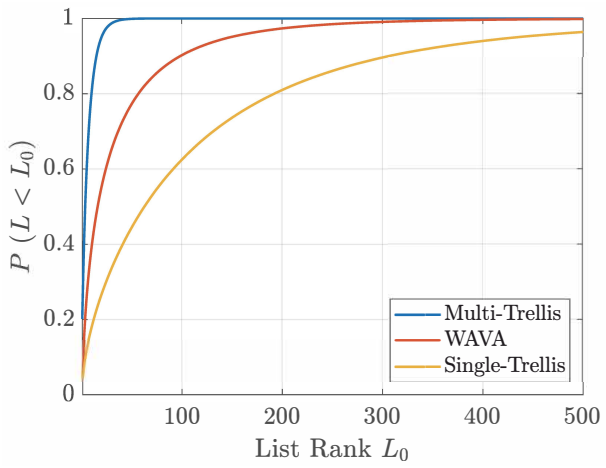

Figure 2.5: Cumulative distribution function (CDF) of list ranks for the single-trellis, multi-trellis, and WAVA decoding approaches for the $(33, 25, 37)$ TBCC with blocklength of 128 at SNR = 2 dB. © 2022 IEEE

2.5.5 Complexity Analysis

In [19], the authors provided the complexity expression for SLVD of CRC-ZTCCs and CRC-TBCCs, where the convolutional encoder is of rate $1/n$. Observe that the dual trellis has no more than 2 outgoing branches per state, similar to the trellis of a rate-$1/n$ CC. Thus, we directly apply their complexity expression to SLVD over the dual trellis.

As noted in [19], the overall average complexity of SLVD can be decomposed into three components:

$$C_{\text{SLVD}} = C_{\text{SSV}} + C_{\text{trace}} + C_{\text{list}}, \tag{2.14}$$

where C_{SSV} denotes the complexity of a standard soft Viterbi (SSV), C_{trace} denotes the complexity of the *additional* traceback operations required by SLVD, and C_{list} denotes the average complexity of inserting new elements to maintain an ordered list of path metric differences.

C_{SSV} is the complexity of ACS operations and the initial traceback operation. For CRC-

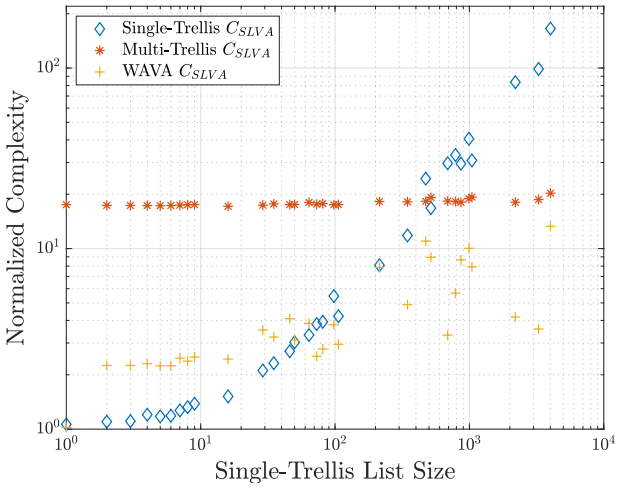

Figure 2.6: The overall complexity comparison of the single-trellis, multi-trellis, and WAVA decoders for the TBCC generated with the $(4, 3, 4)$ encoder $H = (33, 25, 37)$, with blocklength of 128. The CRC polynomial of degree 3 is 0x9. All complexity values are normalized with respect to the single-trellis C_{SSV} at different list sizes. © 2022 IEEE

ZTCCs,

$$C_{\text{SSV}} = (2^{v+1} - 2) + 1.5(2^{v+1} - 2) + 1.5(K + m - v)2^{v+1} + c_1[2(K + m + v) + 1.5(K + m)]. \tag{2.15}$$

For CRC-TBCCs decoded using the single-trellis, this quantity is given by

$$C_{\text{SSV}} = 1.5(K + m)2^{v+1} + 2^v + 3.5c_1(K + m). \tag{2.16}$$

For CRC-TBCCs with the multi-trellis approach,

$$C_{\text{SSV}} = 2^v[1.5(K + m)2^{v+1}] + 3.5c_1(K + m). \tag{2.17}$$

The second component C_{trace} for CRC-ZTCC is given by

$$C_{\text{trace}} = c_1(\mathsf{E}[L] - 1)[2(K + m + \nu) + 1.5(K + m)]. \tag{2.18}$$

For CRC-TBCCs, C_{trace} is given by

$$C_{\text{trace}} = 3.5c_1(\mathsf{E}[L] - 1)(K + m), \tag{2.19}$$

for both single-trellis and multi-trellis approaches.

The third component, which is identical for ZT and TB, is

$$C_{\text{list}} = c_2 \mathsf{E}[I] \log(\mathsf{E}[I]), \tag{2.20}$$

where $\mathsf{E}[I]$ is the expected number of insertions to maintain the sorted list of path metric differences. For CRC-ZTCCs,

$$\mathsf{E}[I] \leq (K+m)\mathsf{E}[L], \tag{2.21}$$

and for CRC-TBCCs with either single-trellis or multi-trellis approach,

$$\mathsf{E}[I] \leq (K+m)\mathsf{E}[L] + 2^v - 1. \tag{2.22}$$

In the above expressions, c_1 and c_2 are two computer-specific constants that characterize implementation-specific differences in the implemented complexity of traceback and list insertion (respectively) as compared to the ACS operations of Viterbi decoding. In this chapter, we assume that $c_1 = c_2 = 1$ and use (2.21) and (2.22) to estimate $\mathsf{E}[I]$ for CRC-ZTCCs and CRC-TBCCs, respectively.

Note that $\mathsf{E}[I]$ and $\mathsf{E}[L]$ values vary depending on whether the single-trellis or multi-trellis approach is used. Using the multi-trellis approach significantly reduces C_{trace} and C_{list} because only TB paths are included. On the other hand, as seen from (2.16) and (2.17), the multi-trellis approach amplifies the first component C_{SSV} by nearly 2^v. The overall tradeoff is depicted in Fig. 2.6, which shows the complexity comparison of the single-trellis and multi-trellis approach on a $v = 4$ TBCC with a $m = 3$ CRC. Various random codewords with blocklength $N = 128$ are generated and their single-trellis list sizes are measured by passing through a single-trellis SLVD with TB checks and CRCs. The runtime of each complexity component is normalized with respect to the value of single-trellis C_{SSV}. When the single-trellis list size is 1, the overall runtime of the multi-trellis approach is over 10 times of that of the single-trellis approach, due to the overhead complexity of constructing multiple trellises. At low noise level, the list size of a single trellis is 1 for the most time, resulting in a substantially lower runtime compared to that of a multi-trellis. As SNR decreases, the

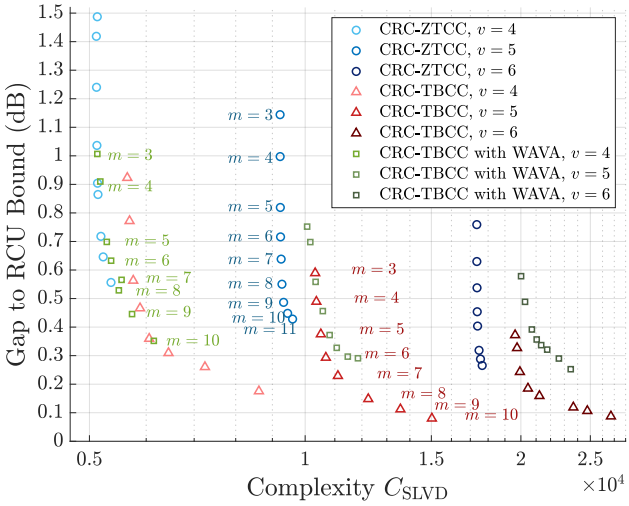

Figure 2.7: The SNR gap to the RCU bound vs. the average complexity of SLVD of CRC-ZTCC codes in Table I and CRC-TBCC codes in Table II for target FER of 10^{-4}. For CRC-TBCCs, results for both single-trellis decoding and WAVA decoding are demonstrated. Each color represents a specific CRC-aided CC shown in the tables. Markers from top to bottom with the same color correspond to DSO CRC polynomials with $m = 3, \ldots, 10$ for TBCCs, and $m = 3, \ldots, 11$ for ZTCCs. © 2022 IEEE

list size of the single-trellis decoder grows larger, leading to an exponential growth in the complexity terms C_{trace} and C_{list}. Although the multi-trellis C_{list} term also has an increasing trend, that value is small compared to the term C_{trace}. Because the construction of the trellis structure mainly contributes to the complexity of multi-trellis, at different SNR levels the complexity maintains the same magnitude. At a list size of around 5×10^2, the overall runtime C_{SLVD} of both approaches become the same. This indicates that at high SNR level, single-trellis is the optimal approach as the list size is low. But in cases where the noise level is high, the multi-trellis approach is guaranteed to satisfy the TB condition, thus it provides a reduced runtime regardless of its complex construction.

Upon applying WAVA, the overall average complexity for CRC-TBCC is incremented by ACS operations during the additional forward pass, if needed. Let the probability that the optimal path of the initial traceback does not satisfy either TB or CRC condition be P_{WAVA}. The list rank of the decoder is 1 with a probability of $1 - P_{\text{WAVA}}$. Thus we have the updated

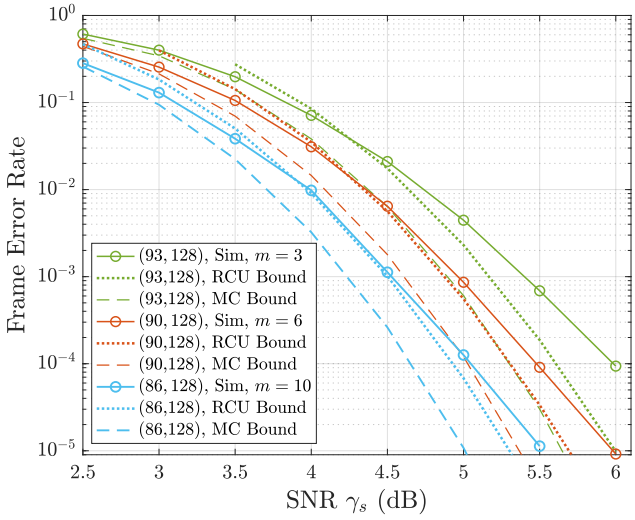

Figure 2.8: FER vs. SNR for various CRC-TBCCs. The TBCC is generated with the $(4, 3, 6)$ encoder $H = (107, 135, 133, 141)$. The DSO CRC polynomials of degrees $3, 6$, and 10 are 0xB, 0x41, and 0x723, respectively. Values in parenthesis denote information length K and blocklength N, respectively. © 2022 IEEE

complexity:

$$C_{\text{SLVD}} = C_{\text{SSV}} + P_{\text{WAVA}} \times (C_{\text{WAVA}} + C_{\text{trace}} + C_{\text{list}}), \tag{2.23}$$

where

$$C_{\text{WAVA}} = 1.5(K + m)2^{v+1} + 2^{v}. \tag{2.24}$$

The yellow data points in Fig. 2.6 represent the overall complexity of the WAVA decoder normalized with respect to the single-trellis C_{SSV} at various list sizes. The complexity for initializing the WAVA decoder is about 2 times of that for the single-trellis, giving it a disadvantage at low noise levels. However, at a list size of exactly 1, due to the traceback after the first iteration, the WAVA decoder matches the complexity of the single-trellis decoder. At a list size of around 5×10^1, the overall complexity of the WAVA decoder reaches the same level as the single-trellis decoder. In addition, the WAVA decoder operates at a complexity lower than the multi-trellis decoder.

2.5.6 Results, Analysis, and Expected List Rank of SLVD

Fig. 2.7 shows the trade-off between the SNR gap to the RCU bound and the average decoding complexity at the target FER 10^{-4} for CRC-ZTCCs designed in Table 2.1 and CRC-TBCCs designed in Table 2.2. The average decoding complexity of SLVD is evaluated according to the expressions in Sec. 2.5.5. We see that for a fixed v (ZT or TB), increasing the CRC degree m significantly reduces the gap to the RCU bound, at the cost of a small increase in complexity. The minimum gap of 0.08 dB is achieved by the CRC-TBCC with $v = 5$ and $m = 10$. However, for the same CRC degree m, increasing the overall constraint length v dramatically increases the complexity, while achieving a minimal reduction in the SNR gap to the RCU bound. For CRC-TBCCs, the WAVA decoder has a slightly larger gap to RCU bound than the single-trellis decoder due to the extra ACS operations during the first traceback. However, the complexity of the WAVA decoder is smaller than that of the single-trellis decoder, and the difference increases as the CRC degree increases. For all constrain lengths v, the WAVA decoder at $m = 10$ has a similar complexity as the single-trellis decoder at $m = 7$.

Additionally, the performance of CRC-ZTCC can be improved drastically by applying CRCs of higher degrees. Fig. 2.7 demonstrates that for all three cases of constraint lengths v, one additional bit in the CRC benefits the decoding performance by moving closer to the RCU gap with a minimal cost in complexity. Thus at a gap to RCU bound for around 0.4 dB, CRC-ZTCC outperforms CRC-TBCC by a substantially reduced decoding complexity.

Fig. 2.8 shows the FER vs. SNR for three CRC-TBCCs at blocklength $N = 128$. At the target FER of 10^{-4}, the SNR gap to the RCU bound is reduced to 0.1 dB for the CRC-TBCC with $m = 10$ and $v = 6$.

2.6 Conclusion

This chapter shows that high-rate CRC-aided CCs are able to approach the RCU bound for the BI-AWGN channel. The best CRC-TBCCs with the single-trellis ML decoder approach

the RCU bound within 0.1 dB for a target FER of 10^{-4} at a blocklength of $N = 128$ bits.

This chapter considers three decoding algorithms for rate-$(n-1)/n$ CCs concatenated with DSO CRC polynomials: a multi-trellis approach, a single-trellis approach, and a modified single trellis approach with pre-processing using the Wrap Around Viterbi Algorithm (WAVA). All three algorithms use the dual trellis to reduce complexity. The multi-trellis approach achieves the smallest expected list rank, but it suffers from a significantly larger overall complexity than the single-trellis approach. For the single trellis approach, we considered both an ML decoder and a non-ML decoder that uses WAVA pre-processing. WAVA pre-processing achieves a significantly smaller expected list size than the ML single-trellis decoder, but the non-ML decoder achieves a slightly worse FER performance. In addition, adding one bit to the CRC can improve the FER far more than adding an additional memory element to the CC does for all three approaches. In the future, it will be interesting to investigate and compare the performance and complexity of a parallel list Viterbi decoder (PLVD) on high-rate CC-CRC codes.

CHAPTER 3

ELF Codes: Concatenated Codes with an Expurgating Linear Function as the Outer Code

3.1 Abstract

An expurgating linear function (ELF) is an outer code that disallows low-weight codewords of the inner code. ELFs can be designed either to maximize the minimum distance or to minimize the codeword error rate (CER) of the expurgated code. A list-decoding sieve can efficiently identify ELFs that maximize the minimum distance of the expurgated code. For convolutional inner codes, this paper provides analytical distance spectrum union (DSU) bounds on the CER of the concatenated code.

For short codeword lengths, ELFs transform a good inner code into a great concatenated code. For a constant message size of $K = 64$ bits or constant codeword blocklength of $N = 152$ bits, an ELF can reduce the gap at CER 10^{-6} between the DSU and the random-coding union (RCU) bounds from over 1 dB for the inner code alone to 0.23 dB for the concatenated code. The DSU bounds can also characterize puncturing that mitigates the rate overhead of the ELF while maintaining the DSU-to-RCU gap. List Viterbi decoding guided by the ELF achieves maximum likelihood (ML) decoding of the concatenated code with a sufficiently large list size. The rate-$K/(K+m)$ ELF outer code reduces rate and list decoding increases decoder complexity. As SNR increases, the average list size converges to 1 and average complexity is similar to Viterbi decoding on the trellis of the inner code. For rare large-magnitude noise events, which occur less often than the FER of the inner code, a deep search in the list finds the ML codeword.

3.2 Introduction

Expurgating a code decreases the number of message symbols without changing the length [29]. Expurgation strengthens a code by removing weaker codewords at the expense of a slight reduction in rate. For example, in his proof of channel capacity, Gallager [30] removes the half of the randomly selected codewords for which error probability is largest to bound maximum rather than average probability of error. For a linear code, where the minimum distance $d_{\min}$ is the minimum weight of a nonzero codeword [29], expurgation that removes the lowest-weight codewords will increase $d_{\min}$ and thus reduce the probability of a codeword error.

Practical expurgation requires a function, i.e. an outer code, that selects which codewords to remove. This paper develops the paradigm of using an expurgating linear function (ELF) to remove the lowest weight codewords of a linear code or, more generally, to remove codewords so as to minimize a union bound on codeword error rate (CER). The improved performance comes at the expense of a reduction in rate by the rate-$K/(K+m)$ ELF outer code, but well-designed ELFs move CER performance closer to an appropriately rate-adjusted random coding union (RCU) bound [10, 31].

Lou *et al.* in [32] significantly improved error detection performance over traditional cyclic redundancy codes (CRCs) used with zero-terminated convolutional codes by specifically designing CRCs that remove low-weight codewords and thus reduce the undetected error rate. Subsequent papers [3, 19, 33–42] used this approach to improve the minimum distance or CER performance of the overall concatenated code, which is decoded using list decoding. These subsequent papers characterized the new designs as CRCs, but we call them ELFs since they are not constrained to be a cyclic code of any particular length and their primary function is expurgation rather than error detection.

For a convolutional inner code with an ELF as the outer code, ML decoding is achieved by serial or parallel list Viterbi decoding with a sufficiently large list size [20]. As observed in [19], as SNR increases, the expected list size converges to one. The average list size is often small at the desired CER.

BCH codes [34] and legacy codes that include CRCs can be reconsidered as ELF codes and decoded using list decoding to decrease CER. Schiavone *et al.* [43] provide a compelling example. The ELF paradigm applies to any inner code. Building on [44], ELFs designed to maximize the minimum distance of the expurgated polar code are described in [37, 45]. An ELF for a trellis code appears in [42]. This paper focuses on ELFs concatenated with tail-biting convolutional codes (TBCCs) [27] as an example.

3.2.1 Contributions

This paper presents generating function techniques for distance spectrum union (DSU) upper bounds on the CER of TBCCs concatenated with ELFs under maximum-likelihood (ML) decoding. These bounds are extended to include puncturing for rate compatibility. Such bounds can characterize the CER of every ELF for a given redundancy m and can be used to select the ELF (and the puncturing) that minimizes CER. Alternatively, this paper provides a sieve approach that uses list decoding from the zero-message codeword to identify the ELF that maximizes the minimum distance of the expurgated code. The DSU bounds show that an ELF can improve the gap between the DSU and RCU bounds from over 1 dB for the inner code alone to 0.23 dB for the concatenated code.

3.2.2 My Work

My contributions to this chapter included work on the list decoding sieve approach, and the related work on expurgated distance spectra.

3.2.3 Organization

Sec. 3.3 presents (DSU) upper bounds on the CER of convolutional inner codes concatenated with ELF outer codes with or without puncturing. Sec. 3.4 presents a sieve approach to identify the ELF that maximizes the minimum distance of the expurgated code. As an example, the best ELFs for expurgating a (152, 76) block code created from a $\nu = 8$ tail-biting convolutional code are identified for ELF redundancies of $0 \leq m \leq 12$. Sec. 3.5 uses

the tools of Sec. 3.3 and Sec. 3.4 to design an example punctured concatenation of an ELF and a tail-biting convolutional code. Sec. 3.6 concludes the paper.

3.3 Distance Spectrum Union Bounds

Generating functions provide upper bounds on the bit error rate [46] and the CER [32, 39] of convolutional codes. This section reviews the generating function approach to computing distance spectrum union (DSU) bounds on the CER of block codes constructed using convolutional codes and describes how such bounds may be applied to zero-terminated and tail-biting convolutional codes with ELFs and with puncturing.

3.3.1 DSU Bounds for Zero Termination and Tail Biting

Consider an (N, K) binary block code transmitted over the binary input AWGN channel where $+\sqrt{E_s}$ is transmitted for binary 0 and $-\sqrt{E_s}$ is transmitted for binary 1. Let A_w be the number of codewords with Hamming weight w. A DSU bound on codeword error rate P_{cw} is shown below:

$$P_{cw} \leq \sum_{w=d_{\min}}^{N} A_w Q\left(\sqrt{\frac{wE_s}{\sigma^2}}\right). \tag{3.1}$$

Define the weight enumerator polynomial as

$$A(w) = \sum_{w=d_{\min}}^{N} A_w W^w. \tag{3.2}$$

Using the bound $Q(\sqrt{x+y}) \leq Q(\sqrt{x}e^{-y/2})$ [46], a slightly looser bound than (3.1) may be computed from the weight enumerator polynomial as follows:

$$P_{cw} \leq Q\left(\sqrt{\frac{d_{\min}E_s}{\sigma^2}}\right) e^{\frac{d_{\min}E_s}{2\sigma^2}} A\left(e^{-\frac{E_s}{2\sigma^2}}\right). \tag{3.3}$$

A transition matrix $T(W)$ facilitates computation of $A(W)$ for either a zero-terminated or tail-biting convolutional code. For the example of a rate $1/n$ convolutional code with ν memory elements, $T(W)$ is an $s \times s$ matrix, where $s = 2^{\nu}$.

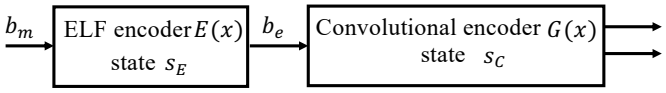

Figure 3.1: Convolutional encoder $G(x)$ with an ELF $E(x)$ as an outer code. © 2023 IEEE

The entry of $T(W)$ at row j and column i represents the transition from state i to state j in the convolutional encoder. If there is no transition from state i to state j, then the entry is zero. Otherwise, the entry is $W^{w_{i,j}}$ where $w_{i,j}$ is the Hamming weight of the n-bit symbol transmitted for that state transition.

Define the i^{th} basic row vector e_i as a length-s vector of all-zeros except a one in position i. We will index the entries in both $T(W)$ and e_i from zero to $s-1$. For example, with $\nu = 2$ and thus $s = 4$, $e_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}$.

For an $(N = n(K+\nu), K)$ block code implemented by sending K information bits through a ν-state, rate-$1/n$, zero-terminated convolutional code,

$$A(W) = e_0 T(W)^{K+\nu} e_0^T - 1. \tag{3.4}$$

Similarly, for an $(N = nK, K)$ block code implemented by sending K information bits through a 2^ν-state tail-biting convolutional code,

$$A(W) = \sum_{i=0}^{2^\nu - 1} e_i T(W)^K e_i^T - 1. \tag{3.5}$$

3.3.2 DSU Bound for a Convolutional Code with an ELF

A degree-m ELF $E(x)$ can be concatenated with a zero-terminated rate-$1/n$ convolutional code with encoder polynomial matrix $G(x)$. The m ELF redundancy bits reduce the rate from $\frac{K}{(K+\nu)n}$ to $\frac{K}{(K+\nu+m)n}$. Eq. (3.4) can be applied to the $T(W)$ with $s = 2^{(\nu+m)}$ for the zero-terminated convolutional code $E(x)G(x)$ so that

$$A(W) = e_0 T(W)^{K+\nu+m} e_0^T - 1. \tag{3.6}$$

For the case of an ELF $E(x)$ concatenated with a rate-$1/n$ tail-biting convolutional code, the rate reduction is from $\frac{K}{Kn}$ to $\frac{K}{(K+m)n}$. To derive the DSU bound, separately consider the

state s_E of the ELF encoder in Fig. 3.1 and the state s_C of the tail-biting convolutional encoder in Fig. 3.1. Note that

$$0 \leq s_E \leq 2^m - 1 \tag{3.7}$$

$$0 \leq s_C \leq 2^\nu - 1 \tag{3.8}$$

Define the overall encoder state of the concatenated code as $s = s_C + 2^\nu \times s_E$. To compute an entry in $T(W)$, the following computations are performed (referencing Fig. 3.1):
1) The message input bit b_m is processed by the ELF encoder with the origin ELF state $s_E^{(o)}$ to compute the destination ELF state $s_E^{(d)}$ and ELF output bit b_e. 2) The ELF output bit b_e is the input to the convolutional encoder which updates its origin state $s_C^{(o)}$ to the destination state $s_C^{(d)}$ and produces an n-bit output with Hamming weight $w_{i,j}$. 3) The weight term $W^{w_{i,j}}$ is written to $T(W)$ at row j and column i where $j = s_C^{(d)} + 2^\nu \times s_E^{(d)}$ and $i = s_C^{(o)} + 2^\nu \times s_E^{(o)}$. Using this $T(W)$, the weight enumerator polynomial for an $(N = 2 \times (K + m), K)$ block code implemented by sending K information bits through an outer ELF code with 2^m states and encoding the ELF output with a 2^ν-state tail-biting convolutional code is

$$A(W) = \sum_{i=0}^{2^\nu - 1} e_i T(W)^{K+m} e_i^T - 1. \tag{3.9}$$

In (3.9), $T(W)$ is an $2^{m+\nu} \times 2^{m+\nu}$ matrix, but the only paths that contribute to $A(W)$ are the 2^ν paths that start and end at the same value of s_C and that start and end with $s_E = 0$.

Fig. 3.2 shows the DSU bounds computed according to (3.9) and (3.3) for all of the 64 possible (142, 64) codes resulting from an $m = 7$ ELF concatenated with the $\nu = 8$ tail-biting convolutional code (561,753), where 561 is octal for $1 + x^4 + x^5 + x^6 + x^8$. The RCU bound for (142,64) codes is shown for comparison. For reference, the DSU bound for the (128,64) code that results from using the tail-biting convolutional code without an ELF and its corresponding RCU bound are also shown. The CER curve from simulating list decoding of the tail-biting code used with the best ELF 0xFF shows that this DSU bound is tight for CER $\leq 10^{-6}$.

Fig. 3.2 shows how the DSU bound on CER varies with the choice of ELF. The best ELF is 0xFF whose DSU bound is 0.35 dB from the (142,64) RCU bound at CER=10^{-6}.

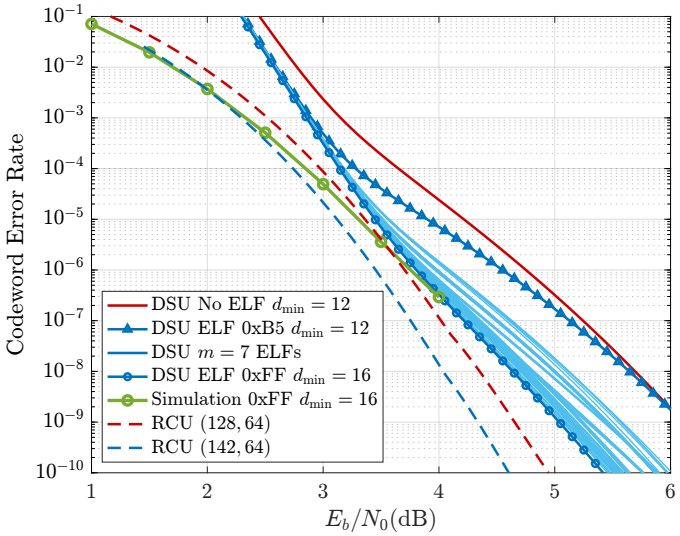

Figure 3.2: DSU bounds for the $\nu = 8$ tail-biting convolutional code with $K = 64$ message bits with no ELF (red) and with each possible $m = 7$ ELF (blue). Also shown is a simulation of list Viterbi decoding of the best ELF 0xFF (green) and, for reference, the (142,64) and (128,64) RCU bounds (dashed). © 2023 IEEE

The worst ELF is 0xB5 which is 1.10 dB from the RCU bound at CER=10^{-6}. The original (128,64) TBCC is 1.05 dB from the (128,64) RCU bound. Thus, in this case a well-designed 7-bit ELF reduced the gap from the RCU bound by 0.65 dB while the poorest choice had a slightly larger gap.

Fig 3.3 shows average list size as a function of E_b/N_0 for the list decoding simulation shown in Fig. 3.2. The maximum list size was set at 2^{20}, but the average list size is 1.26 at E_b/N_0=3.7 dB, where the CER is 1.1×10^{-6}. Thus the average complexity is similar to regular Viterbi decoding on a 256-state trellis but the gap to the RCU bound is only 0.35 dB.

3.3.3 DSU Bound for Punctured Convolutional Code with ELF

This section extends the DSU bounding technique to handle puncturing. The techniques below can be applied to any puncturing scheme. For simplicity of exposition, our description only considers puncturing at most one of the n bits in each convolutional encoder symbol.

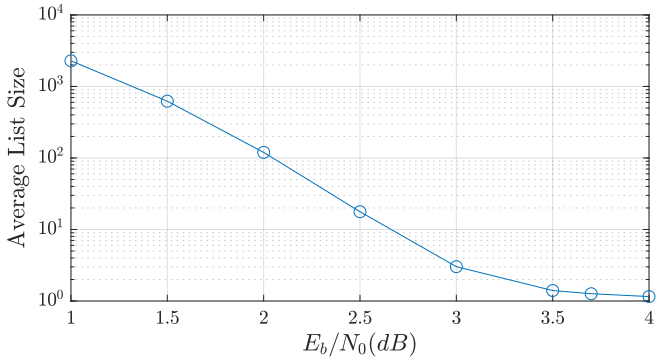

Figure 3.3: Average list size vs. E_b/N_0 for the list Viterbi decoding simulation of a $\nu = 8$ TBCC concatenated with ELF 0xFF shown in Fig.3.2. © 2023 IEEE

Let p be the puncturing index with range $0 \leq p \leq n$, which indicates either that no bit is punctured ($p = 0$) or which bit is punctured $1 \leq p \leq n$.

Let $T_p(W)$ be the transition matrix of a trellis stage with puncturing according to puncturing index p. The matrix $T_0(W)$ is the same as the $T(W)$ discussed above. The matrices $T_p(W)$ for $1 \leq p \leq n$ have the power of W reduced by one for the entries where the p^{th} output bit, i.e. the punctured bit, is a 1. With this description of punctured transition matrices $T_p(W)$, let p_i be the puncturing index that describes the puncturing of the i^{th} trellis stage. The weight enumerator polynomial for the block code implemented by sending K information bits through an outer ELF code with 2^m states and then a 2^ν-state tail-biting convolutional code and then puncturing some number of bits is expressed as follows:

$$A(W) = \sum_{i=0}^{2^\nu - 1} e_i \prod_{j=1}^{K+m} T_{p_j}(W) e_i^T - 1. \tag{3.10}$$

Often, puncturing is designed according to a periodic pattern. If the period includes q trellis stages, then we can define a transition matrix $T_\pi(W)$ for the puncturing period as follows:

$$T_\pi(W) = \prod_{i=1}^{q} T_{p_i}(W)\,. \tag{3.11}$$

If the $K+m$ trellis stages are an integer number of puncturing periods, the weight enumerator

Table 3.1: Best ELFs $E(x)$ for $m = 0$ to $m = 12$ redundancy bits that maximize minimum distance for the $\nu = 8$ TBCC (561,753) for $K = 64$ message bits, $N = 2 \times (64 + m)$ transmitted bits and for $N = 2 \times 76$ transmitted bits, $K = 76 - m$ message bits.

	$K = 64, N = 2 \times (64 + m)$			$N = 2 \times 76, K = 76 - m$		
m	$E(x)$	$d_{\min}$	$A_{d_{\min}}$	$E(x)$	$d_{\min}$	$A_{d_{\min}}$
0	0x1	12	704	0x1	12	836
1	0x3	12	260	0x3	12	304
2	0x5	12	66	0x5	12	76
3	0xF	12	4	0xF	14	380
4	0x11	14	68	0x11	14	76
5	0x33	14	11	0x33	14	4
6	0x7F	16	210	0x55	14	2
7	0xFF	16	86	0x81	16	24
8	0x1AB	18	360	0x195	16	6
9	0x301	18	146	0x325	18	297
10	0x4F5	18	17	0x53D	18	21
11	0x9AF	20	300	0xE0D	18	2
12	0x1565	20	47	0x1565	20	47

of the punctured tail-biting convolutional code with an ELF can be expressed as

$$A(W) = \sum_{i=0}^{2^{\nu}-1} e_i T_{\pi}(W)^{(K+m)/q} e_i^T - 1. \tag{3.12}$$

3.4 A List Decoding Sieve to find the best ELF

This section provides a list decoding sieve method to find the ELFs that maximize the $d_{\min}$ of the concatenated code. This approach is computationally more efficient than the error event construction method of Yang [36]. The sieve performs serial list Viterbi decoding [20] with the (noiseless) all-zeros codeword as the received word. Codewords join the list in order

Table 3.2: Expurgated distance spectra for $m = 0$ (no expurgation) to $m = 12$ for the $\nu = 8$ (N=152,K=76) tail-biting convolutional mother code described by polynomial (561,753).

		Expurgated Distance Spectrum for $w \leq 20$				
m	$E(x)$	A_{12}	A_{14}	A_{16}	A_{18}	A_{20}
0	0x1	836	3800	21736	123880	732564
1	0x3	304	1900	11324	61788	367764
2	0x5	76	988	5776	32300	177840
3	0xF	0	380	3344	15656	90060
4	0x11	0	76	1824	8056	43320
5	0x33	0	4	752	4040	22854
6	0x55	0	2	214	2210	11569
7	0x81	0	0	24	1341	5910
8	0x195	0	0	6	461	2932
9	0x325	0	0	0	297	1449
10	0x53D	0	0	0	21	742
11	0xE0D	0	0	0	2	393
12	0x1565	0	0	0	0	47

of increasing Hamming weight.

To find the best ELF polynomial $E(x)$ of degree m, codewords are grouped into sets S_{w_i} according to their Hamming weight w_i. The first set has weight w_1 equal to the $d_{\min}$ of the inner code, and for each i, $w_i > w_{i-1}$. For each set S_{w_i}, the sieve method removes from contention all polynomials $E(x)$ that divide any message polynomial $m(x)$ that produces a codeword in S_{w_i}. The remaining ELF polynomials correspond to concatenated codes with $d_{\min} > w_i$. This continues until a weight w^* is reached that would cause all the remaining ELF polynomials to be removed from contention. From among the remaining polynomials $E(x)$ at weight w^*, select the $E(x)$ that achieves w^* with the smallest number of neighbors.

Table 3.1 shows ELFs found by the list decoding sieve for TBCCs, e.g., ELF 0x301

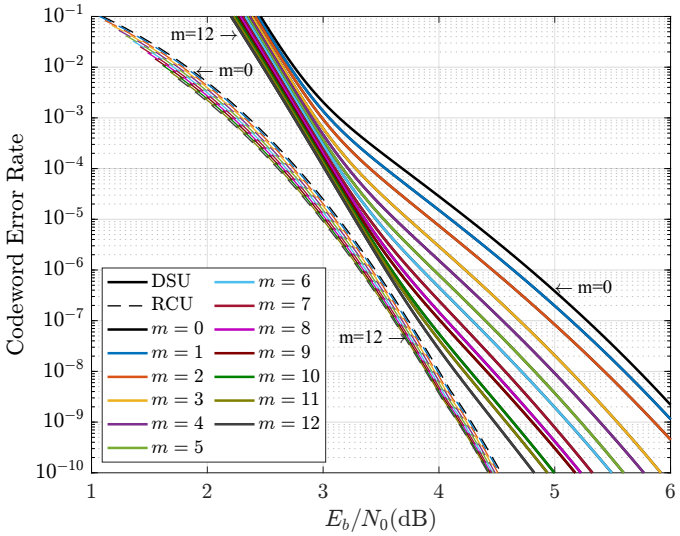

Figure 3.4: DSU and RCU bounds for ELF codes of Table 3.2. © 2023 IEEE

indicates $E(x) = 1 + x^8 + x^9$. The left half of the table holds K constant at 64 bits while the right half holds N constant at 152 bits. ELFs for $0 \leq m \leq 12$ are designed using the sieve approach. The $K = 64$ results for $3 \leq m \leq 10$ match the ELFs reported in [19]. When codeword length is held constant, all the ELFs are expurgating the same mother code. Table 3.2 shows how the the low-weight distance spectrum of the (152,76) mother code thins out as m increases.

Fig. 3.4 uses (3.9) and (3.3) to compute the DSU bounds for the expurgated tail-biting convolutional codes of Table 3.2. The corresponding RCU bounds are shown for reference. As m increases the DSU bound on CER performance steadily improves. Meanwhile the slight rate reduction does not significantly improve the CER performance of the RCU bound. As a result, the gap between the DSU and RCU bounds decreases as m increases. Fig. 3.5 further explores how ELFs reduce the gap between DSU and RCU bounds by showing the gap between DSU and RCU bounds at CER = 10^{-6} vs. m for all the ELFs in Table 3.1. We can see that in both cases the gap decreases as m increases culminating in a gap of 0.227 dB for the $m = 12$ ELF. For the $m = 12$ case, the two codes are actually identical (152, 64) codes. Recall from Fig. 3.3 that this increase in performance requires a minimal increase in average complexity as described in, e.g, [19, 35].

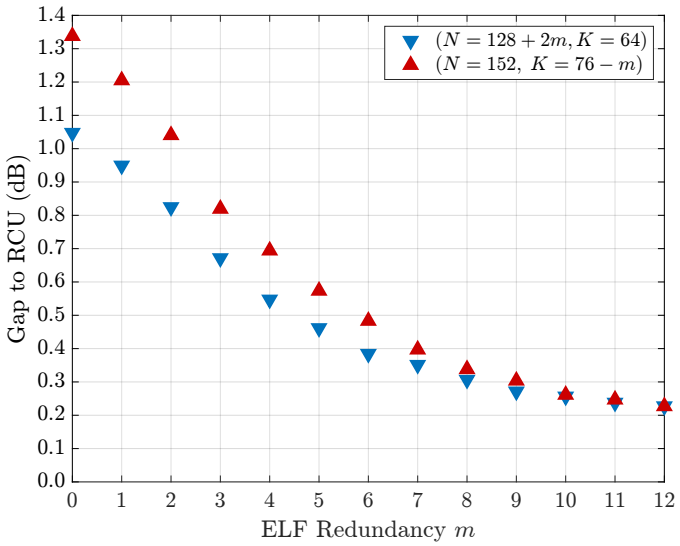

Figure 3.5: Gap at 10^{-6} between DSU and RCU bounds vs. m for Table 3.1 ELFs. © 2023 IEEE

3.5 A Puncturing Example: Rate-1/2 $K = 64$

This section applies Sec. 3.3.3 to explore a (128, 64) code created by puncturing 24 bits from the (152,64) block code formed by concatenating the $m = 12$ ELF 0x1565 from Tables 3.1 and 3.2 with the $\nu = 8$ tail-biting convolutional code. This example uses a periodic puncturing pattern with period $q = 19$, where six bits are punctured from each of the four periods that comprise the $K + m = 76$ trellis stages. The 19 puncturing indices for this periodic puncturing pattern are as follows:

$$[0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 2\ 0\ 1\ 0\ 0\ 2\ 0\ 0\ 2]$$

where puncturing index $p_i = 0$ indicates no puncturing, $p_i = 1$ indicates puncturing the output of the convolutional encoder polynomial 561 and $p_i = 2$ indicates puncturing the output of the convolutional encoder polynomial 753. We did not perform a fully exhaustive search even of puncturing patterns with period $q = 19$, but this pattern gives reasonable performance.

Fig. 3.6 shows the DSU bound for the $\nu = 8$ tail-biting convolutional code (561,753) with ELF 0x1565 and 24 bits punctured as described above. Also shown for comparison are the

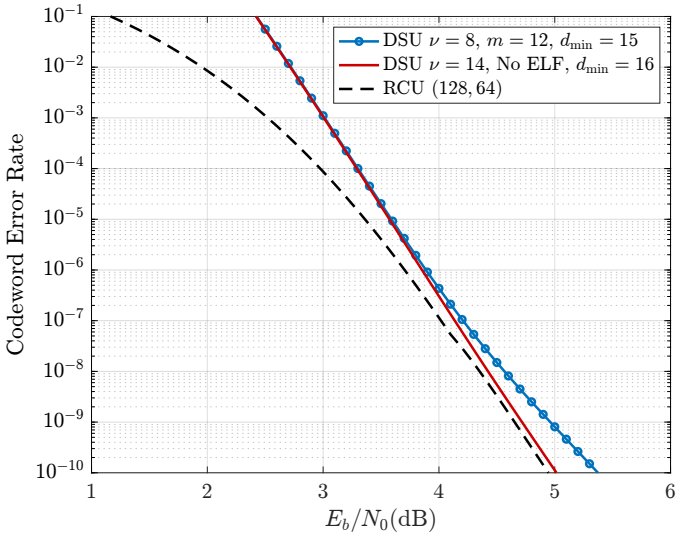

Figure 3.6: DSU bounds for two (128,64) codes and the (128,64) RCU bound. One code is the standard $\nu = 14$ tail-biting convolutional code (75063,56711) with no ELF and no puncturing. The other is the $\nu = 8$ tail-biting convolutional code (561,753) with ELF 0x1565 from Tables 3.1 and 3.2 with 24 bits punctured. © 2023 IEEE

(128,64) RCU bound and the DSU bound for the standard $\nu = 14$ tail-biting convolutional code (75063,56711) with no ELF and no puncturing. The $\nu = 14$ code has DSU to RCU gap of 0.15 dB at CER 10^{-6}. The $\nu = 8$ solution has a gap of 0.18 dB at CER 10^{-6}. At an operating point of CER $= 10^{-6}$ the $\nu = 8$ solution requires less average decoder complexity.

3.6 Conclusions

For short block lengths, expurgating linear functions (ELFs) transform a good inner code into a great concatenated code with a minimal increase in average complexity. This paper presented DSU bounding techniques that allow tight bounds on codeword error rate for ELF codes with and without puncturing and a sieve method for finding good ELFs efficiently.

CHAPTER 4

Linearity-Enhanced Serial List Decoding of Linearly Expurgated Tail-Biting Convolutional Codes

4.1 Abstract

With a sufficiently large list size, the serial list Viterbi algorithm (S-LVA) provides maximum likelihood (ML) decoding of a concatenated convolutional code (CC) and an expurgating linear function (ELF), which is similar in function to a cyclic redundancy check (CRC), but doesn't enforce that the code be cyclic. However, S-LVA with a large list size requires considerable complexity. This chapter exploits linearity to reduce decoding complexity for tail-biting CCs (TBCCs) concatenated with ELFs.

4.2 Introduction

4.2.1 Background

For a feedforward encoder with ν memory elements implementing a tail-biting convolutional code (TBCC) [27], the tail-biting condition can be enforced by setting the initial encoder state with the final ν message bits. By avoiding the ν overhead zero-termination (ZT) bits, TBCCs can achieve higher coding rates and show improvement in frame error rate (FER) vs. E_b/N_0 performance compared to the corresponding ZTCCs. Designing efficient decoders for TBCCs has been a popular research topic [12, 19, 22, 47, 48] and the construction of minimal trellises for TBCCs has also been extensively researched [28, 49–51].

In addition to tail-biting, the paradigm of concatenating a convolutional code with a cyclic

redundancy check (CRC) code further improves the decoding performance and has been applied widely since its proposal in 1994 [7]. Classically, the CRC code functions as an outer error-detecting code and verifies if a codeword has been correctly received. However, when used in conjunction with list decoding [20], the CRC acts as an outer code that expurgates low-weight codewords of the inner code [19, 32]. These expurgating linear functions (ELFs) need not be cyclic, and we will follow [4] and refer to them as ELFs. Recent works using the perspective of a CC concatenated with an ELF include [52], [5], [53], and [54].

Compared to zero-terminated codes, TBCCs require a higher decoding complexity to find the maximum likelihood (ML) trellis path that also satisfies the tail-biting condition. The exhaustive approach of performing a separate Viterbi decoding for each possible beginning/ending state finds the maximum-likelihood TB codeword but requires high complexity. A preferred ML decoder is proposed in Shankar *et al.*, where the ML codeword is found through a Viterbi step followed by an A* search [47]. In contrast, the wrap-around Viterbi algorithm (WAVA) [12] has a complexity that is only three to five times that of standard Viterbi decoding and is often only slightly suboptimal. Parallel and serial list decoding are two additional approaches [19, 20, 48] that provide ML decoding of a TBCC with less complexity than the exhaustive approach or the A* search, but more complexity than WAVA. When decoding a TBCC that is concatenated with an ELF, the serial list Viterbi decoding algorithm (S-LVA) is a natural approach [4, 5, 19], where the terminating conditions require that the trellis path under consideration is both a TB codeword and a codeword of the expurgated code produced by the ELF.

4.2.2 Contributions

This chapter exploits the linear nature of convolutional codes [2] to reduce the complexity required to decode a TBCC concatenated with an ELF. For linear codes, the relative position of all neighboring codewords to a reference codeword is the same regardless of the reference codeword. Once a single trellis path has been found through Viterbi search, the neighborhood around that codeword can be easily accessed by simple XOR operations using a pre-computed

list of offsets. This chapter presents two low-complexity decoders that utilize pre-computed lists to quickly explore nearby TB codewords of a codeword found by either the Viterbi algorithm or S-LVA.

The first decoder is the **offset sphere decoder**, which considers a large sphere of tail-biting codewords centered around the trellis path nearest to the received word. For this decoder, once the standard Viterbi algorithm finds the nearest (probably non-tail-biting) trellis path, the list of tail-biting codewords nearest to that trellis path is enumerated using a list of pre-computed offsets associated with the ending state difference (ESD) of the trellis path, which is the difference between the initial and final states of the trellis path. While the complexity is similar to standard Viterbi decoding, the FER benefits significantly from the TB and ELF constraints. However, for the sphere sizes we considered, the performance falls short of what can be achieved by S-LVA.

Our second decoder is the **list-of-spheres** decoder, which can achieve a near-ML total failure rate (TFR) while maintaining a low average list rank $\mathsf{E}[L]$ for ELF-TBCCs. TFR includes erasures as well as undetected errors. However, with a sufficiently large list size, the erasure probability approaches zero and TFR becomes the same as the undetected error rate. This decoder performs S-LVA and searches a small sphere of tail-biting codewords around each trellis path found by the S-LVA. For decoding to terminate, the squared Euclidean metric of the codeword of the expurgated TB code must be within a threshold value. It is shown that the proposed decoder performs within 0.125 dB of the standard S-LVA decoder while maintaining a significantly smaller list size. At low E_b/N_0, the list-of-spheres decoder achieves the random coding union (RCU) bound developed by Polyanskiy *et al.* [10], which is an upper bound on the error probability of the best code given the blocklength and codeword length.

We follow [2, Chapter 11] to describe the generator matrix $G(D) = [g^{(0)}(D), g^{(1)}(D), \dots g^{(n-1)}(D)]$, where each $g^{(i)}(D)$ is a polynomial of degree up to ν in delay element D associated with the i-th code stream, i.e.,

$$g^{(i)}(D) = g_v^{(i)} D^v + g_{v-1}^{(i)} D^{v-1} + \cdots + g_0^{(i)}, \tag{4.1}$$

where $g_j^{(i)} \in \{0,1\}$. Each $g^{(i)}(D)$ is represented in octal form. For instance, $G(D) = [D^3+D+1, D^3+D^2+D+1]$ can be concisely written as $G = (13, 17)$. The ELF polynomial is represented in hexadecimal, where its binary coefficients are written from the highest to lowest order. For instance, 0xD represents the polynomial $x^3 + x^2 + 1$. As an example in this chapter, we use the ELF-TBCC where a rate-1/2 TBCC with generator matrix $G = (561, 753)$ is concatenated with the degree-7 optimal ELF of 0xFF. The information length for this code is $K = 64$ bits and the codeword length is $N = 142$ bits. The codewords are BPSK modulated. The SNR is defined as $\gamma_s \triangleq 10\log_{10}(A^2)$ (dB), where A represents the BPSK amplitude and the noise is distributed as a standard normal.

4.2.3 My Work

My contributions to this chapter were mainly to the development of the novel decoding algorithms, alongside Beryl Sui.

4.2.4 Organization

Sec. 4.3 describes the offset sphere decoding algorithm and the construction of pre-computed neighboring codeword lists. The performance of the offset sphere decoder for various sphere sizes is explored for an example rate-1/2 ELF-TBCC. Sec. 4.4 introduces the list-of-spheres decoding algorithm for ELF-TBCCs. This section explores how varying termination requirements and the size of the spheres of TB codewords around each trellis path can affect the decoder's performance. We show simulation and complexity results comparing TFR performance and expected list rank $\mathsf{E}[L]$ for the proposed near-ML decoder and the standard S-LVA decoder for our example rate-1/2 ELF-TBCC. Finally, Sec. 4.5 concludes the chapter.

4.3 Offset Sphere Decoding

This section considers an offset sphere decoder where the decoder examines a single large sphere of TB codewords centered on the trellis path found by the Viterbi decoder. Section

4.3.1 explains the process of generating lists of pre-computed offsets associated with the ESD of the trellis path. Section 4.3.2 shows how these pre-computed offset lists are used to find nearby TB codewords in the offset sphere decoding algorithm. Section 4.3.3 shows simulation results for offset sphere decoding on the rate-1/2 $(561, 753)$ ELF-TBCC with blocklength of $N = 142$ bits, and compares performance to a Viterbi decoder as well as a standard S-LVA decoder.

4.3.1 Generating Lists of Neighboring Tail-Biting Codewords

Throughout this chapter, we use "trellis codewords" to refer to trellis paths identified by the Viterbi algorithm. A TB codeword is a trellis codeword that also satisfies the tail-biting constraint. An "ELF-TB codeword" is a TB codeword that is also a codeword of the expurgated code, i.e. it satisfies the ELF or CRC constraint. This subsection explains how we can identify the sphere of nearby TB codewords for any trellis codeword. We can then search that sphere of TB codewords to find the ELF-TB codeword closest to the received word.

Consider our example ELF-TBCC with rate-1/2 TBCC generator polynomials $(561, 753)$ and ELF 0xFF. Using the list decoding sieve described in [4], we can perform S-LVA on the rate-1/2 trellis allowing any starting state and any ending state and using the noiseless all-zeroes ELF-TB codeword as the received word. The list decoder will enumerate trellis codewords in order of increasing Hamming distance from the all-zeros codeword. For each trellis codeword identified by the list decoder, we compute the ESD of that trellis codeword and append it to the list of neighboring trellis codewords associated with that ESD. The list decoding sieve continues until each list associated with an ESD has L_N neighbors.

Since the all-zeros codeword is used as the received word, the list of neighboring codewords with a particular ESD also represent the list of offsets that can be added to any trellis codeword with the same ESD to find the list of L_N closest TB codewords. As a result, the list of neighboring TB codewords can be quickly found with bit-wise XOR operations. The multi-trellis approach described in [5] to perform S-LVA would be preferred when the desired

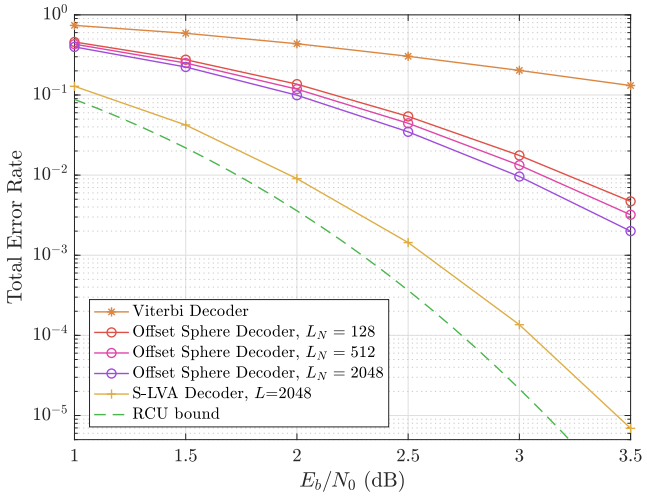

Figure 4.1: TFR vs. E_b/N_0 simulation results for a Viterbi decoder, an offset sphere decoder with a varying number of neighboring codewords $L_N = 128$, 512 and 2048, as well as an S-LVA decoder with a restricted maximum list size ($L_{max} = 2048$) so we can compare across the two approaches. The RCU bound for the $K = 64$, $N = 142$ code is shown as a dashed green line. The rate-1/2 ELF-TBCC used for simulation has generator polynomials $(561, 753)$ in octal and a degree-7 ELF of 0xFF, which adds seven ELF bits to the message.

L_N is large.

4.3.2 Searching the TB Sphere for the Closest ELF-TB Codeword

The offset sphere decoder first uses the standard Viterbi algorithm to find the closest trellis codeword and calculates its ESD. Then, each codeword on the associated ESD list is individually combined with the trellis codeword using the bit-wise XOR operation to produce one of the neighboring TB codewords. The sphere of neighboring TB codewords is searched to identify any ELF-TB codewords. If none are found, an erasure is declared. If one or more ELF-TB codewords are found, the ELF-TB codeword with the smallest squared Euclidean distance to the received codeword is returned as the selected codeword.

4.3.3 Simulation Results and Discussion

Fig. 4.1 shows simulation results comparing a standard Viterbi decoder, the proposed offset sphere decoder with varying sizes of L_N, and an S-LVA decoder with a maximum list size of $L_{max} = 2048$ for our example rate-1/2 ELF-TBCC. Compared to the standard Viterbi decoder, the offset sphere decoder has a significant improvement on the TFR. The decoding performance of the offset sphere decoder improves as L_N increases. For a sufficiently large L_N, performance should approach ML and therefore approach that of S-LVA with a large list size. However, the memory required to support extremely large values of L_N may make the offset sphere decoder unattractive for some applications. Fig. 4.1 shows that TFR for the offset sphere decoder is not comparable to S-LVA even when L_N is as large as the maximum list size of the S-LVA.

4.4 List-of-Spheres Decoder

To avoid the large L_N required for the offset sphere decoder to have comparable performance to S-LVA, this section proposes a second decoder, called the list-of-spheres decoder. This decoder searches a sequence of small spheres of neighboring TB codewords centered around the corresponding sequence of trellis codewords identified by S-LVA. Section 4.4.1 reviews S-LVA and introduces the list-of-spheres decoding algorithm. In Section 4.4.2, we define the parameter that controls the size of offset lists used in the list-of-spheres decoder. Section 4.4.3 presents termination conditions to reduce the likelihood of selecting a non-ML choice of ELF-TB codeword. Section 4.4.4 explores how different sphere sizes and thresholds affect the decoder's performance. TFR vs. E_b/N_0 simulation results of list-of-spheres decoders on the rate-1/2 $(561, 753)$ ELF-TBCC with varying termination conditions and sphere sizes are presented in Section 4.4.4. Section 4.4.5 demonstrates the improvement on the average list rank $\mathsf{E}[L]$ using a list-of-spheres decoder and discusses the decoding complexity, which is closely related to $\mathsf{E}[L]$ for list decoders.

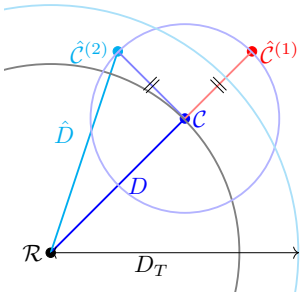

Figure 4.2: Illustration of the relationship between the received word $\mathcal{R}$, a trellis codeword $\mathcal{C}$ identified by S-LVA, and two TB codewords $\hat{\mathcal{C}}^{(1)}$ and $\hat{\mathcal{C}}^{(2)}$ equidistant from $\mathcal{C}$ found through a pre-computed list of offsets corresponding to the ESD of $\mathcal{C}$. The squared Euclidean distances D and $\hat{D}$ are labeled, as well as the threshold D_T. Note that $\hat{\mathcal{C}}^{(2)}$ satisfies D_T but the distance from $\hat{\mathcal{C}}^{(1)}$ to $\mathcal{R}$ is larger than D_T. © 2024 IEEE

4.4.1 Widening the Aperture of S-LVA with a List of Spheres

This section explains how the list-of-spheres decoder decreases complexity by widening the search aperture of S-LVA. Fig. 4.2 illustrates notation that will be useful in our discussion. The noisy received word is denoted by $\mathcal{R}$ and the trellis codeword most recently found by S-LVA is $\mathcal{C}$. As described in Sec. 4.3.1, a list of offsets is pre-computed for each possible ESD. For each trellis codeword identified by S-LVA, the list of offsets corresponding to its ESD is used to find neighboring TB codewords.

Define the i-th offset in the list of offsets for ESD e to be $\mathcal{N}_e^{(i)}$, where $e \in \{0, 1, \ldots, 2^v - 1\}$ is the ESD index of that list of neighboring offsets, and the union of all $\mathcal{N}_e^{(i)}$ for the pre-computed list for a given e to be $\mathcal{N}_e$. Let $\hat{C}$ denote the list of neighboring TB codewords computed using the list of offsets. Then, a list of neighboring TB codewords (in binary representation) can be obtained by the equation

$$\hat{\mathcal{C}}^{(i)} = \mathcal{C} \otimes \mathcal{N}_e^{(i)}, \tag{4.2}$$

where $\hat{\mathcal{C}}^{(i)}$ represents the i-th neighboring TB codeword in the sphere centered on trellis codeword $\mathcal{C}$, which has an ESD of e. The operator $\otimes$ represents the bit-wise XOR operation.

The binary codewords are transmitted using BPSK; for each codeword bit b the level

$(-E_b)^b$ is transmitted. Fig. 4.2 illustrates (in Euclidean space of BPSK transmission) the two TB codewords $\hat{\mathcal{C}}^{(1)}$ and $\hat{\mathcal{C}}^{(2)}$ closest to $\mathcal{C}$, which happen to be equidistant from $\mathcal{C}$.

Define $D \triangleq ||\mathcal{R} - \mathcal{C}||_2$ to be the Euclidean distance between the received word $\mathcal{R}$ and the BPSK representation of the trellis codeword $\mathcal{C}$ found by S-LVA. Similarly, define $\hat{D} \triangleq ||\mathcal{R} - \hat{\mathcal{C}}||_2$ to be the Euclidean distance between $\mathcal{R}$ and the BPSK representation of TB codeword $\hat{\mathcal{C}} \in \hat{C}$.

In [20], Seshadri and Sundberg proposed the S-LVA, where at most L_{max} most likely codewords are found in order of increasing distance from $\mathcal{R}$. As each codeword on the list is found, it is checked to see if it meets the terminating condition, which in [20] was passing a CRC.

When S-LVA is applied to an ELF-TBCC, many of the trellis codewords on the list are not even TB codewords. The list-of-spheres decoder presented in Algorithm 1 implements S-LVA but expands the aperture of consideration at each step to include the TB codewords closest to the trellis codeword identified by S-LVA. If the trellis codeword $\mathcal{C}$ found by S-LVA is an ELF-TB codeword, it is selected as the decoding result. If not, the decoder computes the list of neighboring TB codewords $\hat{C}$ for the corresponding ESD and searches that list of $\hat{\mathcal{C}}$ for ELF-TB codewords.

4.4.2 The Size of the Spheres

The size of the spheres of nearby TB codewords is controlled by the parameter $N_{neighbor}$, which specifies the number of distances of TB codewords permitted in the sphere. If $N_{neighbor} = 1$, only the nearest neighbors, the TB codewords with the smallest distance from the trellis codeword $\mathcal{C}$, are included in the sphere. If $N_{neighbor} = 2$, then the nearest neighbors and the next-nearest neighbors are included. Unlike the offset sphere decoder, where lists corresponding to different ESDs all include the same number of TB codewords, the number of TB codewords in the sphere can be different for different ESD values that induce different numbers of nearest neighbors.

Algorithm 1 List-of-Spheres Decoding Algorithm

Input: $\mathcal{R}$, $N_{neighbor}$, A, L_{max}

Output: Decoded codeword $\mathcal{C}^*$

while $L < L_{max}$ **do**
  Identify $\mathcal{C}$ by S-LVA
  if $\mathcal{C}$ is an ELF-TB codeword **then**
    Return $\mathcal{C}^* = \mathcal{C}$
  else
    Calculate ESD e of $\mathcal{C}$
    Compute $\hat{C}$ using $\mathcal{N}_e$
    if $\exists\ \hat{\mathcal{C}}^{(i)} \in \hat{C}$ that is an ELF-TB codeword **then**
      Compute $\hat{D}^{(i)} \triangleq ||\mathcal{R} - \hat{\mathcal{C}}^{(i)}||_2$
      if $\hat{D}^{(i)} \leq D_T$ and $\hat{D}^{(i)} < \hat{D}^{(j)} \forall j \neq i$ **then**
        Return $\mathcal{C}^* = \hat{\mathcal{C}}^{(i)}$
      end if
    end if
  end if
end while

4.4.3 A Threshold to Avoid Decoding Errors

Just as with the offset sphere decoder, there is a danger that the list-of-spheres decoder can find an ELF-TB codeword that is not the ML choice. To reduce the probability of terminating with the selection of a non-ML ELF-TB codeword, a threshold D_T is placed on ELF-TB codewords that are found when searching the sphere. Fig. 4.2 illustrates how this threshold is applied. When the search of the sphere identifies an ELF-TB codeword, we compute $\hat{D} \triangleq ||\mathcal{R} - \hat{\mathcal{C}}||_2$. We define a threshold distance D_T which can be a function of both the distance $D \triangleq ||\mathcal{R} - \mathcal{C}||_2$ and an "aperture" parameter A that controls how much further from $\mathcal{R}$ than $\mathcal{C}$ an identified ELF-TB codeword can be and still be admissible. The relationship between A and D_T is defined as $D_T = \sqrt{D^2 + A}$. Equivalently, A is the

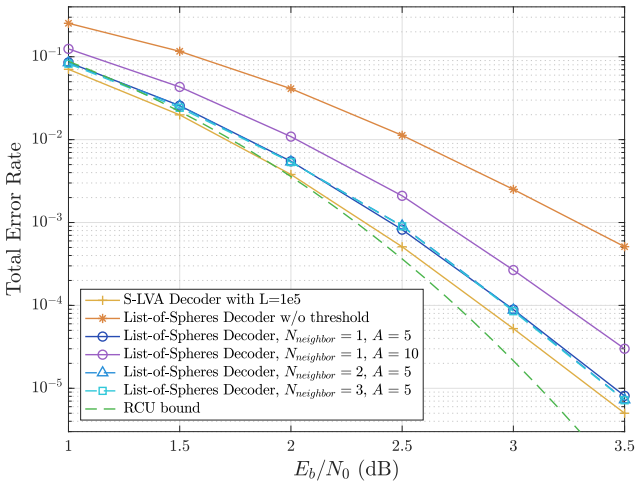

Figure 4.3: TFR vs. E_b/N_0 simulation results for the same ELF-TBCC as Fig. 1 using the list-of-spheres decoder for aperture parameters $A = 10$ with $N_{neighbor} = 1$ and $A = 5$ with $N_{neighbor} \in \{1, 2, 3\}$. The performance of a list-of-spheres decoder without any threshold and an S-LVA decoder with a large maximum list size ($L_{max} = 10^5$) are also presented. The RCU bound for the code is shown as a dashed green line.

difference between the squared Euclidean distances D^2 and $\hat{D}^2$. If the ELF-TB codeword $\hat{\mathcal{C}}$ satisfies the condition $\hat{D} \leq D_T$, it is admissible as a decoder result, and the algorithm terminates. If multiple admissible ELF-TB codewords are found while searching a sphere, the decoder returns the one closest to $\mathcal{R}$.

4.4.4 Selecting the Sphere Size $N_{neighbor}$ and the Threshold D_T

This section explores how the choices of the sphere size $N_{neighbor}$ and the threshold D_T affect the list-of-spheres decoder's performance. To explore performance empirically, simulations were performed using this example rate-1/2 ELF-TBCC, which has generator polynomials $(561, 753)$ in octal and a degree-7 ELF of 0xFF.

The aperture parameter A prevents the decoder from selecting neighboring TB codewords that are too far away from the received word $\mathcal{R}$. This restriction reduces the probability of selecting a non-ML decoding result. When $A = 0$, the list-of-spheres decoder is equivalent to

a regular S-LVA decoder. As A is increased and the "aperture" of our decoder opens, more TB codewords in the sphere become admissible. Thus, the expected list rank and decoding complexity are reduced because decoding terminates sooner, but non-ML codewords are more likely to be selected.

Fig. 4.3 shows simulated TFR vs. E_b/N_0 result for a list-of-spheres decoder without a threshold requirement, which is far off the RCU bound. The TFR performance improves considerably when the threshold requirement is added with an aperture parameter of $A = 10$. Even better TFR performance, within 0.125 dB of S-LVA, is achieved with an aperture parameter of $A = 5$ at a target TFR of 10^{-5}. At low E_b/N_0, the performance of a list-of-spheres decoder with $A = 5$ achieves the RCU bound.

To consider possible choices for $N_{neighbor}$, Fig. 4.3 shows simulation results for a list-of-spheres decoder with a fixed aperture parameter $A = 5$ and varying $N_{neighbor}$ values of 1, 2, and 3, as well as an S-LVA decoder with a sufficiently large list size ($L_{max} = 10^5$) that ensures ML decoding. With the aperture parameter preventing most selections of non-ML codewords, increasing $N_{neighbor}$ does not degrade TFR performance. However, as shown in Fig. 4.4, increasing $N_{neighbor}$ significantly reduces the expected list rank. For $A = 5$, increasing to $N_{neighbor} = 3$ reduces the expected list rank to become close to that of $A = 10$ with $N_{neighbor} = 1$.

4.4.5 Expected List Rank and Complexity

In [19], the authors provided the complexity expression for S-LVA of rate-$1/\omega$ ELF-TBCCs, where the overall average complexity of S-LVA can be decomposed into three components:

$$C_{\text{SLVA}} = C_{\text{SSV}} + C_{\text{trace}} + C_{\text{list}}. \tag{4.3}$$

C_{SSV} denotes the complexity of a standard soft Viterbi (SSV), C_{trace} denotes the complexity of the *additional* traceback operations required by S-LVA, and C_{list} denotes the average complexity of inserting new elements to maintain an ordered list of path metric differences.

For ELF-TBCCs, these metrics are evaluated by Eq. 4.4 through Eq. 4.7, where $\mathsf{E}[I]$ is

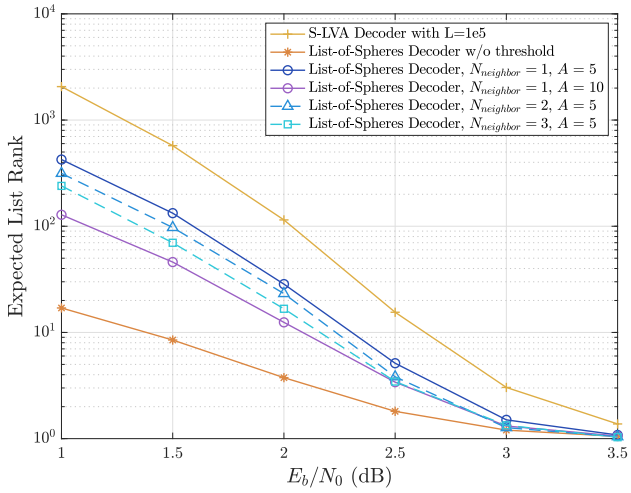

Figure 4.4: $\mathsf{E}[L]$ vs. E_b/N_0 simulation results for the same ELF-TBCC as Fig. 1 using the list-of-spheres decoder for aperture parameters $A = 10$ and $A = 5$ and $N_{neighbor} \in \{1, 2, 3\}$. The $\mathsf{E}[L]$ of a list-of-spheres decoder without any threshold and an S-LVA decoder with a sufficiently large maximum list size ($L_{max} = 10^5$) that ensures ML decoding are shown in solid yellow and orange, respectively. © 2024 IEEE

the expected number of insertions to maintain the sorted list of path metric differences.

$$C_{\text{SSV}} = 1.5(K+m)2^{v+1} + 2^v + 3.5(K+m) \tag{4.4}$$

$$C_{\text{trace}} = 3.5(\mathsf{E}[L]-1)(K+m) \tag{4.5}$$

$$C_{\text{list}} = \mathsf{E}[I]\log(\mathsf{E}[I]) \tag{4.6}$$

$$\mathsf{E}[I] \leq (K+m)\mathsf{E}[L] + 2^v - 1 \tag{4.7}$$

Since forming the neighbor lists can be done exclusively through XOR operations, and the list-of-spheres decoder uses relatively small neighbor lists, it has low complexity compared to the traceback and insertion components of S-LVA. Thus, C_{SLVA} of the list-of-spheres decoder is approximately the same as that of a regular S-LVA decoder, and lowering $\mathsf{E}[L]$ will result in a lower overall complexity.

In Fig. 4.4, the expected list ranks of a list-of-spheres decoder with varying $N_{neighbor}$ and aperture values are compared to that of a standard S-LVA decoder that achieves ML decoding. At low E_b/N_0, as the threshold increases, $\mathsf{E}[L]$ of the list-of-spheres decoder decreases substantially. The expected list rank of a list-of-spheres decoder without a threshold is around $\frac{1}{10}$ of those with thresholds, but this complexity reduction comes at the price of poor decoding performance. While $A = 10$ achieves the second lowest $\mathsf{E}[L]$, the list-of-spheres decoder with $A = 5$ still shows a much lower complexity than the S-LVA decoder, providing a considerable reduction in decoding complexity.

Fig. 4.4 also compares the expected list rank of list-of-spheres decoders with varying $N_{neighbor}$ and a fixed aperture $A = 5$. Despite their similar TFR performance, the decoders show how $\mathsf{E}[L]$ decreases with increasing values of $N_{neighbor}$, especially at low E_b/N_0. Including more neighboring codewords reduces $\mathsf{E}[L]$ because the decoder is able to find an ELF-TB codeword earlier by searching a larger sphere around a trellis codeword.

4.5 Conclusion

This chapter proposes two decoders based on the linearity of convolutional codes: an offset sphere decoder and a list-of-spheres decoder. The offset sphere decoder improves performance over the standard Viterbi decoder by around 1.5 dB, while maintaining a similar level of decoding complexity. The list-of-spheres decoder can closely approach ML performance while substantially reducing the expected list rank of the S-LVA. The list-of-spheres decoder with a smaller aperture of $A = 5$ and larger sphere size $N_{neighbor} = 3$ achieves the best tradeoff of decoding complexity and TFR performance. Future research directions include further exploration of how the distribution of codewords can be applied to determine optimal termination conditions and sphere sizing. Additionally, it may be possible to design a truly ML decoder with reduced complexity compared to S-LVA based on the linearity property of ELF-TBCCs. There are many aspects of the linearity approach that can be applied to enhance decoders of ELF-TBCCs.

CHAPTER 5

Conclusion

As has been shown in this thesis, the coding scheme of CCs concatenated with ELFs offers strong performance, closely approaching the RCU bound at practical operating points with low average decoder complexity. Further directions for research include further improvements of bounding techniques to analyze codes without the need for computationally intensive simulation, and further exploration of increasingly efficient decoding algorithms.

REFERENCES

[1] M. Karimzadeh and M. Vu, "Optimal CRC design and serial list Viterbi decoding for multi-input convolutional codes," in *2020 IEEE Global Commun. Conf.*, 2020, pp. 1–6.

[2] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Pearson, 2004.

[3] W. Sui, H. Yang, B. Towell, A. Asmani, and R. D. Wesel, "High-rate convolutional codes with CRC-aided list decoding for short blocklengths," in *ICC 2022 - IEEE Int. Conf. on Comm.*, 2022, pp. 98–103, © 2022 IEEE. Reprinted, with permission.

[4] R. D. Wesel, A. Antonini, L. Wang, W. Sui, B. Towell, and H. Grissett, "ELF codes: Concatenated codes with an expurgating linear function as the outer code," pp. 287–291, 2023, © 2023 IEEE. Reprinted, with permission.

[5] W. Sui, B. Towell, A. Asmani, H. Yang, H. Grissett, and R. D. Wesel, "CRC-aided high-rate convolutional codes with short blocklengths for list decoding," *IEEE Transactions on Communications*, vol. 72, no. 1, pp. 63–74, 2024, © 2024 IEEE. Reprinted, with permission.

[6] T. Yamada, H. Harashima, and H. Miyakawa, "A new maximum likelihood decoding of high rate convolutional codes using a trellis," *Elec. and Commun. in Japan Part I-commun.*, vol. 66, pp. 11–16, 1983.

[7] M. Rice, "Comparative analysis of two realizations for hybrid-ARQ error control," in *1994 IEEE Global Commun. Conf.*, 1994, pp. 115–119.

[8] "Universal mobile telecommunications system (UMTS); multiplexing and channel coding (FDD); 3GPP TS 25.212 version 7.0.0 release 7," European Telecommunications Standards Institute, Tech. Rep., 2006.

[9] "LTE; evolved universal terrestrial radio access (E-UTRA); multiplexing and channel coding; 3GPP TS 36.212 version 15.2.1 release 15," European Telecommunications Standards Institute, Tech. Rep., 2018.

[10] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.

[11] H. Ji, S. Park, J. Yeo, Y. Kim, J. Lee, and B. Shim, "Ultra-reliable and low-latency communications in 5G downlink: Physical layer aspects," *IEEE Wireless Commun. Mag.*, vol. 25, no. 3, pp. 124–130, 2018.

[12] L. Gaudio, T. Ninacs, T. Jerkovits, and G. Liva, "On the performance of short tail-biting convolutional codes for ultra-reliable communications," in *SCC 2017; 11th Int. ITG Conf. Syst., Commun., and Coding*, Feb. 2017, pp. 1–6.

[13] M. C. Coşkun, G. Durisi, T. Jerkovits, G. Liva, W. Ryan, B. Stein, and F. Steiner, "Efficient error-correcting codes in the short blocklength regime," *Physical Commun.*, vol. 34, pp. 66 – 79, 2019.

[14] C. Yue, M. Shirvanimoghaddam, B. Vucetic, and Y. Li, "A revisit to ordered statistics decoding: Distance distribution and decoding rules," *IEEE Trans. Inf. Theory*, vol. 67, no. 7, pp. 4288–4337, 2021.

[15] L. Dolecek, D. Divsalar, Y. Sun, and B. Amiri, "Non-binary protograph-based LDPC codes: Enumerators, analysis, and designs," *IEEE Trans. Inf. Theory*, vol. 60, no. 7, pp. 3913–3941, 2014.

[16] G. Liva, E. Paolini, B. Matuz, S. Scalise, and M. Chiani, "Short turbo codes over high order fields," *IEEE Trans. Commun.*, vol. 61, no. 6, pp. 2201–2211, 2013.

[17] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[18] E. Arıkan, "From sequential decoding to channel polarization and back again." [Online]. Available: http://arxiv.org/abs/1908.09594

[19] H. Yang, E. Liang, M. Pan, and R. D. Wesel, "CRC-aided list decoding of convolutional codes in the short blocklength regime," *IEEE Trans. on Information Theory*, vol. 68, no. 6, pp. 3744–3766, 2022.

[20] N. Seshadri and C. E. W. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Trans. Commun.*, vol. 42, no. 234, pp. 313–323, Feb. 1994.

[21] R. Schiavone, "Channel coding for massive IoT satellite systems," Master's thesis, Politechnic University of Turin (Polito), 2021.

[22] R. Shao, S. Lin, and M. Fossorier, "Two decoding algorithms for tailbiting codes," *IEEE Transactions on Communications*, vol. 51, no. 10, pp. 1658–1665, 2003.

[23] F. Soong and E.-F. Huang, "A tree-trellis based fast search for finding the n-best sentence hypotheses in continuous speech recognition," in *ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*, 1991, pp. 705–708 vol.1.

[24] M. Roder and R. Hamzaoui, "Fast tree-trellis list Viterbi decoding," *IEEE Trans. Commun.*, vol. 54, no. 3, pp. 453–461, Mar. 2006.

[25] R. Hinze, "Constructing red-black trees," 10 1999.

[26] A. Hasham, "A new class of priority queue organizations," Master's thesis, 1986, aAI0662089.

[27] H. Ma and J. Wolf, "On tail biting convolutional codes," *IEEE Trans. Commun.*, vol. 34, no. 2, pp. 104–111, Feb. 1986.

[28] R. Koetter and A. Vardy, "The structure of tail-biting trellises: minimality and basic principles," *IEEE Trans. Inf. Theory*, vol. 49, no. 9, pp. 2081–2105, Sep. 2003.

[29] R. E. Blahut, *Algebraic Codes for Data Transmission.* Cambridge University Press, 2003.

[30] R. Gallager, "A simple derivation of the coding theorem and some applications," *IEEE Trans. on Information Theory*, vol. 11, no. 1, pp. 3–18, 1965.

[31] J. Font-Segura, G. Vazquez-Vilar, A. Martinez, A. Guillén i Fàbregas, and A. Lancho, "Saddlepoint approximations of lower and upper bounds to the error probability in channel coding," in *2018 52nd Annual Conf. on Information Sciences and Systems (CISS)*, 2018, pp. 1–6.

[32] C. Y. Lou, B. Daneshrad, and R. D. Wesel, "Convolutional-code-specific CRC code design," *IEEE Trans. Commun.*, vol. 63, no. 10, pp. 3459–3470, Oct. 2015.

[33] H. Yang, S. V. S. Ranganathan, and R. D. Wesel, "Serial list Viterbi decoding with CRC: Managing errors, erasures, and complexity," in *2018 IEEE Global Comm. Conf. (GLOBECOM)*, 2018, pp. 1–6.

[34] H. Yang, E. Liang, H. Yao, A. Vardy, D. Divsalar, and R. D. Wesel, "A list-decoding approach to low-complexity soft maximum-likelihood decoding of cyclic codes," in *2019 IEEE Global Comm. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[35] E. Liang, H. Yang, D. Divsalar, and R. D. Wesel, "List-decoded tail-biting convolutional codes with distance-spectrum optimal CRCs for 5G," in *2019 IEEE Global Comm. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[36] H. Yang, L. Wang, V. Lao, and R. D. Wesel, "An efficient algorithm for designing optimal CRCs for tail-biting convolutional codes," in *2020 IEEE Int. Sym. Inf. Theory (ISIT)*, June 2020, pp. 1–6.

[37] J. King, A. Kwon, H. Yang, W. Ryan, and R. D. Wesel, "CRC-aided list decoding of convolutional and polar codes for short messages in 5G," in *ICC 2022 - IEEE Int. Conf. on Comm.*, 2022, pp. 92–97.

[38] J. King, W. Ryan, and R. D. Wesel, "CRC-aided short convolutional codes and RCU bounds for orthogonal signaling," in *GLOBECOM 2022 - 2022 IEEE Global Comm. Conf.*, 2022, pp. 4256–4261.

[39] D. Song, F. Areces, L. Wang, and R. Wesel, "Shaped TCM with list decoding that exceeds the RCU bound by optimizing a union bound on fer," in *GLOBECOM 2022 - 2022 IEEE Global Comm. Conf.*, 2022, pp. 4262–4267.

[40] J. King, "CRC-aided list decoding of short convolutional and polar codes for binary and nonbinary signaling," Master's thesis, University of California, Los Angeles (UCLA), 2022.

[41] L. Wang, D. Song, F. Areces, and R. D. Wesel, "Achieving short-blocklength RCU bound via CRC list decoding of TCM with probabilistic shaping," in *ICC 2022 - IEEE Int. Conf. on Comm.*, 2022, pp. 2906–2911.

[42] L. Wang, D. Song, F. Areces, T. Wiegart, and R. D. Wesel, "Probabilistic shaping for trellis-coded modulation with CRC-aided list decoding," *IEEE Trans. on Communications*, vol. 71, no. 3, pp. 1271–1283, 2023.

[43] R. Schiavone, R. Garello, and G. Liva, "Application of list Viterbi algorithms to improve the performance in space missions using convolutional codes," in *2022 9th Int. Workshop on Tracking, Telemetry and Command Systems for Space Applications (TTC)*, 2022, pp. 1–8.

[44] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[45] J. King, H. Yao, W. Ryan, and R. D. Wesel, "Design, performance, and complexity of CRC-aided list decoding of convolutional and polar codes for short messages." [Online]. Available: https://arxiv.org/abs/2302.07513

[46] A. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. on Communication Technology*, vol. 19, no. 5, pp. 751–772, 1971.

[47] P. Shankar, P. Kumar, K. Sasidharan, B. S. Rajan, and A. Madhu, "Efficient convergent maximum likelihood decoding on tail-biting trellises," *CoRR*, vol. abs/cs/0601023, 01 2006.

[48] J. King, W. Ryan, C. Hulse, and R. D. Wesel, "Efficient maximum-likelihood decoding for TBCC and CRC-TBCC codes via parallel list viterbi," pp. 141–145, 2023.

[49] A. Calderbank, G. Forney, and A. Vardy, "Minimal tail-biting trellises: the golay code and more," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1435–1455, 1999.

[50] S. Lin and R. Shao, "General structure and construction of tail biting trellises for linear block codes," in *2000 IEEE International Symposium on Information Theory (Cat. No.00CH37060)*, 2000, pp. 117–.

[51] D. Conti and N. Boston, "On the algebraic structure of linear tail-biting trellises," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2283–2299, 2015.

[52] A. Antonini, W. Sui, B. Towell, D. Divsalar, J. Hamkins, and R. D. Wesel, "Suppressing error floors in SCPPM via an efficient CRC-aided list viterbi decoding algorithm," pp. 221–225, 2023.

[53] B. Feng, Y. Yang, J. Jiao, and Q. Zhang, "On tail-biting polarization-adjusted convolutional (TB-PAC) codes and small-sizes list decoding," *IEEE Communications Letters*, vol. 27, no. 2, pp. 433–437, 2023.

[54] R. Schiavone, R. Garello, and G. Liva, "Performance improvement of space missions using convolutional codes by CRC-aided list viterbi algorithms," *IEEE Access*, vol. 11, pp. 55 925–55 937, 2023.