Title
Efficient Computation of Viterbi Decoder Reliability with an Application to Variable-Length Coding

Author
Baldauf, Alexander Makoto

Publication Date
2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Efficient Computation of Viterbi Decoder Reliability with an Application to Variable-Length Coding

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

by

Alexander Makoto Baldauf

2022

ABSTRACT OF THE THESIS

Efficient Computation of Viterbi Decoder
Reliability with an Application
to Variable-Length Coding

by

Alexander Makoto Baldauf
Master of Science in Electrical and Computer Engineering
University of California, Los Angeles, 2022
Professor Richard D. Wesel, Chair

This thesis compares the accuracy and complexity of Raghavan and Baum's Reliability Output Viterbi Algorithm (ROVA), Polyanskiy's accumulated information density (AID), and Fricke and Hoeher's approximation of ROVA. It turns out that AID is far less accurate than ROVA in practice. This thesis proposes codeword information density (CID), which modifies AID to improve its accuracy and leads to a lower-complexity implementation of ROVA. This thesis includes an analytical expression for the random variable describing the correct decoding probability computed by ROVA and uses this expression to characterize how the probabilities of correct decoding, undetected error, and negative acknowledgement behave as a function of the selected threshold for reliable decoding. This thesis examines both the complexity and the simulation time of ROVA, CID, AID, and the Fricke and Hoeher approximation to ROVA. This thesis also derives an expression for the union bound on the frame error rate for zero-terminated trellis codes with punctured symbols and uses it to optimize the order of symbol transmission in an incremental retransmission scheme. This thesis compares the performance of an incremental retransmission scheme using ROVA as a stopping condition to one that uses a CRC as a stopping condition. This thesis concludes by

applying the sequential differential optimization algorithm (SDO) to determine the transmission lengths in an incremental transmission scheme to maximize the throughput when limiting the maximum number of transmissions.

The thesis of Alexander Makoto Baldauf is approved.

Chandra J. Joshi

Gregory J. Pottie

Richard D. Wesel, Committee Chair

University of California, Los Angeles

2022

To my family,
who have always supported me
with love and understanding.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to acknowledge and give my heartfelt appreciation to all of the undergraduates who have assisted me in researching the material contained in this thesis. The contributions of Shakeh Kalantarmoradian, Alethea Sung-Miller, and Anreeta Saseetharran have been vital to this thesis.

Secondly, I would like to thank my advisor, Professor Richard Wesel, whose support and guidance has been invaluable throughout my time at UCLA as both an undergraduate and a graduate.

I would also like to thank Professor Chandrashekhar Joshi, who helped create the Fast Track To Success program within the ECE department at UCLA. Without the guidance of this program during my time as an undergraduate, I may never have chosen to obtain my M.S.

CHAPTER 1

Introduction

Short blocklength messages play a critical role in modern messaging ecosystems, where they are utilized in text messages, control messages, sensors in internet-of-things environments, and many other applications. This thesis addresses two questions involving short blocklength messages. The first question is how to tell if a short message is reliably transmitted. The second question is how to increase the efficiency of short blocklength messages in the context of having feedback.

Cyclic redundancy checks (CRCs) are often used to detect errors in convolutional codewords [LDW15, YRW18, KC04a, EST18, 3GP]. CRCs play an important role in many incremental redundancy hybrid automatic repeat requests (ARQ) [3GP, LMS07, CHI98, WWB18, Ric94] but add overhead that can be significant for short block-lengths. As pointed out by [Ric94], an alternative to using a CRC is to directly consider the reliability of the Viterbi decoder output as was proposed by Yamamoto and Itoh [YI80]. Ragavan and Baum [RB98] proposed the reliability-output Viterbi algorithm (ROVA) as an improvement to [YI80]. ROVA explicitly computes the probability that a Viterbi decoding decision is in error. This allows the receiver to set a threshold on the ROVA-computed probability to achieve a target undetected (codeword) error rate (UER) without requiring a CRC.

ROVA was used in [WCW15a,FH09,FH07] to decide whether to request additional redundancy in a hybrid ARQ without the need for a CRC. For [WCW15a], ROVA was adapted as described in [WMW14] for tail-biting convolutional codes. Although ROVA calculates codeword error probability exactly, it suffers from high complexity. Fricke and Hoeher [FH07] developed an approximation of ROVA that reduces complexity.

1.0.1 Overview

For zero-terminated convolutional codes (ZTCCs), this thesis explores an alternative to ROVA for controlling an incremental redundancy hybrid ARQ that is based on information density [PPV11]. Symbol-wise accumulated information density (AID) as proposed by Polyanskiy et al. [PPV11] sums the information density of each received symbol to provide a metric of codeword reliability with a much lower complexity than ROVA.

For ZTCCs, this thesis compares the accuracy of ROVA with that of symbol-wise AID as well as the ROVA approximation. After observing the low accuracy of symbol-wise AID, this thesis proposes codeword information density (CID) as a modification to symbol-wise AID. The CID also computes an information density, but instead of adding the density of each symbol, it computes a single information density for the entire received codeword. CID gives better accuracy than AID and turns out to be equivalent to ROVA. The ROVA approximation of [FH07] has similar accuracy to ROVA, but lacks the exactness of CID.

This thesis develops an analytical expression for the distribution of ZTCC ROVA values over an additive white gaussian noise (AWGN) channel with a fixed signal-to-noise ratio (SNR). From this distribution, the probability of correct, probability of error, and probability of negative acknowledgement (NACK) are computed using Monte Carlo simulation and shown to match the Viterbi simulation results. The complexity of each of the four reliability metrics examined in this thesis are then compared.

This thesis derives a method of computing the frame error rate union bound for punctured codes, where puncturing refers to decreasing the blocklength by withholding certain symbols from transmission as discussed in [Bla12, Hag88, BLW02]. This thesis uses this method to optimize the order of additional symbol transmission in incremental retransmission schemes. This thesis then extends the work by Williamson [WCW15a] to demonstrate that ROVA can be used as a reliability metric for incremental retransmission schemes at short blocklengths for 8 phase-shift keying (8-PSK) modulation. Additionally, this thesis demonstrates that using ROVA as a reliability metric for incremental retransmission outperforms using a CRC as a reliability metric at short blocklengths.

This thesis includes an additional discussion about an information density approach for optimizing the throughput of systems using incremental redundancy controlled by feedback. Polyanskiy's normal approximation combined with a linear model for the information gap of a rate-compatible code family provides an accurate characterization of the behavior of feedback systems employing practical codes such as convolutional codes. Especially for short message lengths on the order of $k < 50$ message bits, the proposed model is more accurate than Vakilinia's model in which the rate of first successful decoding has a Gaussian probability density function. This thesis demonstrates that a relatively small number of feedback transmissions (such as $m = 4$ or $m = 8$) can achieve a throughput similar to systems that transmit feedback after every symbol, i.e. $m = \infty$. The sequential differential optimization (SDO) algorithm selects the m best transmission lengths, or equivalently the M best instances at which to send feedback, when using ROVA as the metric for assessing decoder reliability with ZTCCs.

1.0.2 Organization

Chapter 2 explains incremental redundancy with stop-feedback systems. Chapter 3 explains Viterbi decoding, a well-known maxmimum likelihood decoding algorithm. Chapter 4 reviews the ROVA algorithm of [RB98]. Chapter 5 presents an analytical expression for the random variable describing the correct decoding probability computed by ROVA and uses this expression to characterize how the probabilities of correct decoding, undetected error, and negative acknowledgement behave as a function of the selected threshold for reliable decoding. Chapter 6 discusses utilizing information density as a stopping condition and covers Polyanskiy *et al's.* [PPV11] derivation of the achievable rate for stop-feedback codes. Chapter 7 reviews AID of [PPV11] and shows that it is much less predictive of reliable decoding than ROVA. Chapter 8 proposes CID as a modification of AID, shows that CID is equivalent to ROVA, and uses the CID perspective to compute ROVA with significantly less complexity than [RB98]. Chapter. 9 analyzes and compares the complexity of each of the four reliability metrics presented in this thesis. Chapter 10 develops the union bound on the frame error rate (FER) for ZTCCs with punctured symbols and describes a method to optimize the order

of symbol transmission for the results shown in Sec. 10.2. Sec. 10.2 produces an analogous plot for short-blocklength performance of a reliability-based retransmission scheme using either a CRC or ROVA as shown in [WCW15a], but for a higher order modulation compared to BPSK. Chapter 11 explains the premise behind the sequential differential optimization algorithm and extends the results of Section 10.2 by restricting the maximum number of transmissions.

CHAPTER 2

Incremental Redundancy with Stop Feedback

It is a well known result that feedback does not increase the asymptotic capacity of a memoryless channel, although it can increase the zero-error capacity of the channel [Sha56]. Despite this, utilizing feedback can offer other benefits to channels that have made it particularly useful in modern communications systems. In variable-length coding schemes where the blocklength is not fixed, feedback can drastically reduce the average blocklength required to converge to capacity [PPV11]. It can also reduce the error exponent, which determines the error probability as a function of blocklength [Bur76].

Polyanskiy et al. [PPV11] describe two types of feedback codes for discrete memoryless channels (DMC) where the conditional probability kernels are

$$P_{Y_i|X_1^i Y_1^{i-1}} = P_{Y_i|X_i} = P_{Y_1|X_1} \tag{2.1}$$

and finite input/output alphabets are $\mathcal{A}$ and $\mathcal{B}$ respectively. Formally, a (l, M, ϵ) variable-length feedback (VLF) code is defined by the following list of features for l a positive real number, M a positive integer, and $0 \leq \epsilon \leq 1$:

1. A finite alphabet $\mathcal{U}$ with $|\mathcal{U} \leq 3|$ and a probability distribution P_U on $\mathcal{U}$, which defines a random variable U that is revealed to both transmitter and receiver before the start of transmission. U is used to initialize the encoder and decoder with a common randomness.

2. A sequence of encoders $f_n : \mathcal{U} \times \{1, ..., M\} \times \mathcal{B}^{n-1} \to \mathcal{A}$ for $n \geq 1$. These define the channel inputs as

$$X_n = f_n(U, W, Y^{n-1}) \tag{2.2}$$

where Y^{n-1} is the sequence $\{Y_1, ..., Y_{n-1}\}$ of channel outputs and $W \in \{1, ..., M\}$ is the equiprobable message.

3. A sequence of decoders $g_n : \mathcal{U} \times \mathcal{B}^n \to \{1, ..., M\}$ which provides the best estimate of W at time n.

4. A non-negative integer random variable τ, which is a stopping time of the filtration $\mathbb{F}_n = \sigma\{U, Y_1, ..., Y_n\}$, where $\sigma(\cdot)$ is the sigma-algebra of events. τ also satisfies

$$\mathbb{E}[\tau] \leq l \tag{2.3}$$

Variable-length feedback with termination (VLFT) codes are defined identically to VLF codes, except that the 4^{th} condition is replaced with

4. A non-negative integer random variable τ, which is a stopping time of the filtration $\mathbb{F}_n = \sigma\{W, U, Y_1, ..., Y_n\}$, where $\sigma(\cdot)$ is the sigma-algebra of events. τ also satisfies

$$\mathbb{E}[\tau] \leq l \tag{2.4}$$

In variable-length feedback (VLF) codes, the decision to stop transmission is made purely by the decoder's observation of the channel outputs in a causal manner. In variable-length feedback with termination (VLFT) codes, the decision to stop transmission includes the knowledge of the pre-encoded message W in addition to the channel output observations up to the n^{th} symbol, Y^n. In VLFT, the transmitter decides when to stop transmitting based on feedback from the receiver, usually transmitted over a special noiseless channel. VLF codes are a special case of VLFT codes, where the stopping time is determined solely by the decoder. This thesis focuses on a special case of VLF codes called variable-length stop-feedback (VLSF) codes, where feedback is used only to tell the transmitter when to stop sending additional symbols. This is denoted by the encoder function satisfying $f_n(U, W, Y^{n-1}) = f_n(U, W)$. In other words, the encoder does not use previous channel observations Y^{n-1} to generate the encoded input X_n given a message W.

The fundamental limit of channel coding with feedback for VLF and VLFT codes is given by the following

$$M^*_{\text{VLF}}(l,\epsilon) = \max\{M : \exists(l, M, \epsilon) - \text{VLF code}\} \tag{2.5}$$

$$M^*_{\text{VLFT}}(l,\epsilon) = \max\{M : \exists(l, M, \epsilon) - \text{VLFT code}\} \tag{2.6}$$

Polyanskiy derives several theorems in [PPV11] that demonstrate how effective stop-feedback is in helping codes approach capacity with significantly smaller blocklength than non-feedback codes. [PPV11] defines the error-capacity under variable-length coding without feedback as

$$[\![C_\epsilon]\!] = \frac{C}{1-\epsilon}, \epsilon \in (0,1). \tag{2.7}$$

where C is the capacity of the channel and ϵ is the error probability. [PPV11] goes on further to arrive at a an expression for capacity for both feedback and non-feedback systems:

$$\lim_{l\to\infty} \frac{1}{l} \log(M(l,\epsilon)) \tag{2.8}$$

where $\log(M(l,\epsilon))$ is bounded by

$$\frac{lC}{1-\epsilon} - \log l + O(1) \leq \log(M_{\text{VLF}}(l,\epsilon)) \leq \frac{lC}{1-\epsilon} + O(1) \tag{2.9}$$

for stop-feedback codes and $\log(M(n,\epsilon))$ is equal to

$$\log(M(n,\epsilon)) = nC - \sqrt{nV}Q^{-1}(\epsilon) + O(\log(n)) \tag{2.10}$$

for non-feedback codes, where V is the channel dispersion and Q^{-1} is the inverse of the standard Q-function. Note that the limit of M for both cases results in the same $[\![C_\epsilon]\!]$, which shows that feedback does not increase the error capacity. With stop-feedback, the back-off from the error-capacity is governed by the $\frac{\log l}{l}$ term, while the non-feedback back-off is primarily governed by the $\frac{1}{\sqrt{l}}$ term. Thus, the primary advantage of stop-feedback systems over systems without feedback is a significant reduction of the penalty for short blocklength. Stop-feedback systems where feedback is only used to let the encoder know that the decoder has made its final decision are often used in practice with acknowledgement (ACK) and negative acknowledgement (NACK) feedback messages.

This thesis utilizes several different methods of assessing whether to terminate transmission. The first is to perform a cyclic redundancy check (CRC), where both receiver and transmitter have access to a CRC polynomial. The transmitter performs polynomial long division using the CRC polynomial to determine the check bits for the transmission, which the transmitter appends to the end of the message. After receiving the message, the receiver performs polynomial long division using the CRC polynomial on the decoded message. If the result of the long division is zero, then the CRC check passes. CRCs will be used in the results shown in Fig. 10.1. Otherwise, the receiver decides additional bits are necessary for transmission. The other methods will be covered in Chapters 4, 7, and 8.

CHAPTER 3

Viterbi Decoding

An n bit sequence can take 2^n possible values. If this n bit sequence is transmitted across a noisy channel, computing the likelihood of observing the sequence from each of these possible values scales exponentially with n. Viterbi decoding allows a method of computing the highest likelihood sequence that will results in the observed sequence without explicitly tracking all 2^n possible sequences. Viterbi decoding is a type of convolutional decoder that performs maximum-likelihood decoding [Vit67].

The Viterbi algorithm for zero terminated convolutional codes (ZTCC) with constraint length v and number of transmitted symbols n can be visualized as a trellis diagram where each row $\in \{0, ..., 2^v - 1\}$ in the trellis represents a single possible state that the convolutional encoder can take. The transition between each column $\in \{0, ..., n\}$ represents one processed symbol as time. For each possible state transition that is possible in the encoder, the corresponding nodes in the trellis are connected by branches that take on the output values of that state transition. Each possible sequence of n bits is represented by a path through the branches in the diagram, and the encoded output of the n bits is equivalent to the concatenated values of each branch. For ZTCCs, each possible sequence starts at state 0. Fig. 3.1 shows an example trellis diagram for $v = 2$ and $n = 4$.

The Viterbi Algorithm works as follows for a zero-terminated convolutional code: The decoder examines a received sequence of length n. The decoder computes a metric for each accessible branch starting from state 0 and continues this process for each received symbol, forming paths that represent possible transmitted sequences. If multiple paths enter the same node, the path with the better metric is kept as a survivor path while the others are discarded. At any given moment, the Viterbi algorithm only has to store 2^v paths through

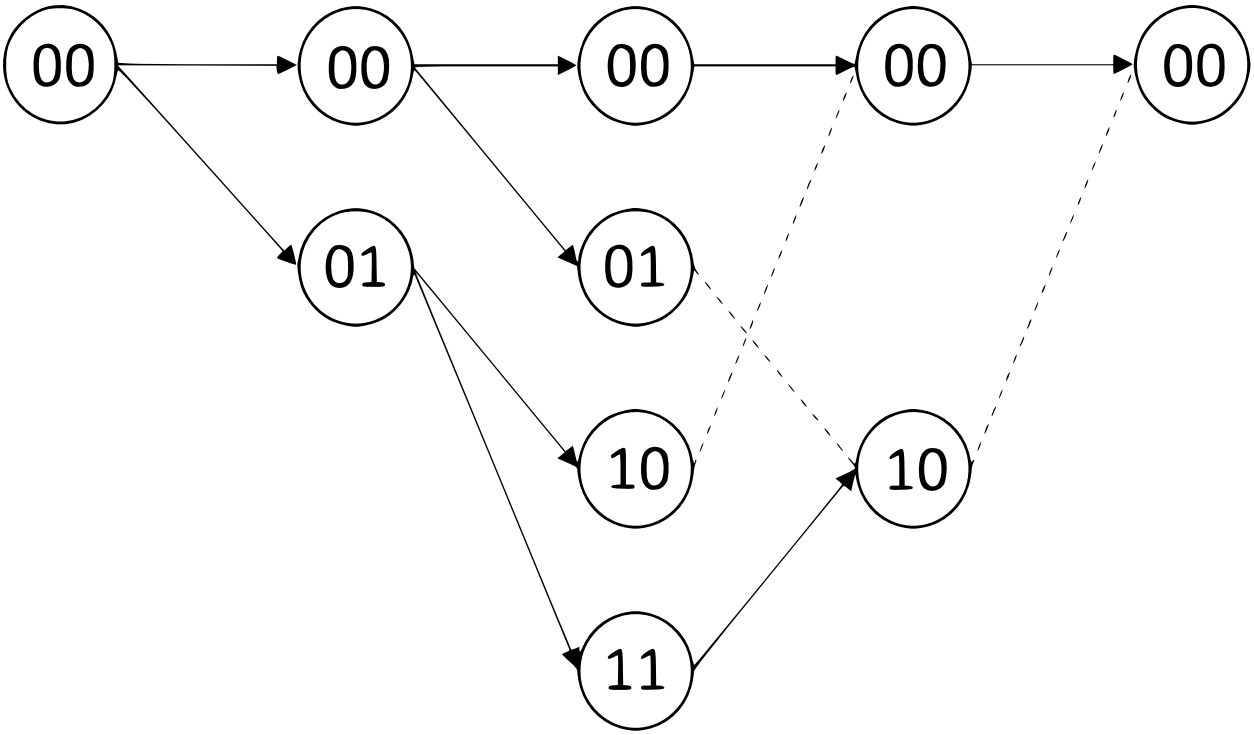

Figure 3.1: A four-state trellis that begins at state $s_0 = 0$ and terminates at state $s_4 = 0$ after four symbol transmissions. Solid branches represent survivor paths while dashed branches were rejected by the Viterbi algorithm

This figure was previously presented in part in an IEEE TCOM journal paper [BBK22].

the trellis, with a guarantee that one of the stored paths represents the maximum likelihood path. For ZTCCs, the last v bits are zeros to force the encoding state back to the zero state. By the time the last v bits are processed, all but one path has been discarded, and the remaining path is the maximum likelihood sequence. Fig. 3.1 shows the process of path elimination until there is only one survivor path.

Viterbi decoding algorithms can be either hard-decision or soft-decision algorithms. In hard-decision algorithms, the Viterbi decoder utilizes the hamming distance between the received sequence and the sequence through the trellis as the metric when comparing two paths. In soft-decision algorithms, the Viterbi decoder acts on the reliability of each received symbol. It uses squared euclidean distance when comparing two paths. This thesis exclusively uses the soft-decision Viterbi decoding algorithms.

CHAPTER 4

Reliability Output Viterbi Algorithm

[1] Most implementations of the Viterbi algorithm are performed in the logarithmic domain so that the products of the path metrics become a sum of products. This reduces complexity and enables fixed-point implementation. However, operating in the logarithmic domain makes the computation of exact probabilities such as ROVA more difficult, as the sum of products of the path metrics must be tracked in addition to the product. This thesis does not use the logarithmic domain for this reason. There are approximation methods that could be employed for the logarithmic domain as described in [BCJ74], but this remains a possible topic for future research.

As described in [RB98], ROVA finds the probability that the n_c-symbol codeword $\hat{x}^{n_c}$ selected by maximum likelihood Viterbi decoding is also the transmitted codeword $x_t^{n_c}$. In general, this thesis uses x^n as a shorthand for the sequence $x_1, \ldots, x_n$. Given a received noisy sequence $y^{n_c} = x_t^{n_c} + z^{n_c}$, the probability that $\hat{x}^{n_c} = x_t^{n_c}$ can be expressed as follows:

$$P(\hat{x}^{n_c} = x_t^{n_c} | y^{n_c}) = \frac{P(\hat{x}^{n_c}) f_{Y|X}(y^{n_c}|\hat{x}^{n_c})}{\sum_{x^{n_c} \in \mathcal{C}} P(x^{n_c}) f_{Y|X}(y^{n_c}|x^{n_c})} \tag{4.1}$$

$$= \frac{f_{Y|X}(y^{n_c}|\hat{x}^{n_c})}{\sum_{x^{n_c} \in \mathcal{C}} f_{Y|X}(y^{n_c}|x^{n_c})} \tag{4.2}$$

where $\mathcal{C}$ is the set of valid codewords and $f_{Y|X}(y^{n_c}|x^{n_c})$ is the conditional probability density for y^{n_c} if x^{n_c} were the transmitted sequence. The simplification from (4.1) to (4.2) follows from the assumption that all codewords are *a priori* equally likely.

A natural application of ROVA is to set a threshold on the ROVA value (4.2) computed as in [RB98] and consider codewords with a ROVA value below the threshold as erasures

[1] This chapter was previously presented in part in an IEEE TCOM journal paper [BBK22].

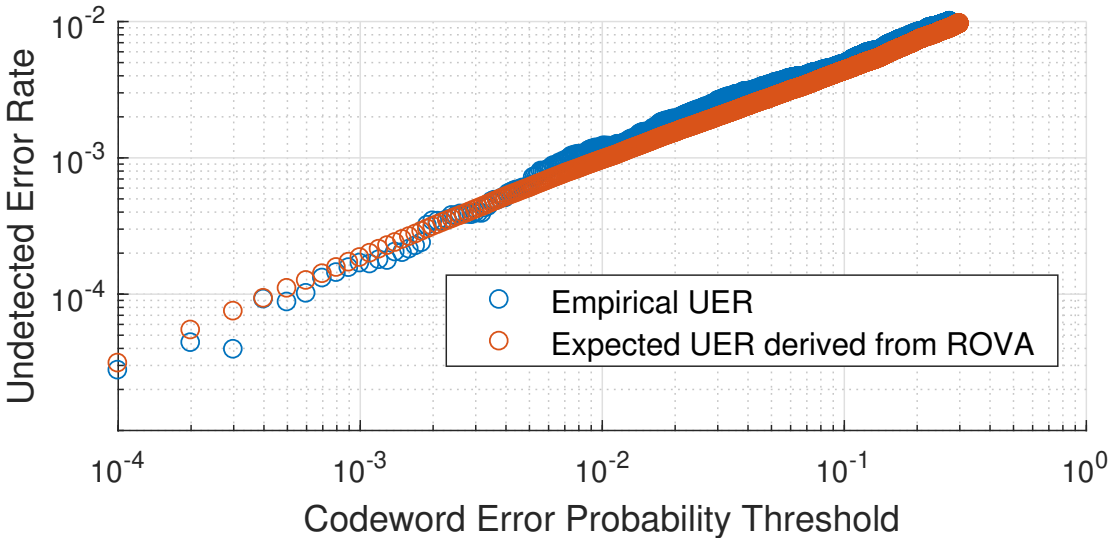

Figure 4.1: Graph of empirical and expected undetected codeword error rate (UER) as a function of the ROVA threshold for 100,000 decodings of the 4-state, rate-1/2 convolutional code with $k = 128$ message bits at SNR 4.5 dB.

This figure was previously presented in part in an IEEE TCOM journal paper [BBK22].

because they are not sufficiently reliable. Fig. 4.1 shows how varying the threshold can control the UER for an example 4-state rate-1/2 convolutional code with {101,111}, which is (5,7) in octal, as described in [RL09]. The empirical UER achieved with a ROVA threshold is shown for threshold values from $P(\hat{x}^{n_c} = x_t^{n_c}|y^{n_c}) = 0.7$ to $1 - 10^{-4}$ in increments of 10^{-4}.

Also shown is the expected UER associated with the threshold, which is computed as the UER implied by the empirical average of observed ROVA values that exceed the threshold. There is excellent agreement between the observed and expected UER. The average $P(\hat{x}^{n_c} = x_t^{n_c}|y^{n_c})$ will be substantially higher than the threshold because the threshold is the lowest acceptable value. As a result, the empirical UER achieved when applying a particular UER threshold is significantly below that UER threshold.

4.0.1 Computing ROVA as in [RB98]

Algorithm 1 describes the procedure for computing the ROVA value (4.2) as described in [RB98]. Consider a trellis with 2^v states defined by the set $\mathcal{S} = \{0, 1, \ldots 2^v - 1\}$. Let s_m be the trellis state after the m^{th} transmitted symbol. Let the trellis be initialized to state

$s_0 = 0$ and assume that terminating input bits drive the state back to $s_n = 0$ at the last transmitted symbol x_{n_c}. For each received symbol y_m for $m \in \{1, 2, \ldots n_c\}$ and for every possible value i of the trellis state s_m in $\mathcal{S}$, two probabilities are computed in [RB98]: P_i^m and $\bar{P}_i^m$. We now define these two probabilities.

Let $\hat{x}^m(i)$ be the symbol sequence corresponding to the Viterbi survivor path terminating at state i after the m^{th} transmitted symbol. P_i^m is $P(s_m = i, \hat{x}^m(i) = x_t^m | y^m)$, which is the probability that i is the correct state after the m^{th} transmitted symbol and that the Viterbi algorithm has correctly identified the survivor path to state i so that $\hat{x}^m(i) = x_t^m$. $\bar{P}_i^m$ is $P(s_m = i, \hat{x}^m(i) \neq x_t^m | y^m)$, which is the probability that i is the correct state at symbol m but the Viterbi algorithm has not correctly identified the survivor path to state i so that $\hat{x}^m(i) \neq x_t^m$. Thus $\bar{P}_i^m$ is the probability that Viterbi decoding has incorrectly pruned away the transmitted sequence x_t^m, which is a path to state i, after the m^{th} transmitted symbol.

For each m, Algorithm 1 makes use of the scaling factor

$$\Delta_m = \frac{\sum_{\mathcal{T}} \gamma_1 \gamma_2 \cdots \gamma_m}{\sum_{\mathcal{T}} \gamma_1 \gamma_2 \cdots \gamma_{m-1}}\,, \tag{4.7}$$

where γ_m is the branch metric for the m^{th} symbol associated with one of the paths in the trellis $\mathcal{T}$ so that $\gamma_1 \gamma_2 \ldots \gamma_m$ is a path metric for one of the paths in the trellis $\mathcal{T}$ and $\sum_{\mathcal{T}} \gamma_1 \gamma_2 \ldots \gamma_m$ is the sum of all the path metrics in the first m stages of trellis $\mathcal{T}$ regardless of whether they are survivors in the Viterbi algorithm.

For a ZTCC, when $m = n_c$, the state is forced to zero by terminating input bits so that $P_0^{n_c} + \bar{P}_0^{n_c} = 1$, and $\bar{P}_0^{n_c}$ is the probability that the codeword selected by Viterbi is incorrect. For $m < n_c$ and a particular state $i \in \mathcal{S}$, $P_i^m + \bar{P}_i^m$ will generally be less than one. These values must be summed over all states to account for all the probability:

$$\sum_{i=0}^{2^v - 1} \left(P_i^m + \bar{P}_i^m \right) = 1\,. \tag{4.8}$$

Let $N_s = |\mathcal{S}|$ be the number of states, and let N_b be the number of branches entering each state. Table .1 describes the complexity of Algorithm 1 for processing one trellis stage assuming all branches in the trellis are active. Step 2 requires $N_s N_b$ metric computations. Step 3 requires N_s additions to compute $P_i^{m-1} + \bar{P}_i^{m-1}$ and $N_s N_b$ additional multiplications

Algorithm 1: An algorithmic adaptation of the ROVA Algorithm described in [RB98].

Initialization: For $i, j \in \mathcal{S}$, let $\mathcal{T}_m$ be the set of valid trellis branches possible during transmission of the m^{th} symbol. Each such trellis branch is defined by the ordered pair (i, j) where i is the origin state and j is the destination state. Note that this set is smallest at $m = 1$ when there are only 2^k branches emanating from $s_0 = 0$ to s_1 and at $m = n_c$ when there are only 2^k branches entering $s_n = 0$. Initialize $m = 0$, $P_0^0 = 1$ and $\bar{P}_0^0 = 0$

Iterations: The calculation of (4.2) in [RB98] proceeds as follows:

1. $m = m + 1$

2. For each valid branch $(i, j) \in \mathcal{T}_m$ compute metrics

$$\gamma_m(i, j) = f\big(y_m | x_m(i, j)\big) \tag{4.3}$$

   where $x_m(i, j)$ is the symbol transmitted on branch (i, j), and $f\big(y_m | x_m(i, j)\big)$ is the conditional probability that we receive sequence y_m at stage m if $x_m(i, j)$ were transmitted.

3. Compute the scaling factor

$$\Delta_m = \sum_{(i,j)\in\mathcal{T}_m} \gamma_m(i, j)(P_i^{m-1} + \bar{P}_i^{m-1})\,. \tag{4.4}$$

4. For each $j \in \mathcal{S}$ with branches $(i, j) \in \mathcal{T}_m$ where Viterbi has identified branch (i^*, j) to be the survivor branch to j compute

$$P_j^m = \Delta_m^{-1}\gamma_m(i^*, j)P_{i^*}^{m-1} \tag{4.5}$$

$$\bar{P}_j^m = \Delta_m^{-1} \sum_{\{i:(i,j)\in\mathcal{T}_m\}} \gamma_m(i, j)(P_i^{m-1} + \bar{P}_i^{m-1}) - P_j^m \tag{4.6}$$

5. if $m = n_c$ conclude by reporting the ROVA value of $P_0^{n_c}$ and the probability of codeword error as $\bar{P}_0^{n_c} = 1 - P_0^{n_c}$, otherwise, go to step 1.

and additions to compute Δ_m. Step 4 requires one multiplicative inverse computation to produce Δ_m^{-1} and then $2 \times N_s$ multiplications to compute $P_{j^*}^m$ and $N_s N_b$ multiplications and additions to compute $\bar{P}_{j^*}^m$.

CHAPTER 5

Derivation of the ROVA Distribution

[1] An analytical expression for the distribution of $P_0^{n_c}$ reveals the relationship between the selected threshold and the induced UER (as shown in Fig. 4.1) and between the selected threshold and the induced throughput. The analysis below assumes BPSK symbols 1 and -1 are transmitted over an additive white Gaussian noise (AWGN) channel with noise variance σ^2.

Consider the computed conditional pdf $f_{Y|X}(y^{n_c}|\hat{x}^{n_c})$ in (4.2) as a random variable F and recall that for an AWGN channel it is computed as

$$F = \prod_{i=1}^{n_b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \hat{x}_i)^2}{2\sigma^2}} \tag{5.1}$$

where n_b is the number of binary symbols in x^{n_c}. For example, with a rate-1/3 convolutional code, $n_b = 3n_c$. If $\hat{x}^{n_c} = x_t^{n_c}$, using a subscript to denote the Hamming distance $d_H(\hat{x}^{n_c}, x_t^{n_c}) = 0$,

$$F_0 = \prod_{i=1}^{n_b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z_i^2}{2\sigma^2}} \tag{5.2}$$

where z_i is the AWGN in the i^{th} symbol. If $d_H(\hat{x}^{n_c}, x_t^{n_c}) = 1$, with the one difference bit in the j^{th} symbol, then

$$F_1 = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z_j+2)^2}{2\sigma^2}} \prod_{i=1, i\neq j}^{n_b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z_i^2}{2\sigma^2}} \tag{5.3}$$

$$= e^{-\frac{4+4z_j}{2\sigma^2}} \prod_{i=1}^{n_b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z_i^2}{2\sigma^2}} \tag{5.4}$$

$$= e^{-\frac{4+4z_j}{2\sigma^2}} F_0 \tag{5.5}$$

[1] This chapter was previously presented in part in an IEEE TCOM journal paper [BBK22].

where the mean of Gaussian describing the j^{th} symbol in (5.3) is shifted by the difference between the true and decoded values of x_j. For our BPSK modulation, this difference is always 2. This can be generalized to any Hamming distance. For $d_H(\hat{x}^{n_c}, x_t^{n_c}) = m$,

$$F_m = e^{-\frac{4m+\sum_{\ell=1}^{m} 4z_\ell}{2\sigma^2}} F_0 \tag{5.6}$$

Because Viterbi decoding only considers valid codewords $x^{n_c} \in \mathcal{T}$, the multiplicity of each possible value of $d_H(\hat{x}^{n_c}, x_t^{n_c})$ is a function of the specific convolutional code used to encode the message, and for a terminated trellis $d_H(\hat{x}^{n_c}, x_t^{n_c})$ has some maximum value D. Let A_m be the number of valid codewords $\hat{x}^{n_c}$ with $d_H(\hat{x}^{n_c}, x_t^{n_c}) = m$, which by linearity is the number of valid codewords with Hamming weight u:

$$A_u = \left|\{\hat{x}^{n_c} \in \mathcal{T} : d_H(\hat{x}^{n_c}, x_0^{n_c}) = u\}\right|, \tag{5.7}$$

where $x_0^{n_c}$ is the transmitted codeword for the all-zeros input.

Viterbi selects the correct codeword with high probability, and when it doesn't the selected $\hat{x}^{n_c}$ usually has similar value of $f_{Y|X}(y^{n_c}|\hat{x}^{n_c})$. Thus we can approximate $f_{Y|X}(y^{n_c}|\hat{x}^{n_c})$ with F_0 so that

$$P_0^{n_c} \approx \frac{F_0}{F_0 \sum_{u=0}^{U} A_u e^{-\frac{4u+\sum_{\ell=1}^{u} 4z_\ell}{2\sigma^2}}}, \tag{5.8}$$

which can also be expressed as

$$P_0^{n_c} \approx \left(1 + \sum_{u=1}^{U} A_u e^{-\frac{4u+\sum_{\ell=1}^{k} 4z_\ell}{2\sigma^2}}\right)^{-1}. \tag{5.9}$$

The expression for $P_0^{n_c}$ given in (5.9) includes a sum of U log-normal random variables. Because the magnitude of the terms decreases rapidly, summing a few of the most significant terms gives a good estimate. For a convolutional code with {117, 127, 155} as described in [WCW15b] as well as Table 12.1 of [LC04], Fig. 5.1 compares the cumulative histogram of $P_0^{n_c}$ found by a simulation of Viterbi decoding with $P_0^{n_c}$ computed as described in Algorithm 4 with the cumulative histogram of $P_0^{n_c}$ given in (5.9) generated by Monte Carlo using $U = 21$, using seven active terms for $u = 15$ to $u = 21$ and neglects terms with $u > 21$. These seven active terms provide an excellent approximation in Fig. 5.1.

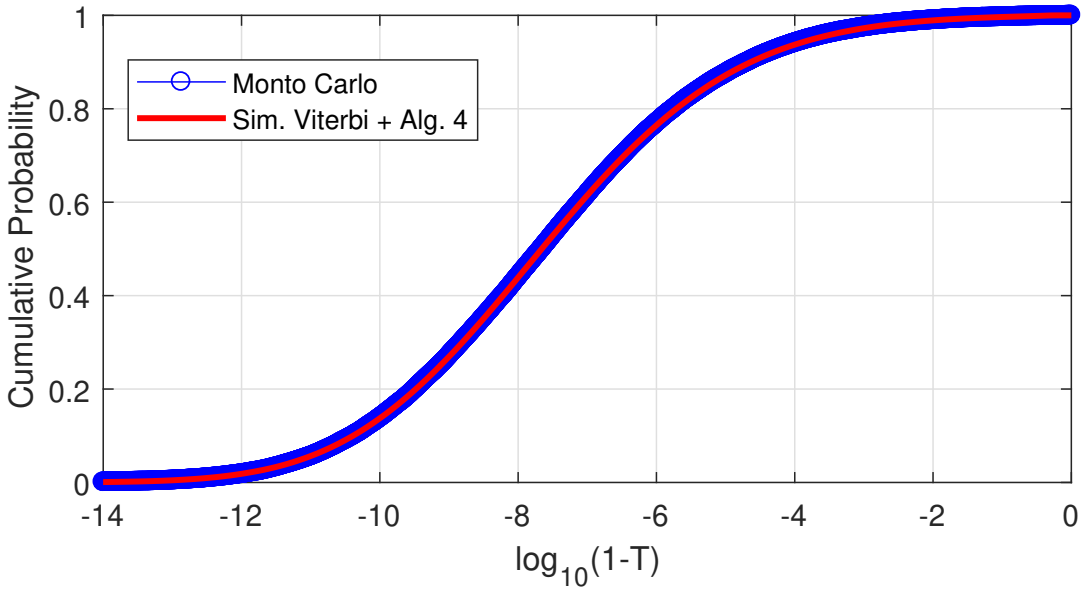

Figure 5.1: Cumulative histogram of ROVA metric computed by simulation of Viterbi/Algorithm 4 and by Monte Carlo of (5.9) with U truncated to 21 for 64-state, rate-1/3 convolutional code with $n = 32$ at SNR 1.0 dB.

This figure was previously presented in part in an IEEE TCOM journal paper [BBK22].

For the following analysis, $P(\mathrm{C})$ is the probability of Viterbi selecting the correct codeword a.k.a. the throughput, $P(\mathrm{E})$ is the probability of Viterbi selecting an incorrect codeword, i.e., UER, and $P(\mathrm{NACK})$ is probability of negative acknowledgement, i.e. rejecting the selected codeword because $P_0^{n_c} < T$, where T is the ROVA threshold.

The expression of (5.9) indicates a probability distribution f_P on the computed probability of correct decoding $P_0^{n_c}$. The corresponding computed probability of incorrect decoding is $1 - P_0^{n_c}$. Thus, with $P_0^{n_c} > T$ required to accept the Viterbi decoding result, we have the following expressions:

$$P(\mathrm{C}) = \int_{p=T}^{1} p f_P(p) dp \tag{5.10}$$

$$P(\mathrm{E}) = \int_{p=T}^{1} (1-p) f_P(p) dp \tag{5.11}$$

$$P(\mathrm{NACK}) = \int_{p=0}^{T} f_P(p) dp\,. \tag{5.12}$$

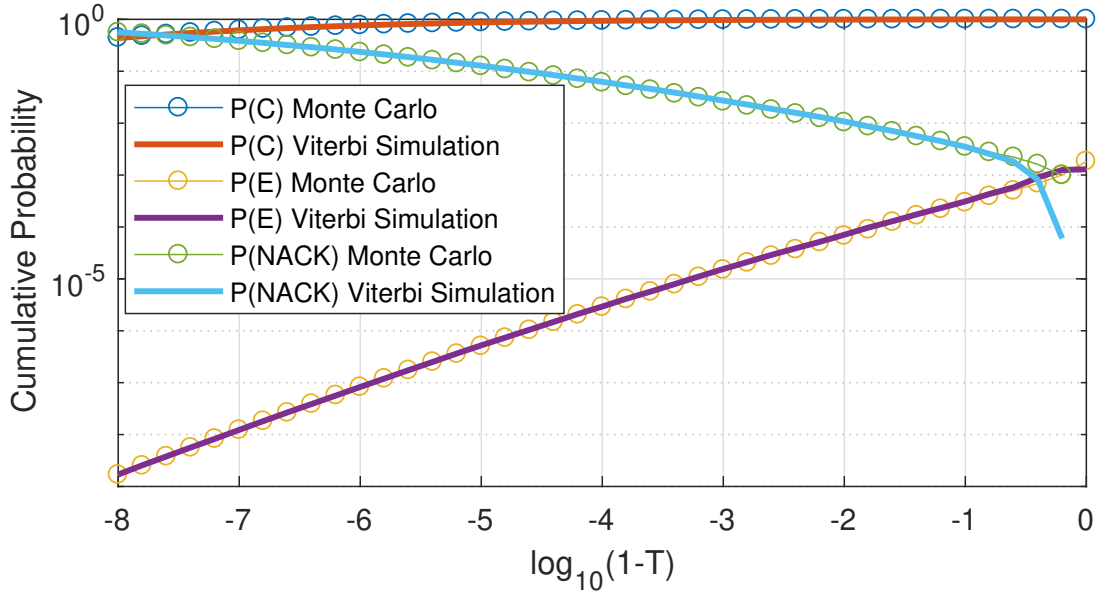

Figure 5.2: Comparison of throughput $P(\text{C})$, $P(\text{E})$, and $P(\text{NACK})$ between ROVA probabilities obtained by simulation for the code and channel of Fig. 5.1 and Monte Carlo using (5.9) with $U = 21$.

This figure was previously presented in part in an IEEE TCOM journal paper [BBK22].

Fig. 5.2 compares the application of (5.10), (5.11), and (5.12) using the cumulative histogram of $P_0^{n_c}$ generated by Monte Carlo using $U = 21$ (shown in Fig. 5.1) to the values of $P(\text{C})$, $P(\text{E})$, and $P(\text{NACK})$ obtained by simulation of Viterbi decoding with $P_0^{n_c}$ computed as described in Algorithm 4 and then applying the threshold T to decide if the codeword selected by Viterbi should be accepted.

The Monte Carlo prediction is very close to the simulated values except for $P(\text{NACK})$ for values of $\log(1-T)$ above -0.5. As shown in Fig. 7.1 (for a different code), when $\log(1-T)$ is sufficiently large, the fraction of incorrectly decoded codewords increases significantly causing the approximation of $f_{Y|X}(y^{n_c}|\hat{x}^{n_c})$ with F_0 to be inaccurate.

CHAPTER 6

Information Density as a Stopping Rule and Polyanskiy's Achievable Rate

Polyanskiy *et al.* in [PPV11] derives an achievability bound for an (l, M, ϵ) VLSF code on a DMC. (Theorem 3, [PPV11]) Fix a scalar $\gamma > 0$ Let X and $\overline{X}$ be independent copies of the same process. Let Y be the output of the DMC when X is the input. Define a sequence of information density functions

$$i(a^n; b^n) = \log \frac{P_{Y^n|X^n}(b^n|a^n)}{P_{Y^n}(b^n)} \tag{6.1}$$

and a pair of hitting times

$$\tau = \inf\{n \geq 0 : i(X^n; Y^n) \geq \gamma\} \tag{6.2}$$

$$\overline{\tau} = \inf\{n \geq 0 : i(X^n; Y^n) \geq \gamma\} \tag{6.3}$$

Then, for an integer message size $M \geq 2$, there exists an (l, M, ϵ) VLSF code with

$$l \leq E[\tau] \tag{6.4}$$

$$\epsilon \leq (M-1)\mathbb{P}[\overline{\tau} \leq \tau] \tag{6.5}$$

Although computing $\mathbb{E}[\tau]$ and $\mathbb{P}[\overline{\tau} \leq \tau]$ precisely is difficult, Polyanksiy *et al.* proved the following upper bounds when drawing i.i.d. X^n from a capacity achieving input distribution P_X

$$\mathbb{E}[\tau] \leq \frac{\gamma + a_0}{C} \tag{6.6}$$

$$\mathbb{P}[\overline{\tau} \leq \tau] \leq 2^{-\gamma} \tag{6.7}$$

where $a_0 = \sup_{x \in \mathcal{X}, y \in \mathcal{Y}} i(x;y) < \infty$. By setting $\gamma = \log \frac{M-1}{\epsilon'}$ in (6.4) and (6.5) for a given target probability of error $\epsilon' \in (0,1)$, (6.2) and (6.3) become

$$l \leq \frac{\log \frac{M-1}{\epsilon'} + a_0}{C} \tag{6.8}$$

$$\epsilon \leq (M-1)2^{-\log \frac{M-1}{\epsilon'}} = \frac{(M-1)\epsilon'}{M-1} = \epsilon' \tag{6.9}$$

This thesis uses the (6.8) and (6.9) to compute Polyankskiy's VLSF achievability bound on the rate for an AWGN channel with 8PSK modulated symbols in Chapter 10.

CHAPTER 7

Comparison to Frick and Hoeher and AID

[1]

7.1 Fricke and Hoeher Approximation

Fricke and Hoeher [FH07] developed an approximation of ROVA that reduces complexity. The computations utilize the path metric $\Gamma_m^j = \gamma_1\gamma_2\ldots\gamma_m$ which is the product of the path metrics $\gamma = f\big(y|x(i,j)\big)$ that have been selected for the survivor path that concludes at state $s_m = j$. The Fricke and Hoeher approximation computes, for each surviving branch, the probability $\hat{P}_m(i^*, j^*)$ that the survivor branch to j^* was correctly selected, assuming that the state i^* is the correct state (corresponding to the transmitted codeword) and that the survivor path to i^* was correctly selected. Fricke and Hoeher's algorithm is presented as *Algorithm 2* below.

Table .2 describes the complexity of the Fricke & Hoeher approximation for one stage assuming all branches in the trellis are active.

To understand the inaccuracy introduced by the approximation proposed in [FH07] as compared to the original ROVA proposed in [RB98], consider the simple example shown in Fig. 3.1. Applying *Algorithm 1*, produces the result

$$P_0^4 = \frac{\gamma_1(0,0)\gamma_2(0,0)\gamma_3(0,0)\gamma_4(0,0)}{\sum_{\mathcal{T}}\gamma_1\gamma_2\gamma_3\gamma_4}\,, \tag{7.4}$$

[1]This chapter was previously presented in part in an IEEE TCOM journal paper [BBK22].

Algorithm 2: An algorithmic adaptation of the Approximate ROVA Algorithm from [FH07].

Initialization: Let $\mathcal{T}_m$ be the set of valid trellis branches as defined in Alg. 1. Initialize $m = 0$, and $\Gamma_0^0 = 1$.

Iterations:

1. $m = m + 1$

2. For symbol m compute the branch metric

$$\gamma_m(i,j) = f\big(y_m | x_m(i,j)\big) \tag{7.1}$$

   for each valid branch (i,j) in $\mathcal{T}_m$, as in *Algorithm 1.*

3. For each $j \in \mathcal{S}$ with branches $(i,j) \in \mathcal{T}_m$ where Viterbi has identified branch (i^*,j) to be the survivor branch to j compute

$$\Gamma_m^j = \Gamma_{m-1}^{i^*} \gamma_m(i^*,j)\,, \tag{7.2}$$

$$\hat{P}_m(i^*,j) = \frac{\Gamma_m^j}{\sum_{(i,j)\in\mathcal{T}_m} \Gamma_{m-1}^i \gamma_m(i,j)}\,. \tag{7.3}$$

4. if $m = n_c$ conclude by reporting the probability of codeword error as $1 - \prod_{m=1}^{n_c} \hat{P}_m(i^*,j)$ for the branches (i^*,j) of the winning path selected by Viterbi, otherwise, go to step 1.

where for this trellis,

$$\begin{aligned}\sum_{\mathcal{T}} \gamma_1\gamma_2\gamma_3\gamma_4 =& \gamma_1(0,0)\gamma_2(0,0)\gamma_3(0,0)\gamma_4(0,0) \\ &+ \gamma_1(0,1)\gamma_2(1,2)\gamma_3(2,0)\gamma_4(0,0) \\ &+ \gamma_1(0,1)\gamma_2(1,3)\gamma_3(3,2)\gamma_4(2,0) \\ &+ \gamma_1(0,0)\gamma_2(0,1)\gamma_3(1,2)\gamma_4(2,0)\end{aligned}$$

Equation (7.4) is precisely the probability given in (4.2) that the codeword has been correctly decoded.

In contrast, applying *Algorithm 2*, produces the result

$$\prod_{m=1}^{n_c} \hat{P}_m(i^*, j^*) = \frac{\gamma_1(0,0)\gamma_2(0,0)\gamma_3(0,0)\gamma_4(0,0)}{D}\,, \tag{7.5}$$

$$\begin{aligned}\text{where } D = \quad & \gamma_1(0,0)\gamma_2(0,0)\gamma_3(0,0)\gamma_4(0,0) \\ &+ \gamma_1(0,1)\gamma_2(1,2)\gamma_3(2,0)\gamma_4(0,0) \\ &+ \gamma_1(0,1)\gamma_2(1,3)\gamma_3(3,2)\gamma_4(2,0) \\ &+ \gamma_1(0,1)\gamma_2(1,3)\gamma_3(3,2)\gamma_4(2,0)\alpha\end{aligned}$$

In this example the $\gamma_1(0,0)\gamma_2(0,1)\gamma_3(1,2)\gamma_4(2,0)$ term of $\sum_{\mathcal{T}} \gamma_1\gamma_2\gamma_3\gamma_4$ computed by ROVA is replaced by a different term that scales the "available" alternative path $\gamma_1(0,1)\gamma_2(1,3)\gamma_3(3,2)\gamma_4(2,0$ by

$$\alpha = \frac{\gamma_1(0,1)\gamma_2(1,2)\gamma_3(2,0)}{\gamma_1(0,0)\gamma_2(0,0)\gamma_3(0,0)}\,. \tag{7.6}$$

As this example illustrates, the limitation of the Fricke-Hoeher approximation is that it does not have access to certain paths in $\mathcal{T}$ that were pruned away by states that are not on the final winning path. The denominator in (7.3) is a sum of the path metrics of the surviving paths at stage $m-1$ that enter state j at stage m. Thus, the probability of codeword error $1-\prod_{m=1}^{n_c} \hat{P}_m(i^*, j)$ will only have access to the winning path and $(N_b-1)(n_c-v)$ terminated paths out of the total $(n_c-v)^{N_b}$ paths. To the extent that the pruned paths have low probability, the Fricke-Hoeher approximation can be accurate.

7.2 Accumulated Information Density

Polyanskiy et al. [PPV11] used a threshold on information density at the receiver to decide when to terminate random codes. This termination approach provides bounds on achievable throughput for codes with finite blocklength. The information density of a received symbol y_i with respect to a selected codeword symbol $\hat{x}_i$ is computed as

$$i(y_j, \hat{x}_j) = \log_2 \frac{f_{Y|X}(y_j|\hat{x}_j)}{f_Y(y_j)} \,. \tag{7.7}$$

In (7.7), $f_Y(y_j)$ is computed assuming that each possible symbol $x \in \mathcal{X}$ is drawn i.i.d. according to an input distribution, either a probability density function (PDF) $f_X(x)$ or a probability mass function (PMF) $P_X(x)$. For practical communication systems in which a convolutional code is used in conjunction with a constellation of possible transmitted symbols, the input alphabet $\mathcal{X}$ is finite and is exactly the constellation. For a typical encoder (without probabilistic shaping [WDY21]), each constellation point is equally likely so that $P_X(x) = |\mathcal{X}|^{-1}$.

Following the termination approach of [PPV11], accumulated information density (AID) sums (7.7) for each symbol in the codeword to produce $i_{\text{AID}}(y^{n_c}, \hat{x}^{n_c})$ as follows:

$$\begin{aligned} i_{\text{AID}}(y^{n_c}, \hat{x}^{n_c}) &= \sum_{j=1}^{n_c} i(y_j, \hat{x}_j) && (7.8)\\ &= \sum_{j=1}^{n_c} \log_2 \left(\frac{f_{Y|X}(y_j|\hat{x}_j)}{f_Y(y_j)} \right) && (7.9)\\ &= \log_2 \left(\frac{\prod_{j=1}^{n_c} f_{Y|X}(y_j|\hat{x}_j)}{\prod_{j=1}^{n_c} f_Y(y_j)} \right) && (7.10)\\ &= \log_2 \left(\frac{f_{Y|X}(y^{n_c}|\hat{x}^{n_c})}{\sum_{x^{n_c} \in \mathcal{X}^{n_c}} |\mathcal{X}|^{-n} f_{Y|X}(y^{n_c}|x^{n_c})} \right) && (7.11) \end{aligned}$$

where $\mathcal{X}^{n_c}$ is the set of all sequences of n_c symbols. For AID, the denominator in (7.11) includes every possible sequence of n_c symbols from the alphabet (constellation) $\mathcal{X}$. However, only sequences that are actually codewords could have been transmitted. Including all possible sequences allows the computation of AID to be symbol-wise and thus much simpler than ROVA, but it introduces an inaccuracy.

Algorithm 3: Computation of i_{AID}.

Initialization: Let $\mathcal{T}_m$ be the set of valid trellis branches as defined in Alg. 1. Initialize $m = 0$, $\Gamma_0^0 = 1$, $\Pi(0) = 1$.

Iterations:

1. $m = m + 1$

2. Compute branch metrics $\gamma_m(i, j)$ as in Alg. 1.

3. For each $j \in \mathcal{S}$ with branches $(i, j) \in \mathcal{T}_m$ where Viterbi has identified survivor branch (i^*, j) compute

$$\Gamma_m^j = \Gamma_{m-1}^{i^*} \gamma_m(i^*, j)\,. \tag{7.12}$$

4. Compute $f_Y(y_m) = \sum_{x \in \mathcal{X}} |\mathcal{X}|^{-1} f(y_m|x)$ and

$$\Pi(m) = \Pi(m-1) f_Y(y_m) \tag{7.13}$$

5. if $m = n_c$ conclude by reporting

$$i_{\mathrm{AID}}(y^{n_c}, \hat{x}^{n_c}) = \log_2 \frac{\Gamma_n^0}{\Pi(n)}\,, \tag{7.14}$$

   otherwise, go to step 1.

Algorithm 3, above, provides a procedure for computing i_{AID}. Let N_s be the number of states $|\mathcal{S}|$ and N_b be the number of branches entering each state. For the common scenario where $N_b = 2$, Algorithm 1 (ROVA) requires about $6N_s$ multiplications per trellis stage, but Algorithm 3 (AID) requires only about N_s multiplications.

Figs. 7.1 and 7.2 compare the efficacy of ROVA and AID by plotting the probability density function of metric values for correctly and incorrectly decoded sequences. For AID, the sequences are organized by the AID metric, which is the accumulated information density. For ROVA, they are organized by the ROVA metric of word-error probability. In Figs. 7.1 and 7.2, a smaller overlap between the density functions for correctly and incorrectly

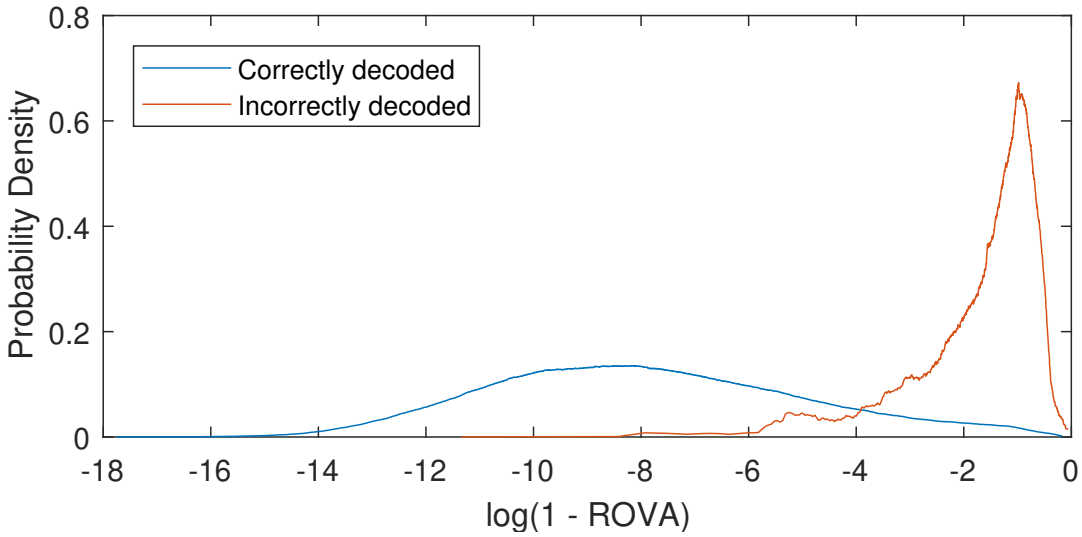

Figure 7.1: Probability density of ROVA error values for correct and incorrect decodings for the same conditions as in Fig. 4.1. The minimal overlap between the incorrect and correct decodings indicates that setting a decision threshold using ROVA is an effective way to reduce undetected errors.

This figure was previously presented in part in an IEEE TCOM journal paper [BBK22].

decoded sequences indicates a better ability for the metric to indicate when a sequence should be deemed unreliable. The better separation (smaller overlap area) seen in Fig. 7.1 as compared to Fig. 7.2 shows how ROVA is more effective than AID in this example.

Fig. 7.3 shows the UER versus throughput for ROVA, the ROVA approximation by Frick and Hoeher, and AID, where the throughput is a function of the threshold that determines whether to accept the Viterbi result as reliable. Throughput is defined as the ratio of correctly decoded sequences that passed the threshold to the total number of received sequences. Fig. 7.3 confirms the relatively poor performance of AID that was suggested in Figs. 7.1 and 7.2 . For a given target UER, AID supports a much lower throughput than ROVA. Despite its lower complexity, AID turns out to be too inaccurate to use as a decoder reliability metric in practice. The ROVA approximation performs similarly to ROVA, suggesting that either of these could be used as a decoder reliability metric.

To generate Figs. 7.1 and 7.2, it was necessary to simulate a sufficient number of incorrectly decoded sequences in the tail of the distribution, which posed a challenge as these events are much less likely to occur. In order to reduce overall simulation time, importance

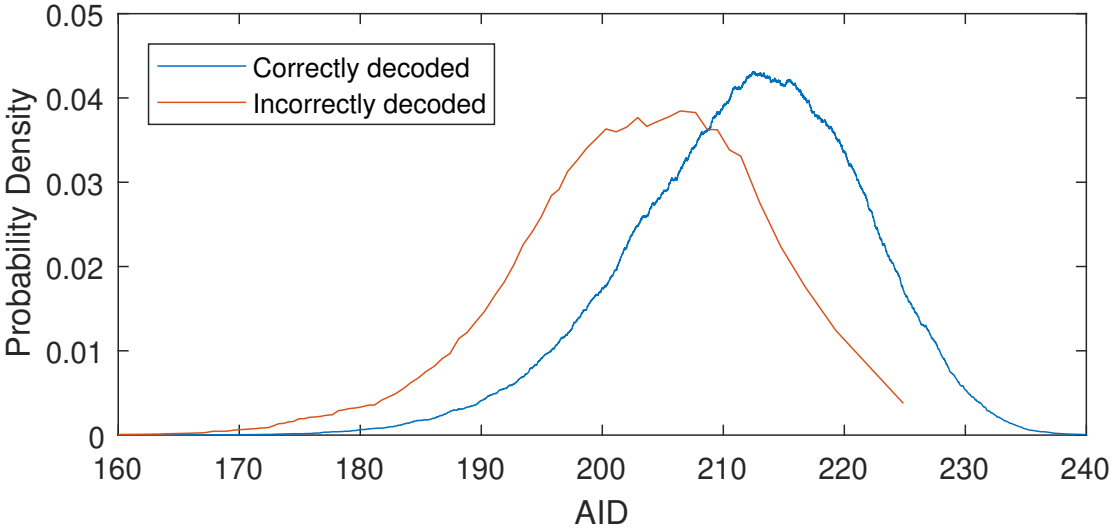

Figure 7.2: Probability density of AID values for correct and incorrect decodings for the same scenario as Fig. 4.1. There is considerable overlap between the incorrect and correct decodings, which suggests that using AID is an ineffective way to reduce undetected errors.

This figure was previously presented in part in an IEEE TCOM journal paper [BBK22].

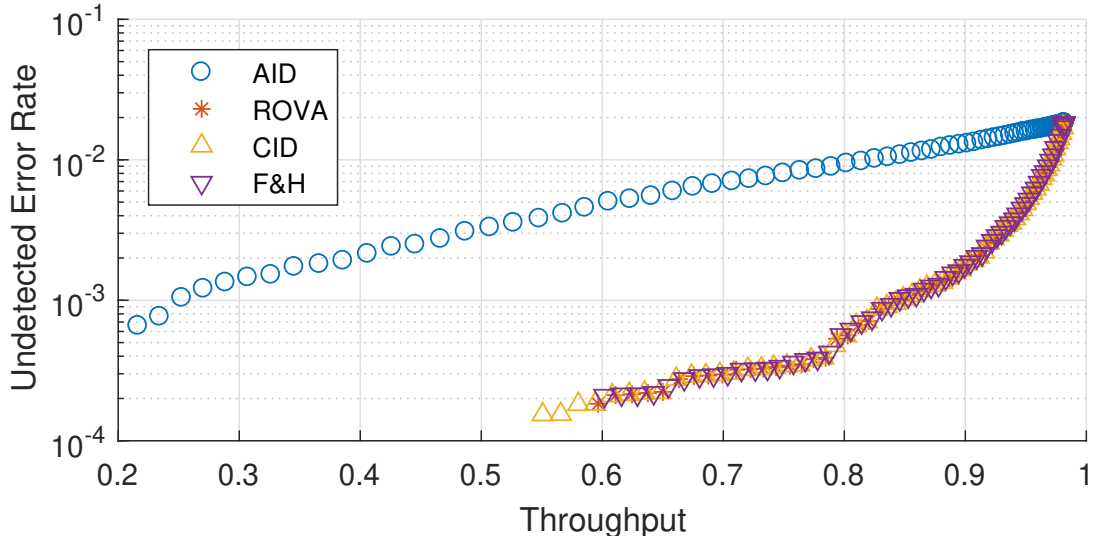

Figure 7.3: Undetected error rate (UER) as a function of throughput for ROVA, approximate ROVA, AID, and CID showing the operating points of (throughput, UER) achievable with thresholds on ROVA, approximate ROVA AID, and CID metrics for the same scenario as Fig. 4.1. ROVA and CID give identical performance as expected by (8.5). Approximate ROVA has near identical performance to ROVA and CID.

This figure was previously presented in part in an IEEE TCOM journal paper [BBK22].

sampling was utilized to increase the likelihood of the decoder choosing incorrect codewords. The probability of incorrect decoding increases with the Euclidean norm of the Gaussian noise. Noise points with a large Euclidean distance were drawn with higher probability than with the Gaussian noise distribution according to an importance-sampling bias function.

The distribution of the norm of an N-dimensional IID zero-mean Gaussian noise vector with variance σ^2 in each dimension is equivalent to the Nakagami distribution with $m = N/2$ and $\Omega = N\sigma^2$. Fig. 7.4 shows the noise norm distribution after the importance-sampling bias function. It is a mixture distribution including two equally-likely components: the above Nakagami distribution and a uniform distribution on the interval $[9, 14]$.

The combination of these two distributions provides sufficient data samples in both the body and tail of the distribution. Following the importance-sampling paradigm, each generated data sample is weighted by the ratio of the original probability distribution to the biased probability distribution. The weighted data is then used to generate a weighted cumulative distribution, which is then used to approximate a probability density curve by taking the local derivative.

The proof that the AWGN vector norm for BPSK modulation follows a Nakagami distribution is as follows: The noise vector norm $||g||$ is defined as

$$||g|| = \sqrt{(g_1)^2 + ... + (g_N)^2} = \sqrt{\sum_{k=1}^{N} (g_k)^2} \tag{7.15}$$

where each g_k is an i.i.d. normal variable with mean 0 and standard deviation σ_k.

$$g_i \sim \mathcal{N}_i(0, \sigma_i^2) \rightarrow \frac{g_i}{\sigma_i} \sim \mathcal{N}_i(0, 1) \tag{7.16}$$

$$\left(\frac{g_i}{\sigma_i}\right)^2 \sim (\mathcal{N}_i(0, 1))^2 = \mathcal{X}^2(1) \equiv \Gamma\left(\frac{1}{2}, 2\right) \tag{7.17}$$

where $\mathcal{X}^2(1)$ is a chi-squared random variable with 1 degree of freedom and $\Gamma(k, \theta)$ is the gamma distribution random variable with shape k and scale θ.

$$g_i^2 \sim \sigma_i^2 \Gamma\left(\frac{1}{2}, 2\right) = \Gamma\left(\frac{1}{2}, 2\sigma_i^2\right) \tag{7.18}$$

Each g_i has the same standard deviation σ, which leads to the following simplification.

$$\Gamma\left(\frac{1}{2}, 2\sigma_i^2\right) = \Gamma\left(\frac{1}{2}, 2\sigma^2\right) \tag{7.19}$$

The sum of independent Gamma distributions with the same rate is equivalent to a single Gamma distribution with individual shapes summed together. Therefore, $||g||$ is the square root of a gamma distribution random variable with $\frac{N}{2}$ shape and $2\sigma^2$ scale, which is a Nakagami distribution.

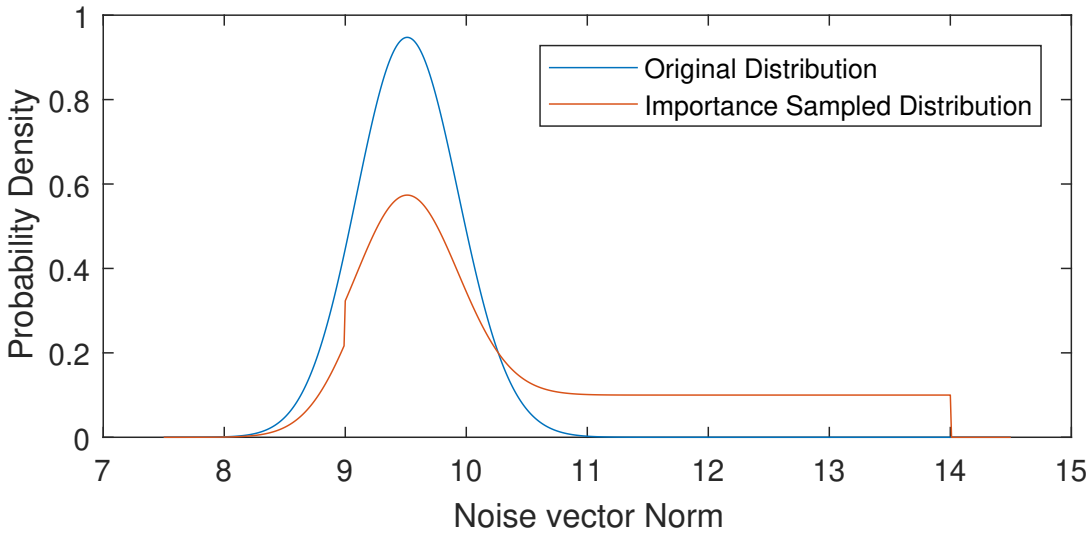

Figure 7.4: The original Nakagami distribution which describes the distribution of the noise vector norms compared to the bias distribution for the same scenario as Fig. 4.1. The bias distribution is a mixture distribution composed of one half of the original Nakagami distribution with one half of a uniform distribution on the interval $[9, 14]$.

This figure was previously presented in part in an IEEE TCOM journal paper [BBK22].

CHAPTER 8

CID

[1] As an improvement to AID, we propose a new metric, the codeword information density (CID), which similar to AID but computed for an entire codeword rather than for a single symbol. The CID metric is computed for the codeword selected by Viterbi as follows:

$$i_{\mathrm{CID}}(y^{n_c}, \hat{x}^{n_c}) = \log_2 \frac{f_{Y|X}(y^{n_c}|\hat{x}^{n_c})}{\sum_{x^{n_c}\in\mathcal{C}} P(x^{n_c}) f_{Y|X}(y^{n_c}|x^{n_c})} \tag{8.1}$$

CID operates on the complete sequences y^{n_c} and $\hat{x}^{n_c}$ and is limited to only consider valid codewords $x^{n_c} \in \mathcal{C}$. This gives a higher complexity, but higher accuracy relative to AID. Algorithm 4 below provides a procedure for computing i_{CID}.

Comparing (4.2) and (8.1), we find that ROVA and CID have almost the same formula. Starting with (4.2), including a $P(x^{n_c})$ term in the denominator and taking a logarithm produces (8.1). Consequently, CID and ROVA have a one-to-one transformation given by

$$i_{\mathrm{CID}}(y^{n_c}, \hat{x}^{n_c}) = \log_2 \left(\frac{P_0^{n_c}}{P(x^{n_c})} \right). \tag{8.5}$$

Thus, CID and ROVA turn out to be identical metrics. However, the journey from AID to CID and the recognition that CID computes exactly the same value as ROVA reveals a lower-complexity approach to computing ROVA. The original ROVA implementation computes the normalization factor Δ_m for each trellis stage, which in turn requires the additional computation of $\bar{P}_j^m$ for each surviving trellis branch. The algorithm we propose for computing CID does not require these computations.

As an example, consider the number of multiplications required for the common case of a rate-$1/n$ convolutional code where $N_b = 2$, excluding the number of multiplications necessary

[1] This chapter was previously presented in part in an IEEE TCOM journal paper [BBK22].

Algorithm 4: Computation of proposed i_{CID} (and ROVA).

Initialization: For $i, j \in \mathcal{S}$, let $\mathcal{T}_m$ be the set of valid trellis branches as defined in *Algorithm 1.* Initialize $m = 0$, $\Gamma_0^0 = 1$, $Z_0^0 = 1$.

Iterations:

1. $m = m + 1$
2. Compute branch metrics $\gamma_m(i, j)$ as in Algorithm 1.
3. Compute Γ_m^j as in Algorithm 2.
4. For each $j \in \mathcal{S}$ compute
$$Z_m^j = \sum_{(i,j)\in\mathcal{T}_m} Z_{m-1}^i \gamma_m(i, j)\,. \tag{8.2}$$
5. if $m = n_c$ conclude by reporting either
$$i_{\mathrm{CID}}(y^{n_c}, \hat{x}^{n_c}) = \log_2\left(\frac{\Gamma_n^0}{P(x^{n_c})Z_n^0}\right), \text{or} \tag{8.3}$$
$$P_0^{n_c} = \frac{\Gamma_n^0}{Z_n^0} \tag{8.4}$$
otherwise, go to Step 1.

for the branch metric $\gamma_m(i,j)$. The original ROVA algorithm requires approximately $(3 + 2N_b)N_s = 7N_s$ multiplications per trellis stage. AID requires only N_s multiplications per trellis stage, but is inaccurate. The CID-inspired ROVA computation in Algorithm 4 requires only $(1 + N_b)N_s = 3N_s$ multiplies per trellis stage and computes the identical ROVA value of $P_0^{n_c}$ as in Algorithm 1. It is noted that the total complexity savings are not directly proportional to the difference in the listed number of operations, as each algorithm must still separately compute $\gamma_m(i,j)$.

CHAPTER 9

Complexity Analysis of Reliability Metrics

[1] This section analyzes the additional complexity beyond standard Viterbi decoding required by each reliability metric in a ZTCC with N_s states, N_b branches per state, k trellis stages, and n_d transmitted dimensions per transmitted symbol, i.e. per trellis stage.

For analysis, the trellis stages are decomposed into three sections:

1. The initialization section where the number of active trellis states is increasing from one to N_s.

2. The regular transmission section where all N_s states are trellis active.

3. The termination section where the number of active trellis states is decreasing from N_s to one.

The decoder performs reliability metric computations along each branch. For each of the three sections we will compute the total number of branches on which reliability metric computations are performed.

The initialization section begins in only one state, the zero state. With each subsequent stage, the number of active states increases by a factor of N_b. Thus, the number of stages needed to activate all available trellis states is given by $\log_{N_b}(N_s)$. The number of branches in the initialization section on which reliability metric computations are performed is given by

$$\sum_{i=1}^{\log_{N_b}(N_s)} (N_b)^i = \frac{N_b^{\log_{N_b}(N_s)+1} - N_b}{N_b - 1} \tag{9.1}$$

[1] This chapter was previously presented in part in an IEEE TCOM journal paper [BBK22].

There are $k - 2\log_{N_b}(N_s)$ stages in the regular transmission section. Each stage in this section contains $N_b N_s$ branches connecting the previous states to the current states. The decoder must therefore perform $N_b N_s(k-2\log_{N_b}(N_s))$ reliability metric branch computations in this section.

The termination section begins with all N_s states active. With each subsequent stage, the number of active states decreases by a factor of N_b until only the zero state is active at the end of the transmission. Thus, the number of stages in the termination section is given by $\log_{N_b}(N_s)$. The number of branches in the termination section on which reliability metric computations are performed is the same as for the initialization section, as described in (9.1).

Thus, the total number of reliability metric computations performed throughout the decoding process is given by

$$N_b N_s(k - 2\log_{N_b}(N_s)) + \frac{2(N_b^{\log_{N_b}(N_s)+1} - N_b)}{N_b - 1} \tag{9.2}$$

Neglecting overhead computations that occur once per codeword or once per stage, the computational complexity required by a reliability metric can be estimated by multiplying (9.2) by the number of operations needed per branch for that reliability metric. The initialization and termination sections each occupy $\log_{N_b}(N_s)$ trellis stages. If k is much greater than $2\log_{N_b}(N_s)$, then the regular transmission section will dominate the overall complexity.

As shown in step 2 of each of Algs. 1-4, each reliability metric requires the computation of branch metric γ_m along every branch in the trellis. For an AWGN channel, γ_m is computed for each branch by multiplying the conditional densities for each dimension of the symbol corresponding to that branch as follows:

$$\gamma_m = \prod_{i=1}^{n_d} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \hat{x}_i)^2}{2\sigma^2}} \tag{9.3}$$

If the SNR of the channel is fixed, the terms involving σ can be treated as a constants and precomputed.

For all the algorithms considered, in addition to computing γ_m, the reliability metric algorithm requires steps that need to be performed either along every branch within the

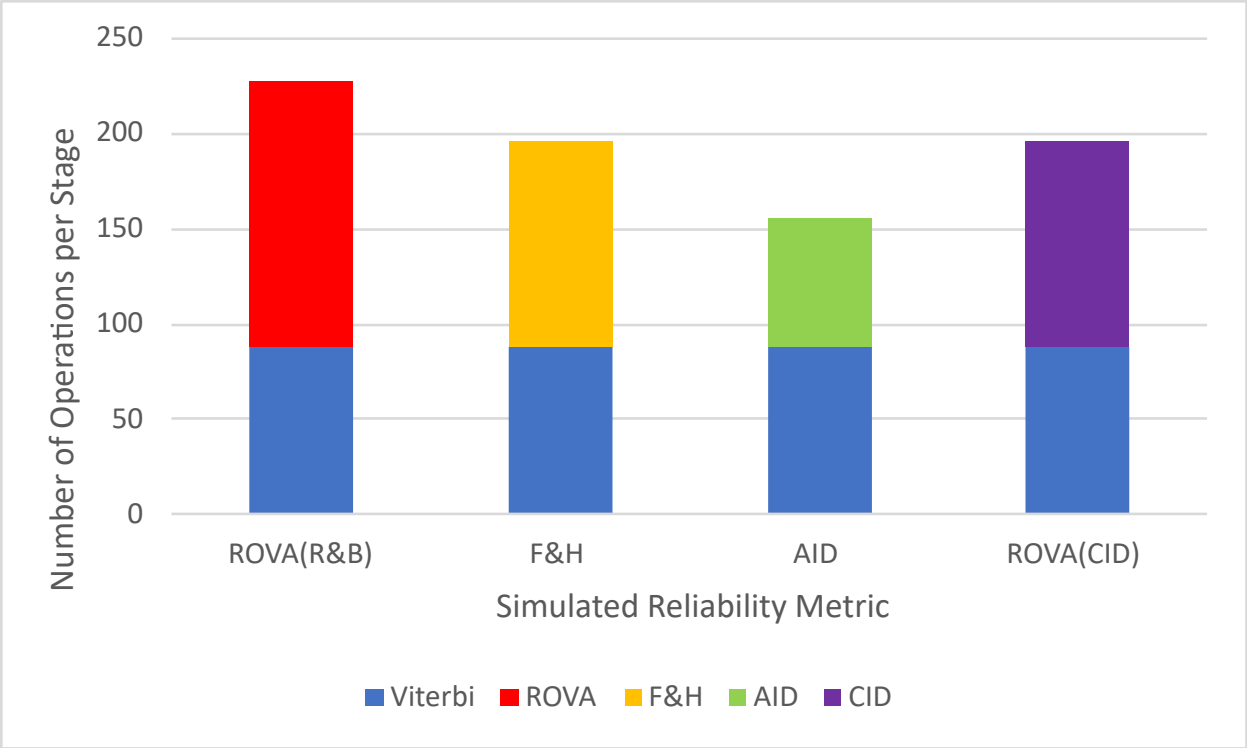

Figure 9.1: Number of operations needed per stage to implement Viterbi alongside the chosen reliability metric. Number of operations is based on a 4 state, 128 information bit, rate 1/2 decoder.

This figure was previously presented in part in an IEEE TCOM journal paper [BBK22].

trellis stage $\mathcal{T}_m$, or along the surviving branches chosen by the Viterbi algorithm. For the Raghavam and Baum implementation of ROVA [RB98], the computation of Δ_m in (4.4) must be performed once per stage using values from all valid branches, while the computation of P_j^m and $\bar{P}_j^m$ in (4.5)-(4.6) only needs to be performed along the surviving branches. The Frick and Hoeher Approximation of ROVA [FH07] requires the computation of Γ_m^j in (7.2) along every valid branch as those values are necessary for the computation of the denominator in (7.3) for the winning branches. The AID algorithm only requires the computation of Γ_m^j in (7.12) on the surviving branches chosen by Viterbi, and $\Pi(m)$ in (7.13) can be computed once for every stage. The CID implementation of ROVA requires the computation of Z_m^j in (8.2) for each state and Γ_m^j for each surviving path.

Table 9.1 shows the number of operations per stage necessary to compute each reliability metric using the most efficient implementation we could devise. Fig. 9.1 illustrates how the additional computation required for each reliability metric compares with the computations

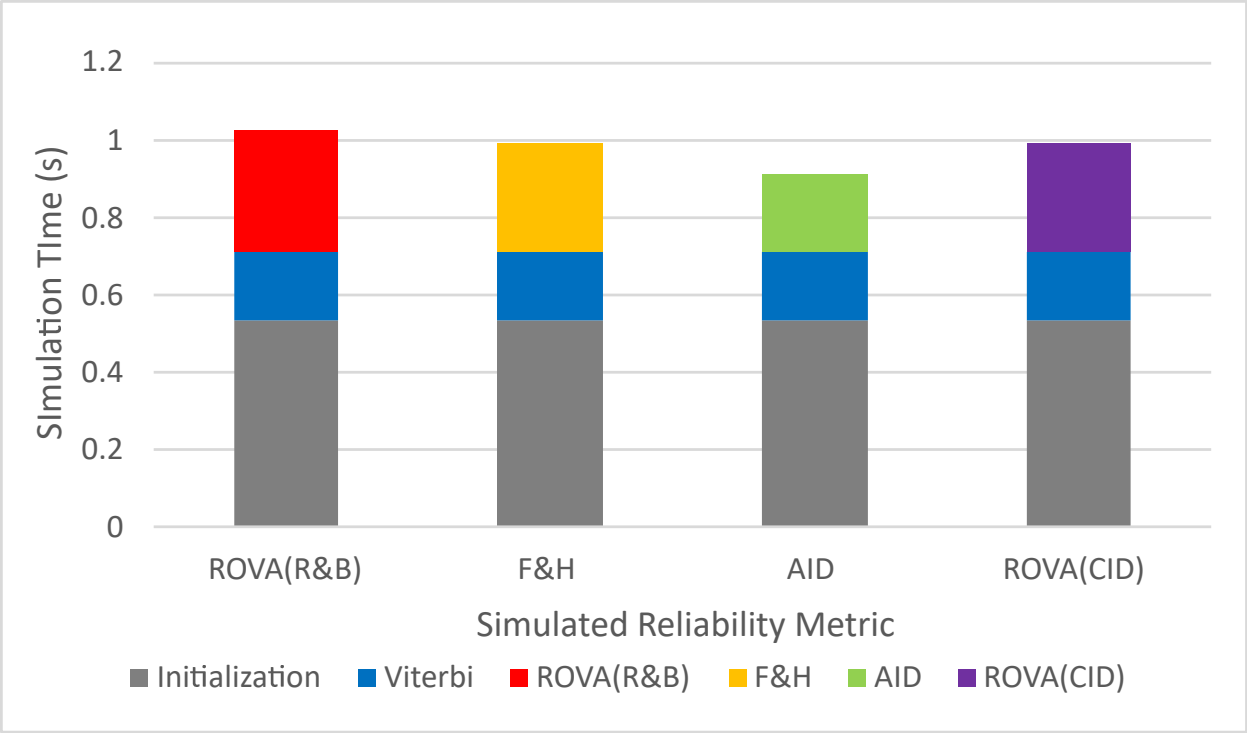

Figure 9.2: Simulation time per 100 decodings with Viterbi and the chosen reliability metric. Initialization includes everything that was simulated that was not a part of the Viterbi algorithm or the chosen reliability metric. Initialization includes tasks such as creation of the trellis structure, encoding the input sequence, and computing every state transition bit sequence in the trellis.

This figure was previously presented in part in an IEEE TCOM journal paper [BBK22].

required for Viterbi decoding for the example of $\alpha = 2$, $N_s = 4$, $N_b = 2$, and $n_d = 2$. Fig. 9.2 shows the simulation time results for this same example.

While Viterbi requires only squared Euclidean distance as a metric, the various reliability metrics all require the computation of (9.3) over all branches. This is the most expensive step in each metric evaluation. A straightforward approach requires $7n_d$ operations per branch. However, pre-computing each multiplicand in (9.3) once per stage can significantly reduce the computation. For each unique symbol value in each dimension of the mapped symbol alphabet, the term

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \hat{x}_i)^2}{2\sigma^2}} \tag{9.4}$$

can be computed for a received symbol $\hat{x}$. This requires $6n_d$ calculations α times, where α represents the number of unique values of the mapped symbol alphabet in each dimension.

Thus the overall reduction is from $7n_dN_sN_b$ computations per stage to $n_dN_sN_b + 6n_d\alpha$ computations per stage.

For example, a code that uses BPSK modulation would have an α of 2, while 8PSK modulation would have an α of 5, corresponding to the five possible values per dimension $\{-1, -1/\sqrt{2}, 0, 1/\sqrt{2}, 1\}$. Thus, (9.4) can be computed once for each of these five values, and the results can be cached and accessed when they are multiplied together in (9.3). Each metric computation now requires $n_d + 1$ steps per branch to multiply the cached values together and incorporate the result into Γ_m^j.

The implementation of each metric contains an additional $3n_d$ overhead operations per branch and $2n_d$ operations per stage. ROVA as in [RB98] has an additional $3N_bN_s$ and $4N_s$ term necessary to compute (4.4)-(4.6) of Algorithm 1. The ROVA approximation [FH07] and ROVA computed as CID have an additional N_bN_s term necessary to compute (7.3) and (8.2) respectively. After these optimizations, AID [PPV11] is the fastest metric, ROVA computed as CID and the ROVA approximation share the same complexity, and ROVA computed as in [RB98] has the highest complexity. Although the ROVA approximation is very accurate in our simulation, ROVA computed as CID offers the exact reliability at no additional complexity as compared to the approximation.

Table 9.1: Operations per trellis stage (OPTS) for the Viterbi algorithm and the additional complexity beyond Viterbi for each of the four considered reliability metrics, when all trellis states are active. The Example OPTS value is the value of OPTS for the example where $\alpha = 2$, $N_s = 4$, $N_b = 2$, and $n_d = 2$. Additional Time is the average additional time needed to simulate the chosen reliability metric in addition to the Viterbi algorithm per 100 decoding simulations. Simulations are performed with a 4 state rate-1/2 BPSK modulated decoder where $\alpha = 2$, $N_s = 4$, $N_b = 2$, and $n_d = 2$ on an Intel i7-4720HQ processor.

Algorithm	Operations per trellis stage	Example OPTS value	Add. Time (s)
Viterbi	$N_s N_b(5n_d + 1)$	88	0
ROVA (R&B)	$N_b N_s(4n_d + 4)$ $+4N_s + 2n_d + 6n_d\alpha$	140	0.317
F&H	$N_s N_b(4n_d + 2)$ $+2n_d + 6n_d\alpha$	108	0.280
AID	$N_s(4n_d + 1)$ $+4n_d + 6n_d\alpha$	68	0.203
ROVA (CID)	$N_s N_b(4n_d + 2)$ $+2n_d + 6n_d\alpha$	108	0.281

CHAPTER 10

Example of incremental redundancy based on ROVA and Comparison

[1]

10.1 Approximately Optimal Ordering of Symbol Transmission for Incremental Retransmission

The techniques described in this section will optimize the order of additionally transmitted symbols when a retransmission request occurs in an incremental retransmission scheme. The results of this section will be applied to Sec. 10.2.

Let $\boldsymbol{\varepsilon}_{x,x'}$ denote the event of decoding codeword x' when x is sent, and let $\mathbf{e}_{x,x'}$ denote the event that codeword x' is more likely than codeword x given the received y. We have the union bound

$$\text{FER} = P\left(\bigcup_{\substack{x,x'\in\mathcal{C}\\ x\neq x'}} \boldsymbol{\varepsilon}_{x,x'}\right) = P\left(\bigcup_{\substack{x,x'\in\mathcal{C}\\ x\neq x'}} \mathbf{e}_{x,x'}\right) \tag{10.1}$$

$$\leq \sum_{\substack{x,x'\in\mathcal{C}\\ x\neq x'}} P\left(\mathbf{e}_{x,x'}\right). \tag{10.2}$$

[1] This chapter was previously presented in part in an IEEE TCOM journal paper [BBK22].

We denote $P\left(\mathbf{e}_{x,x'}\right)$ as the pairwise error probability. We have the Q-function approximation:

$$P\left(\mathbf{e}_{x,x'}\right) = Q\left(\sqrt{\frac{d^2\left(x,x'\right)}{2N_0}}\right) p(x) \tag{10.3}$$

$$\leq Q\left(\sqrt{\frac{d_{free}^2}{2N_0}}\right) e^{\frac{d_{\text{free}}^2}{4N_0}} e^{-\frac{d^2(x,x')}{4N_0}} p(x)\,, \tag{10.4}$$

where $d^2(x,x')$ denotes the squared Euclidean distance between codewords x and x', and

$$p(x) = \prod_{\ell=1}^{n^c} q(x_\ell) \tag{10.5}$$

is the probability that the codeword x is sent, where $q(\cdot)$ is input distribution chosen for the channel.

Let

$$W = e^{-\frac{1}{4N_0}}\,. \tag{10.6}$$

Using the additivity of squared Euclidean distance over components, we have

$$\sum_{\substack{x,x'\in\mathcal{C}\\ x\neq x'}} \exp\left(-\frac{d^2\left(x,x'\right)}{4N_0}\right) p(x) \tag{10.7}$$

$$= \sum_{\substack{x,x'\in\mathcal{C}\\ x\neq x'}} \exp\left(-\frac{1}{4N_0}\sum_{\ell=1}^{n^c} d^2\left(x_\ell,x'_\ell\right)\right) p(x) \tag{10.8}$$

$$= \sum_{\substack{x,x'\in\mathcal{C}\\ x\neq x'}} \prod_{\ell=1}^{n^c} \left[W^{d^2\left(x_\ell,x'_\ell\right)} q\left(x_\ell\right)\right] \tag{10.9}$$

$$= \sum_{x,x'\in\mathcal{C}} \prod_{\ell=1}^{n^c} \left[W^{d^2\left(x_\ell,x'_\ell\right)} q\left(x_\ell\right)\right] - \sum_{x} \prod_{\ell=1}^{n^c} \left[W^0 q\left(x_\ell\right)\right] \tag{10.10}$$

$$= \sum_{x,x'\in\mathcal{C}} \prod_{\ell=1}^{n^c} \left[W^{d^2\left(x_\ell,x'_\ell\right)} q\left(x_\ell\right)\right] - 1\,, \tag{10.11}$$

We convert the sum over codewords into a sum over paths through the trellis. In order to compute (10.11), it is necessary to compute $W^{d^2(x_\ell,x'_\ell)}$ for each branch in each codeword. All paths that start and end at the zero state are valid for a zero-terminated convolutional

code. Since the decoded codeword is always zero-terminated, the starting and ending states are always correct.

Previous work in [Wes04] describes a symmetry-based technique for reducing the size of state-diagrams necessary to describe trellis codes with standard constellations and labeling. The minimal state transition diagram as a function of W takes the form

$$\begin{bmatrix} \mathbf{d} & \mathbf{c} \\ \mathbf{b} & \mathbf{A} \end{bmatrix} \tag{10.12}$$

where $\mathbf{A}$ specifies the transition labels from errored states to errored states, $\mathbf{b}$ specifies the transitions from correct states to errored states, $\mathbf{c}$ specifies the transitions from errored states to correct states, and $\mathbf{d}$ specifies the transitions from correct states to correct states. Each element in the minimal state transition diagram is a linear combination of the $W^{d^2(x_\ell, x'_\ell)}$ terms for each equivalence class associated with that state transition as defined by [Wes04]. We can rewrite (10.11) as the following transfer function

$$T(W) = \begin{bmatrix} \mathbf{d} & \mathbf{c} \end{bmatrix} \begin{bmatrix} \mathbf{d} & \mathbf{c} \\ \mathbf{b} & \mathbf{A} \end{bmatrix}^{n^c - 2} \begin{bmatrix} \mathbf{d} \\ \mathbf{b} \end{bmatrix} - 1 \,, \tag{10.13}$$

Noting that the Hamming weight does not appear in (10.11), we can combine (10.4) and (10.13) as the bound

$$\text{FER} \leq Q\left(\sqrt{\frac{d_{free}^2}{2N_0}}\right) e^{\frac{d_{free}^2}{4N_0}} T\left(W = e^{-\frac{1}{4N_0}}\right) \,, \tag{10.14}$$

where d_{free} is the free distance of the code. For punctured trellis codes, [WL98] states that the transfer function in (10.14) should not be evaluated with a single W. Each matrix in (10.13) must be evaluated separately with $W = e^{-\frac{1}{4N_0}}$ for transmitted symbols and $W = 1$ for punctured symbols. For an 8-PSK zero-terminated trellis code with constraint length v and puncturing, (10.14) becomes the following:

$$\text{FER} \leq Q\left(\sqrt{\frac{r^2}{2N_0}}\right) e^{\frac{r^2}{4N_0}} T\left(W^\ell\right) \,, \tag{10.15}$$

where r is the residual Euclidean distance of the punctured code and W^{ℓ} is the set of all $W_i = \exp -\frac{a_i}{4N_0}$ from i = 1 to ℓ where a_i is 1 if the i-th symbol from the end of the transmission is transmitted and 0 if it is punctured. The transfer function with puncturing is given by

$$T(W^{\ell}) = \begin{bmatrix} \mathbf{d} & \mathbf{c} \end{bmatrix} \Big|_{W_1} \prod_{j=2}^{n^c-1} \left(\begin{bmatrix} \mathbf{d} & \mathbf{c} \\ \mathbf{b} & \mathbf{A} \end{bmatrix} \Big|_{W_j} \right) \begin{bmatrix} \mathbf{d} \\ \mathbf{b} \end{bmatrix} \Big|_{W_\ell} - 1 \tag{10.16}$$

A greedy search algorithm detailed in Algorithm 5 is performed in Section 10.2 to select the order of additionally transmitted symbols. Greedy algorithms are in general not globally optimal, but are approximations of the globally optimal solution. The algorithm uses (10.15) to lower bound the FER for every valid additional transmitted symbol when a retransmission request occurs. The additional symbol that results in the lowest FER is the locally optimal symbol to transmit for that request given the previous selections. This process starts with the most aggressive puncturing pattern and repeats until all symbols have been transmitted.

10.2 Numerical Results for Higher Order Modulation for Variable Length Coding with a Comparison to a CRC-based Approach

Previous work by Williamson [WCW15a] has demonstrated the performance of reliability-based retransmission schemes over the AWGN channel using ROVA as the reliability metric for BPSK modulation. This section demonstrates using ROVA as a reliability metric in an incremental transmission scheme for 8-PSK modulation and compares it to using a CRC as the reliability metric.

Fig. 10.1 shows the performance of both a ROVA-based and a CRC-based incremental transmission scheme over the AWGN channel at an SNR of 6 dB and target error rate $\epsilon = 10^{-3}$ using 8-PSK modulation and convolutional code {173,46,133} as described in [WLS00] with soft-decision decoding and feedback of the selected metric after every symbol to determine when to terminate the transmission. Every CRC polynomial was chosen from [KC04b] such that each data point in Fig. 10.1 achieves a frame error rate less than target ϵ. Fig. 10.2 shows the observed frame error rates for each of the simulated data points in Fig. 10.1. It can be seen that ROVA is able to target specific error rates much more

Algorithm 5: Greedy search algorithm to determine the order of additionally transmitted symbols.

Initialization: Compute the minimal state transition diagram as in eq. (10.12). Select the most aggressive puncturing pattern to be considered as the initial puncturing pattern. Initialize m to be equal the number of punctured symbols in the initial pattern.

Iterations:

1. $m = m - 1$
2. Select a punctured symbol in the puncturing pattern.
3. Compute (10.15) for the current puncturing pattern if the selected symbol were additionally transmitted. Store the result.
4. Return to step 2 until all punctured symbols have been evaluated.
5. Update the puncturing pattern so that the additional symbol that results in the lowest FER bound is transmitted.
6. If $m \neq 0$, return to step 1.

precisely than the CRC. Both schemes utilize the techniques in Section 10 to determine the optimal order of symbol transmission. The performance of the ROVA-based approach when additional symbols are chosen randomly is also shown. The optimal-order ROVA achieves a higher throughput at similar average blocklengths compared to the random-order ROVA.

The throughput R_t of the channel is plotted against the average blocklength λ for various values of the message length k, which is a hidden parameter of Fig. 10.1. The total number of symbols processed is $k+v$ for ROVA and $k+v+m$ for the CRC, where m is the number of CRC bits. The variables λ and R_t are defined as the following:

$$\lambda \leq \frac{1+\sum_{i=1}^{N-1} P_{\text{NACK}}(i)}{1-P_{\text{NACK}}(N)} \tag{10.17}$$

$$R_t = \frac{k}{\lambda}(1-P_{\text{UE}}) \tag{10.18}$$

where $P_{\text{NACK}}(i)$ is the probability that the receiver generates a NACK due to ROVA metric being below the threshold when i coded symbols have been received. P_{UE} is the probability of undetected error. The achievability curve for variable-length coding with feedback represents the random coding lower bound as defined in [WCW15a] according to [PPV11]. Similar to the results shown in [WCW15a], the simulations in Fig. 10.1 demonstrate that the throughput of this convolutional code exceeds the random-coding lower bound at short blocklengths. Fig. 10.1 also demonstrates that the CRC-based retransmission scheme performs poorly compared to ROVA at lower average blocklength. This is expected, as the additional bits required for the CRC are expensive when the number of transmitted symbols is low.

It is possible to form a hybrid scheme by combining ROVA and a CRC. If this decoder receives a message and the CRC does not pass, then the decoder choice is selected. If the message passes the CRC, then computations similar to the ROVA computations described in this thesis can aid in determining if the decoding is sufficiently reliable.

10.3 Correction of results in [BBK22]

The comparison between the ROVA-based retransmission scheme and the CRC-based retransmission scheme published in [BBK22] did not compare the two in a fair manner. This

section will correct the result shown for the CRC-based retransmission scheme while noting that the qualitative analysis of the comparison stay the same. The results in [BBK22] all share the same initial transmission lengths for the same values of the total number of input bits processed, which corresponds to the number of information bits k in Table .5. This initial transmission length was two-fifths of the total number of input bits. For the ROVA-based retransmission scheme, the empirical probability that the computed ROVA values were sufficiently high at the initial transmission rate was small. However, for smaller values of k, the CRC-based scheme could reliably terminate transmission with a smaller number of symbols than were initially transmitted. Fig. 10.3 shows the results when the initial transmission starts at one symbol and continues until the decoder terminates. It can be seen that as the average blocklength increases, the two curves converge on each other. The reason is that at larger values of k, the CRC-based scheme needs more than two-fifths of the bits to decode reliably. The CRC-based scheme performs better than was previously shown, but importantly the ROVA-based scheme continues to perform better at lower average blocklengths. The analysis performed in [BBK22] and in this thesis still hold true.

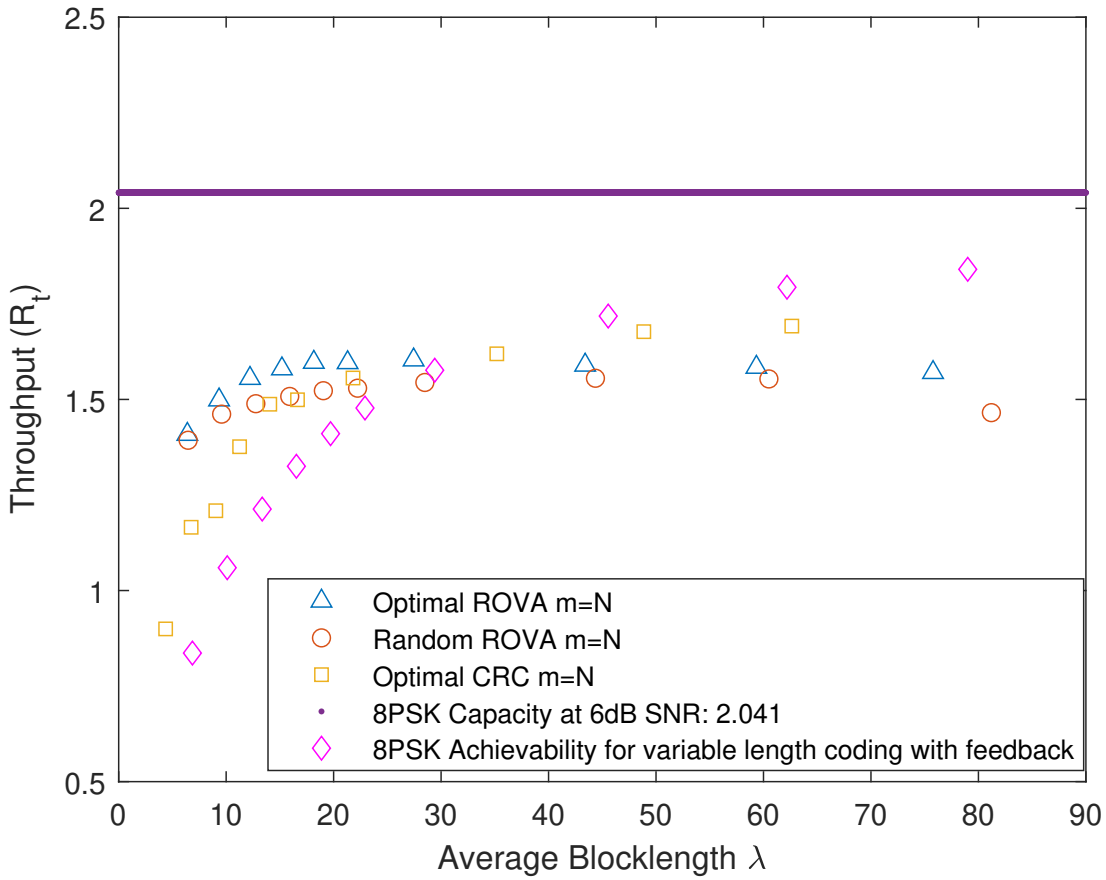

Figure 10.1: Short-blocklength performance of the m = N ROVA-based retransmission scheme over the AWGN channel with SNR 6.00 dB and target probability of error $\epsilon = 10^{-3}$ using a 64 state rate 1/3 code with 8-PSK modulation. The ROVA for terminated convolutional codes is used. The performance of a CRC-based retransmission scheme with the same characteristics is also shown. Each code shares the same set of the total number of input bits processed by the code: 15, 20, 25, 30, 35, 40, 50, 75, 100, 125.

This figure was previously presented in part in an IEEE TCOM journal paper [BBK22].

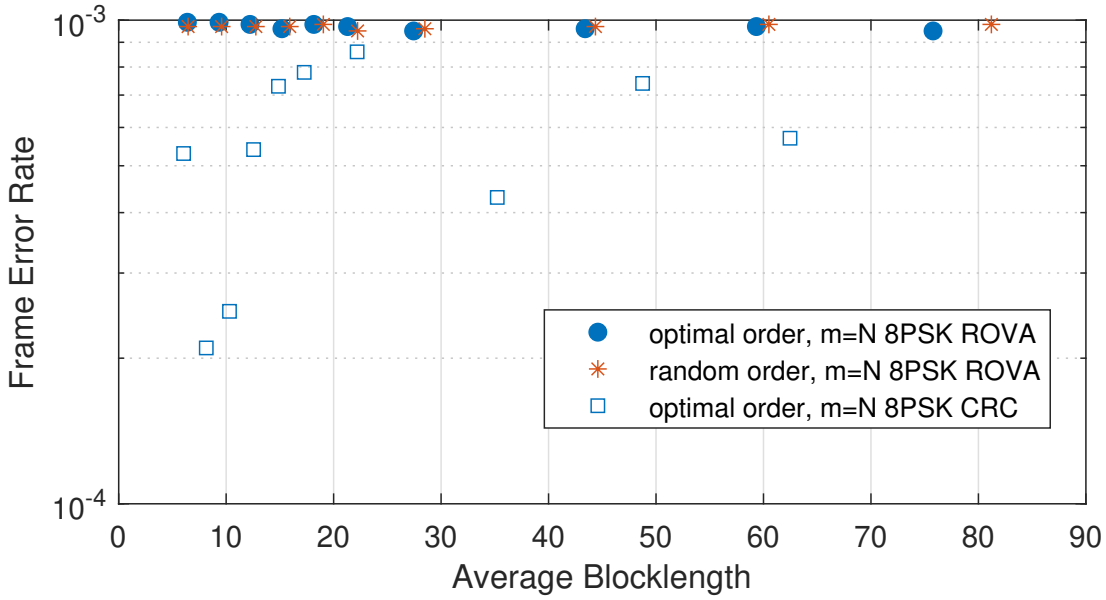

Figure 10.2: Frame error rate plotted against the average blocklength for each of the simulated points in Fig. 10.1. Each ROVA threshold and CRC polynomial is chosen such that the frame error rate is below target error rate ϵ. ROVA is able to target desired error rates more precisely than the CRC.

This figure was previously presented in part in an IEEE TCOM journal paper [BBK22].

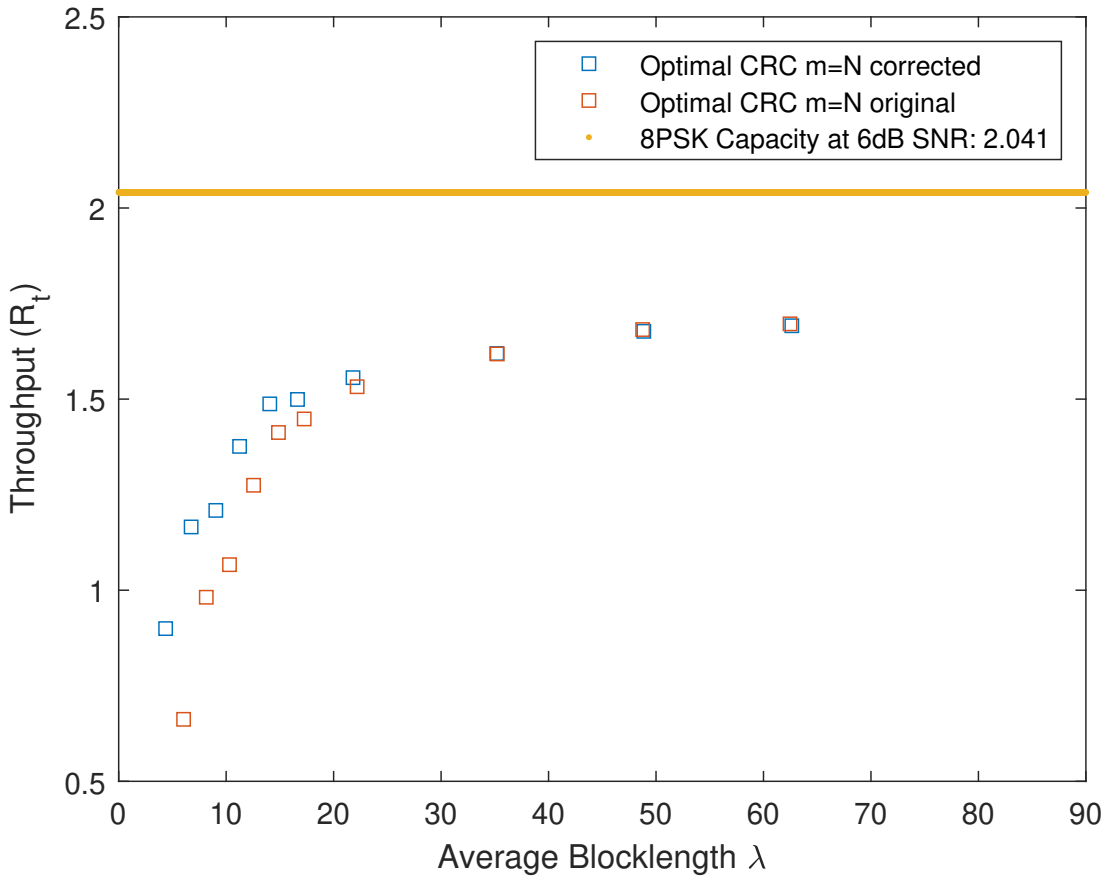

Figure 10.3: A previous paper [BBK22] computed a throughput vs. average blocklength comparison to ROVA using a CRC as a stopping condition. The initial number of transmitted symbols was too large at lower average blocklengths for the CRC, leading to a transmission rate that was too small at lower average blocklengths. The corrected curve represents simulations where transmission starts with a single symbol and continues to transmit additional symbols until the decoder terminates transmission. At higher average blocklengths, the original initial number of transmitted symbols was sufficient, leading to convergence of the results at higher average blocklength. The CRC still performs worse than ROVA at smaller average blocklengths, so the analysis in [BBK22] remains unchanged.

CHAPTER 11

Sequential Differential Optimization

[1]

11.1 The incremental redundancy accumulation cycle

Consider a system that communicates a k-bit message by using incremental redundancy to send up to m possible transmissions in an accumulation cycle. The transmissions have lengths of $l_1, ..., l_m$, where sending each additional transmission depends on ACK/NACK feedback. Each subsequenct attempt in the accumulation cycle has the advantage of a successively larger cumulative blocklength of N_i where

$$N_i = \sum_{j=1}^{i} l_j \tag{11.1}$$

The decision of whether to send an ACK to terminate the transmission or to send a NACK to request additional redundancy is based on some indicator of reliable decoding such as a cyclic redundancy check (CRC) code [KC04b] or comparing a soft indicator of decoding reliability to a threshold such as the ROVA algorithm of [WMW14]. The receiver does not know whether it has truly decoded correctly; the CRC or soft information threshold is designed to achieve a desired target probability of undetected error P_{UE}. A decoding error is only possible when the receiver makes a final decision and an ACK is sent, and all such errors are undetected.

Let P_{ACK}^{Nj} and $P_{NACK}^{N_j}$ be the marginal probabilities of a decoding "success" or "failure"

[1] This chapter was previously presented in part in an IEEE ISIT conference paper [WWB17].

based on the reliability indicator when the decoder is presented with a received codeword having blocklength N_j. Note that $P_{ACK}^{Nj} + P_{NACK}^{N_j} = 1$.

If decoding is still unsuccessful after all m decoding attempts in the accumulation cycle, the associated k-bit message is not lost, rather the transmission is attempted again from scratch. This is referred to by Heindlmaier and Soljanin in [HS14] as a fixed incremental redundancy scheme, but as shown in [HS14] the loss from the infinite incremental redundancy scheme where $m = \infty$ is small when the failure rate is low.

11.2 Optimizing the lengths $l_1, ..., l_m$ to maximize throughput

Let I be the number of successfully transmitted information bits in an accumulation cycle. Let N be the number of symbols transmitted in an accumulation cycle. The throughput rate R_T is defined as

$$R_T = \frac{E[I]}{E[N]} = \frac{k(1 - P_{NACK}^{N_m} - P_{UE})}{E[N]} \tag{11.2}$$

where

$$E[N] \approx N_1 P_{ACK}^{N_1} + \sum_{j=2}^{m} N_j [P_{ACK}^{N_j} - P_{ACK}^{N_{j-1}}] + N_m P_{NACK}^{N_m} \tag{11.3}$$

The expression for $E[N]$ in (11.3) is an approximation because it assumes that if an ACK was sent when decoding a message of length N_{j-1}, then certainly an ACK would also be sent when decoding the corresponding longer message with length N_j . While observed to be true for the non-binary LDPC codes explored in (11.3), this is not true in general and specifically not the case for the convolutional codes explored in this thesis. However, events where an ACK would be followed by a NACK are relatively rare, and ignoring this effect simply leads to a slight underestimate of throughput by under-counting the ACKs that occur for the first time on the j^{th} attempt.

To further simplify the optimization of R_T , noting that $E[I]$ is not affected much by varying the length N_j (indeed $E[I] \approx k$ for most systems of interest in which P_{NACK}^{Nm} and

P_{UE} are both small), maximizing R_T is essentially equivalent to minimizing $E[N]$.

11.3 An Improved Model for $P_{ACK}^{N_j}$

11.3.1 A Gaussian approximation of highest rate of ACK

The model proposed in [VRD16] approximates $P_{ACK}^{N_j}$ with a Gaussian distribution on the rate at which decoding is first successful. The model's parameters are the mean μ_s and variance σ_s^2 of the first successful decoding rate. Thus the approximation is

$$P_{ACK}^{N_j} = Q\left(\frac{\frac{k}{N_j} - \mu_s}{\sigma_s}\right) \tag{11.4}$$

where $Q(t)$ is the probability that a unit-variance, zero-mean Gaussian random variable (r.v.) takes a value larger than t. This model was shown to be extremely accurate for the cases studied in [VRD16] where the average blocklengths were all larger than 150. However, [WVW17] found that the approximation became inaccurate for average blocklengths less than 100.

Fig. 11.1 illustrates the accuracy problem identified in [WVW17] by comparing simulated throughput to throughput approximated using the GR model of [VRD16] as well as the model we will introduce in the next section. The figure shows how the throughput rate R_T increases as a function of the number of transmissions m in an accumulation cycle. The key observation for the purposes of this thesis is that for $k = 16$ and $k = 32$ with average blocklengths below 100 symbols, the GR model has a noticeable approximation error. For $k = 64$ with average blocklengths above 100 symbols, the GR model provides an excellent approximation, consistent with the results in [VRD16].

11.3.2 Polyanskiy's Gaussian information-density model

To improve on the model of [VRD16], we utilize the analysis of Polyanskiy et al. in [PPV11]. Consider a discrete memoryless channel in which the transmitted symbol is the random r.v.

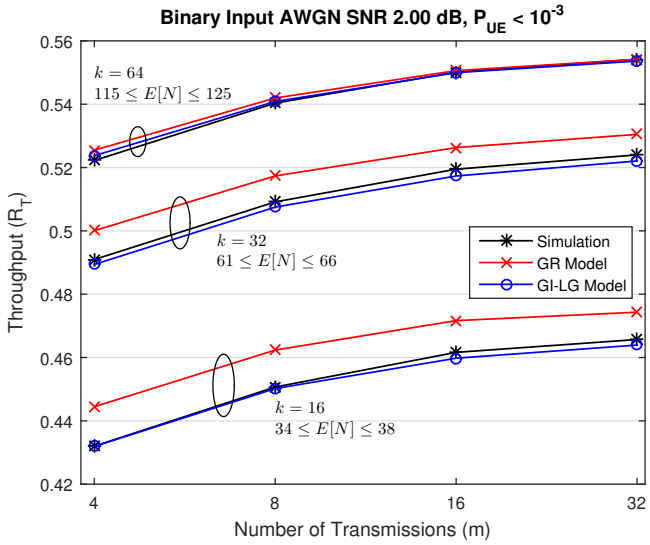

Figure 11.1: Throughput as a function of number of transmissions m in an incremental redundancy cycle for information lengths $k \in \{16, 32, 64\}$. Simulation performance is compared with GR and GI-LG approximations.

This figure was previously presented in part in an IEEE ISIT conference paper [WWB17].

X and the corresponding received symbol is the r.v. Y . In [PPV11], a fundamental tool used to establish the achievability results for communication with feedback is the information density, defined for each received symbol y corresponding to transmitted symbol x as

$$i(x, y) = \log_2 \frac{f_{Y|X}(y|x)}{f_Y(y)} \tag{11.5}$$

The accumulated information density for n symbols is

$$i(x^n, y^n) = \sum_{n}^{i=1} \log_2 \frac{f_{Y|X}(y_i|x_i)}{f_Y(y_i)} \tag{11.6}$$

The receiver does not know the true sequence of transmitted x values, and thus does not know the "true" accumulated information density. However, at least in theory, the receiver can compute a tentative accumulated information density for each possible codeword x_n. The approach of [PPV11] is to set a threshold γ and terminate decoding whenever one of these tentative accumulated information densities exceeds γ.

The sequence of information densities, one associated with each symbol, is independent

and identically distributed (i.i.d.). Furthermore, the mean and variance of the symbol-wise information densities can be computed from the channel model directly, with no need for simulation. In particular, the mean of the symbol-wise information density is the mutual information $I(X;Y)$ of the channel.

For the example of the BI-AWGN channel with $x \in \pm 1$, the symbol-wise information density has the p.d.f.

$$1 - \log_2(1 + \exp -2(z+1)/(\sigma^2)) \tag{11.7}$$

where z is the zero-mean Gaussian noise $y - x$. The mean and variance are obtained by computing the appropriate expectations with respect to the Gaussian p.d.f. of z. Because the sequence of information densities is i.i.d., Polyanskiy et al. have modeled the accumulated information density as a Gaussian r.v. [PPV10]. This "normal approximation" is quite accurate even for a relatively small number of symbols.

Consider a rate-compatible code communicating k bits of information by transmitting a growing amount of redundancy. Asymptotically (as k grows large), information theory informs us that the receiver should be able to decode successfully when the accumulated information density $i(x_n, y_n)$ slightly exceeds k and that this would happen approximately when n exceeds $k/I(X;Y)$. By using a fixed constant γ, the analysis in [PPV11] adds a fixed "gap" ϵ to the asymptotically limiting case where γ would simply be k. In other words, $\gamma = k + \epsilon$, where ϵ is the constant information gap needed to ensure that a particular desired frame error rate is achieved with a particular finite k.

Thus, the model for $P_{ACK}^{N_j}$ due to the analysis of [PPV11], [PPV10] is

$$P_{ACK}^{N_j} = Q\left(\frac{k + \epsilon - N_j \mu_i}{\sigma_i \sqrt{N_j}}\right) \tag{11.8}$$

$$= Q\left(\frac{\frac{k}{\sqrt{N_j}} + \frac{\epsilon}{\sqrt{N_j}} - \mu_i}{\frac{\sigma_i}{\sqrt{N_j}}}\right) \tag{11.9}$$

where μ_i and σ_i^2 are the mean and variance of the symbol-wise information density, which can be derived analytically from the channel probability distribution. In contrast

to the model of [VRD16], the accumulated information density is modeled as a Gaussian rather than the highest rate of successful being modeled as a Gaussian. Note that while (11.4) evaluates the tail of a Gaussian that always has the same variance, the comparable expression in (11.9) evaluates the tail of a sequence of Gaussians whose variance is decreasing with blocklength N_j.

Despite the theoretical appeal of the Gaussian information with constant gap (GI-CG) model in (11.8)-(11.9), it turns out also to be inaccurate for the TBCC using ROVA for $k = 16$. Fig. 11.2 shows the probability of ACK as a function of blocklength N_j according to simulation, the GR model, the GI-CG model, and the model we will discuss in the next subsection. The GI-CG model fails to capture the behavior of the simulation.

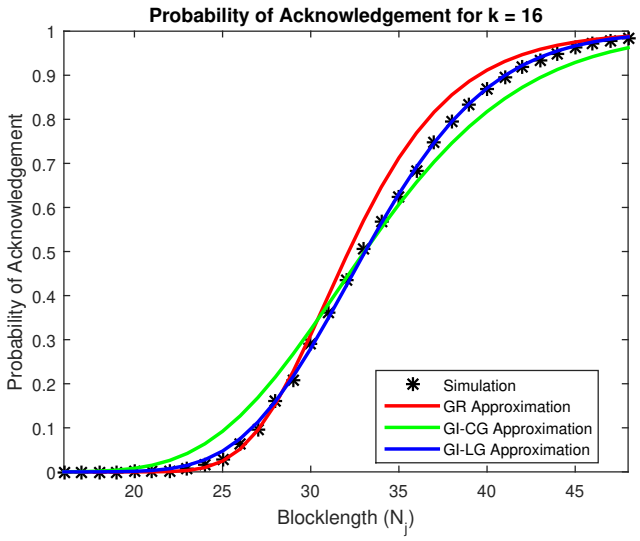

Figure 11.2: Probability of acknowledgment $P_{ACK}^{N_j}$ for TBCC-ROVA with $k = 16$ as a function of blocklength according to simulation and according to the Gaussian rate, and Gaussian information density with constant gap, and Gaussian information density with linear gap models.

This figure was previously presented in part in an IEEE ISIT conference paper [WWB17].

11.3.3 Gaussian information density with a linear coding gap

The inaccuracy of the GI-CG model is caused by the assumption that the gap ϵ remains constant for a variable-length code as its blocklength is increased with incremental redundancy. This turns out not to be the case for the rate-compatible TBCC examined in this thesis and

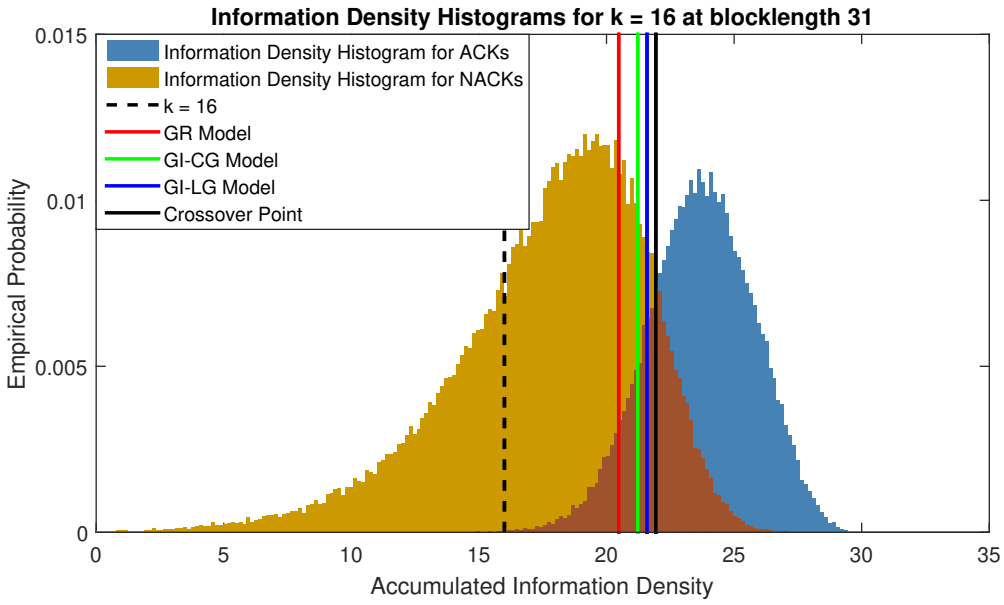

Figure 11.3: Histograms of information density at successful and unsuccessful decoding showing the crossover point.

This figure was previously presented in part in an IEEE ISIT conference paper [WWB17].

perhaps other variable-length codes with short average blocklengths.

To examine how the gap behaves as a function of blocklength, we study the amount of accumulated information density when the TBCC-ROVA algorithm triggered an ACK. Fig. 11.3 shows separate histograms of information density for length $N_j = 33$ codewords for which an ACK was declared and for which a NACK was declared. Unlike the algorithm analyzed theoretically in [PPV11], the TBCC does not declare an ACK explicitly based on information density. However, using histogram data as shown in Fig. 11.3, the crossover point can be determined so the approximate amount of information density required to trigger an ACK is identified.

According to (11.8) of the GI-CG model, the amount of information density required to trigger an ACK should remain constant at $k + \epsilon$. However, this is not the behavior observed for the $k = 16$ TBCC using ROVA. Fig. 11.4 plots the accumulated information density crossover points (as shown by example in Fig. 11.3) as a function of blocklength. Fig. 11.4 shows that the crossover points decrease as a function of blocklength.

To capture the behavior observed in Fig. 11.4, a linear gap term α is added to the

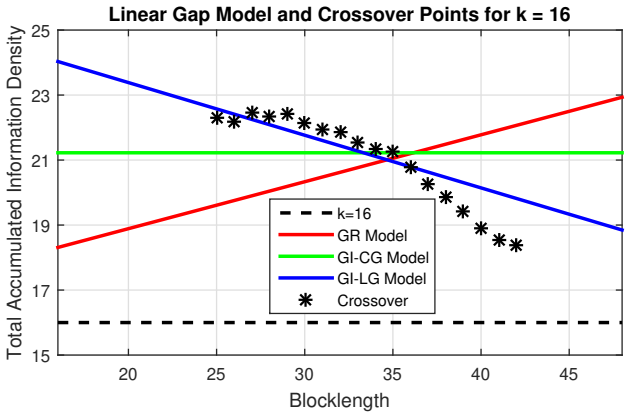

Figure 11.4: Information density empirical crossover point between successful and unsuccessful decoding, information density threshold estimated by Gauss- Newton using the linear gap model, and the reference value of $k = 16$.

This figure was previously presented in part in an IEEE ISIT conference paper [WWB17].

argument of the Q function to create the Gaussian information density with linear gap (GI-LG) model, with $P_{ACK}^{N_j}$ shown below:

$$P_{ACK}^{N_j} = Q\left(\frac{k + \epsilon - N_j(\mu_i + \alpha)}{\sigma_i\sqrt{N_j}}\right) \tag{11.10}$$

$$= Q\left(\frac{\frac{k}{\sqrt{N_j}} + \frac{\epsilon}{\sqrt{N_j}} - (\mu_i + \alpha)}{\frac{\sigma_i}{\sqrt{N_j}}}\right) \tag{11.11}$$

where μ_i and σ_i are computed from the channels p.d.f. for symbol-wise information density. The parameters ϵ and α in the linear gap model must be estimated. Monte-Carlo simulation provides an empirical estimate of $P_{ACK}^{N_j}$, which can be used to estimate ϵ and α through least squares optimization.

Let $p = (\epsilon, \alpha)$ be the parameter vector. The objective function that needs to be minimized is

$$C(\mathbf{p}) = \sum_{N_j=b}^{N_j=e} \left(P_{ACK}^{N_j} - \hat{P}_{ACK}^{N_j}\right)^2 \tag{11.12}$$

where $P_{ACK}^{N_j}$ is the modeled probability of acknowledgment (11.10) and $\hat{P}_{ACK}^{N_j}$ is the empirical probability of acknowledgment, and b and e represent the shortest and longest blocklengths

of the range of blocklengths for the optimization. The gradient of the objective function can be obtained as

$$\nabla C(\mathbf{p}) = 2 \cdot (\mathbf{J_G}(\mathbf{p}))^T \cdot \mathbf{G}(\mathbf{p}) \tag{11.13}$$

where $\mathbf{J_G}(\mathbf{p})$ is the Jacobian matrix of the difference vector $\mathbf{G}(\mathbf{p})$, and $\mathbf{G}(\mathbf{p})$ can be calculated as

$$\mathbf{G}(\mathbf{p}) = \left[P^b_{ACK} - \hat{P}^b_{ACK},\ \ P^{b+1}_{ACK} - \hat{P}^{b+1}_{ACK},\ \ \cdots,\ \ P^e_{ACK} - \hat{P}^e_{ACK}\right]' \tag{11.14}$$

We use the Gauss-Newton algorithm [Ake96] to estimate $\mathbf{p}$ by solving a nonlinear least squares problem. The algorithm minimizes the objective function through a sequence of linear approximations of the difference vector $\mathbf{G}(\mathbf{p})$. The parameters obtained by Gauss-Newton for all three models can be found in Table 11.1. Note that with the ϵ and α identified by Gauss-Newton, the GI-LG model is shown to closely approximate performance in Fig. 11.2, which was our goal.

Table 11.1: Parameters obtained for the three models via Gauss-Newton Optimization

	μ_i or μ_s	σ_i^2 or σ_s^2	ϵ	α
GR	0.497716	0.00522976		
GI-CG	0.642149	0.606315	5.22411	
GI-LG	0.642149	0.606315	10.6350	0.162808

Examining Table 11.1 in conjunction with Fig. 11.4 reveals the limitations of the GR and GI-CG models. With respect to the GI-CG model, the lack of the linear term α forced the ϵ parameter to place the horizontal line of the GI-CG model in Fig. 11.4 where it would best intersect the actual line of crossover points. Looking at the GR model, we can see that μ_s is essentially $\mu_i + \alpha$ with $\alpha = -0.1444333$. Because the GR model lacks the term ϵ, the y-intercept must be at $k = 16$, so that α is chosen to provide a slope that allows the line to best intersect the actual crossover points in a reasonable way. Only the GI-LG model has both the α of the GR model and the ϵ of the GI-CG model so that it has the necessary degrees of freedom to properly model the crossover points of Fig. 11.4.

11.4 Sequential Differential Optimization

A key benefit of the GR model of [VRD16] is that it facilitates SDO, which greatly simplifies the optimization of the incremental lengths $l_1, ..., l_m$ to maximize R_T. The SDO approach applies equally well to all three models described in this chapter: GR, GI-CG, and the GI-LG model which turns out to be the most accurate.

With the approximations $E[I] \approx k$, we seek to minimize $E[N]$ as approximated by (11.3). Over a range of possible N_1 values, SDO optimizes $\{N_2, ..., N_m\}$ to minimize $E[N]$ for each fixed value of N_1.

For each $j \in \{2, ..., m\}$, the optimal value of N_j is found by setting $\frac{\partial E[N]}{\partial N_{j-1}} = 0$, yielding a sequence of relatively simple computations. In other words, we select the N_j that makes our previous choice of N_{j-1} optimal in retrospect. For example, to find N_2 we compute the derivative

$$\frac{\partial E[N]}{\partial N_1} = P_{ACK}^{N_1} + (N_1 - N_2)P_{ACK}^{'N_1} = 0 \tag{11.15}$$

and solve for N_2 as

$$N_2 = \frac{P_{ACK}^{N_1} + N_1 P_{ACK}^{'N_1}}{P_{ACK}^{'N_1}} \tag{11.16}$$

where for the GI-LG model $P_{ACK}^{'N_j}$ is defined as

$$\frac{\partial P_{ACK}^{N_j}}{\partial N_j} = \frac{(\mu_i + \alpha)N_j + k + \epsilon}{2\sqrt{2\pi}\sigma_i N_j^{1.5}} e^{-\frac{(k+\epsilon-(\mu_i+\alpha)N_j)^2}{2\sigma_i^2 N_j}} \tag{11.17}$$

For j > 2, $\frac{\partial E[N]}{\partial Nj-1} = 0$ depends only on $\{N_{j-2}, N_{j-1}, N_j\}$ as follows:

$$\frac{\partial E[N]}{\partial N_{j-1}} = P_{ACK}^{N_{j-1}} + (N_{j-1} - N_j)P_{ACK}^{'N_{j-1}} - P_{ACK}^{N_{j-1}} \tag{11.18}$$

Thus we can solve for N_j as

$$N_j = \frac{P_{ACK}^{N_{j-1}} + N_{j-1}P_{ACK}^{'N_{j-1}} - P_{ACK}^{N_{j-2}}}{P_{ACK}^{'N_{j-1}}} \tag{11.19}$$

For each possible N_1, SDO can be used to produce an infinite sequence of N_j values using (11.16) and (11.19). Each such sequence is an optimal sequence of increment lengths

for a given density of ACK/NACK transmissions on the time axis. As N_1 increases, the density decreases. Using SDO to compute the optimal m points is equivalent to selecting the most dense SDO-optimal sequence that when truncated to m points results in the highest throughput.

11.4.1 Application of SDO to Variable-Length Coding with Higher Order Modulation

As an extension to the results shown in Fig 10.1, we now apply the SDO algorithm to determine the optimal transmission lengths when we limit the system to m transmissions. For an 8-PSK modulated signal, if the probability of selecting input symbol $x \in \mathbb{R}^2$ is uniformly distributed around the constellation, the symbol-wise information density takes the form

$$i(x,y) = \log_2\left(\frac{f_{Y|X}(y|x)}{f_Y(y)}\right) = \log_2\left(\frac{f_{Y|X}(y|x=x_0)}{f_Y(y)}\right) \tag{11.20}$$

where the second equivalence is due to symmetry in the constellation points. For an 8-PSK modulated signal on an AWGN channel,

$$f_{Y|X}(y|x=x_0) = \frac{1}{2\pi\sigma^2} e^{-\frac{||y-x_0||_2^2}{2\sigma^2}} \tag{11.21}$$

where x_0 is the transmitted constellation point and

$$f_Y(y) = \sum_{i=0}^{7} f_{Y|X}(y|x=x_i) \tag{11.22}$$

where x_i is the i^{th} constellation point. Taking the symbol-wise information density mean and variance to be

$$\mu_i = E[i(x,y)] \tag{11.23}$$

$$= \int_{y\in\mathbb{R}^2} f_{Y|X}(y|x=x_0)\log_2\frac{f_{Y|X}(y|x=x_0)}{f_Y(y)}dy \tag{11.24}$$

$$\sigma_i^2 = E[i(x,y)^2] - \mu_i^2 \tag{11.25}$$

$$= \int_{y\in\mathbb{R}^2} f_{Y|X}(y|x=x_0)\left(\log_2\frac{f_{Y|X}(y|x=x_0)}{f_Y(y)}\right)^2 dy - \mu_i^2 \tag{11.26}$$

we numerically integrate over a large area of y to obtain $\mu_i = 2.0428$ and $\sigma_i = 1.0938$ for an SNR of 6dB. Each point in Fig. 11.5 corresponds to a different message length k, and therefore has a unique value of α and ϵ for the GI-LG model that models the probability that ROVA returns an ACK. These α and ϵ values are detailed in Table. .6 in the appendix.

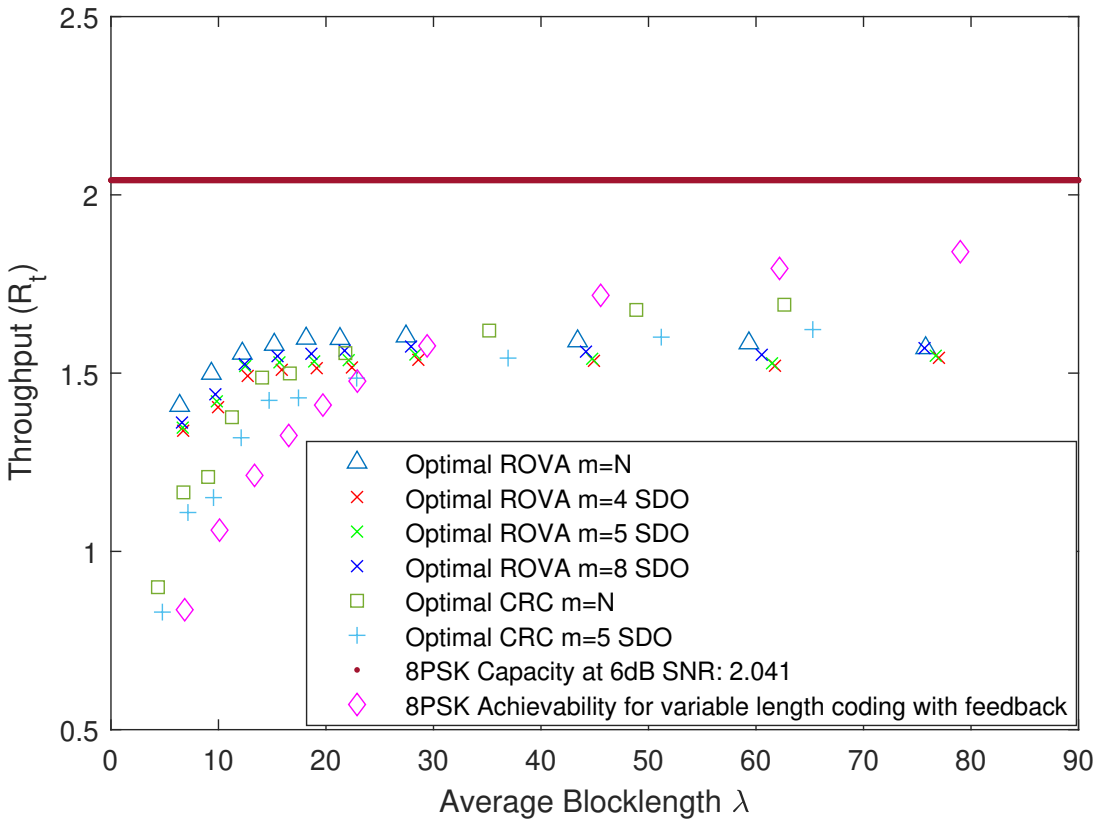

Figure 11.5: Results of the ROVA-based retransmission scheme and CRC-based retransmission scheme when utilizing the SDO algorithm. Predictably, the performance of $m = M$ retransmission converges to the $M = N$ performance as M increases.

Fig. 11.5 shows the performance of the feedback system when using $m = \{4, 5, 8\}$ transmissions for ROVA and $m = 5$ transmissions for the CRC. The performance of the system only slightly decreases when limited to these transmission lengths. Predictably, the performance of $m = M$ transmission converges to $m = N$ transmission as M increases.

CHAPTER 12

Conclusion

This thesis compares the accuracy and complexity of Raghaven and Baum's ROVA, Polyanskiy's AID, and Fricke and Hoeher's approximation of ROVA. It turns out that AID is far less accurate than ROVA because it considers all x^{n_c} sequences as possible rather than restricting attention only to valid codewords. When AID is modified to consider only valid codewords, it becomes equivalent to ROVA, and reveals a lower complexity approach to ROVA as compared to the algorithm presented in [RB98]. This CID implementation of ROVA achieves similar complexity reduction as Fricke and Hoeher while computing the exact probability rather than an approximation.

This thesis derives an analytical expression for the random variable describing the correct decoding probability computed by ROVA and uses this expression to characterize how the probabilities of correct decoding, undetected error, and negative acknowledgement behave as a function of the selected threshold for reliable decoding.

This thesis examines both the complexity and the simulation time of ROVA, CID, AID, and the Fricke and Hoeher approximation to ROVA. This thesis also derives an expression for the union bound on the frame error rate for zero-terminated trellis codes with punctured symbols and uses it to optimize the order of symbol transmission in an incremental retransmission scheme.

This thesis compares the performance of an incremental retransmission scheme using ROVA as a stopping condition to one that uses a CRC as a stopping condition. This thesis found that that at short average blocklengths, the scheme using ROVA as a stopping condition outperforms the scheme using a CRC as a stopping condition due to the extra overhead required by the CRC.

This thesis concludes by applying the SDO algorithm to determine the transmission lengths in an incremental transmission scheme to maximize the throughput when limiting the maximum number of transmissions. This thesis demonstrates that a relatively small number of feedback transmissions can achieve a throughput similar to systems that transmit feedback after every symbol.

Short blocklength messages are commonly used in modern communications systems for a variety of tasks. Whether they are used in text messages, control messaging, or low power internet-of-things sensors, assessing the reliability of these transmissions in an efficient manner can be crucial. This thesis presents tools and techniques for improving the efficiency of these assessments.

APPENDIX A

Table .1: Complexity of a ROVA iteration when all trellis states are active.

Algorithm Step	Complexity
Step 2	$N_s N_b$ metric computations
Step 3	$N_s(N_b + 1)$ additions and $N_s N_b$ multiplications
Step 4	Computation of one multiplicative inverse: Δ_m^{-1} $2 \times N_s$ multiplications to compute $P_{j^*}^m$ $N_s(N_b + 1)$ multiplications and $2N_s N_b$ adds to compute $\bar{P}_{j^*}^m$

Table .2: Complexity of an iteration of the Fricke and Hoeher approximation of ROVA when all trellis states are active.

Algorithm Step	Complexity
Step 2	$N_s N_b$ metric computations
Step 3	N_s multiplications to compute Γ_m^j $N_s(N_b - 1)$ adds. and mults. for denom. of (7.3) N_s divisions to compute (7.3) for each state
Step 4	n_c multiplies to compute $\prod_{m=1}^{n_c} \hat{P}_m(i^*, j^*)$.

Table .3: Complexity of an iteration of the AID algorithm when all trellis states are active.

Algorithm Step	Complexity
Step 2	$N_s N_b$ metric computations
Step 3	N_s multiplications to compute Γ_m^j
Step 4	$\lvert\mathcal{X}\rvert$ metric computations to compute $f_Y(y_m \lvert x)$ $\lvert\mathcal{X}\rvert$ adds. and mults. to compute $f_Y(y_m)$ 1 multiply to compute $\Pi(m)$

Table .4: Complexity of an iteration of the CID implementation of ROVA when all trellis states are active.

Algorithm Step	Complexity
Step 2	$N_s N_b$ metric computations
Step 3	N_s multiplications to compute Γ_m^j
Step 4	$N_s N_b$ adds. and mults. to compute Z_m^j.

Table .5: Implementation details of the comparison between incremental retransmission schemes utilizing either ROVA or the CRC in Fig. 10.1. k is the number of information bits for ROVA, m is the number of CRC bits, k' is the number of information bits for the CRC, λ is the average blocklength, and R_T is the throughput. Both the ROVA and CRC process an additional v termination bits.

ROVA with optimal ordering			CRC with optimal ordering			
k	λ	R_T	m	$k' = k - m$	λ	R_T
9	6.39	1.409	5	4	6.04	0.662
14	9.34	1.499	6	8	8.15	0.982
19	12.22	1.555	8	11	10.31	1.067
24	15.19	1.580	8	16	12.55	1.275
29	18,16	1.597	8	21	14.87	1.413
34	21.30	1.596	9	25	17.26	1.448
44	27.44	1.603	10	34	22.19	1.532
69	43.51	1.590	12	57	35.23	1.618
94	59.34	1.584	12	82	48.75	1.682
119	75.78	1.570	13	106	62.47	1.697

Table .6: Implementation details of the SDO algorithm for the 8PSK AWGN channel using either ROVA or a CRC as the reliability metric

ROVA			CRC			
k	α	ϵ	m	k	α	ϵ
9	0.5225	6.3865	5	4	-0.2354	1.8855
14	0.9102	13.1134	6	8	-0.4238	0.8132
19	0.8294	15.3622	8	11	-0.2189	4.0857
24	0.5282	14.4189	8	16	-0.2024	3.9213
29	0.3232	13.7053	8	21	-0.2184	3.3779
34	0.3969	17.7099	9	25	-0.2345	3.8005
44	0.3152	20.7218	10	34	-0.3790	1.0030
69	-0.0020	19.4503	12	57	-0.5963	-7.3907
94	-0.3235	8.4628	12	82	-0.6856	-16.8761
119	-0.5213	-3.2577	13	106	-0.7253	-24.7011

REFERENCES

[3GP] "European Telecommunications Standards Institute 3GPP TS 25.212 version 7.0.0 Release 7.".

[Ake96] Bjorck Ake. *Numerical methods for least squares problems*. SIAM, 1996.

[BBK22] Alex Baldauf, Adam Belhouchat, Shakeh Kalantarmoradian, Alethea Sung-Miller, Dan Song, Nathan Wong, and Richard D. Wesel. "Efficient computation of Viterbi decoder reliability with an application to variable-length coding." *IEEE Transactions on Communications*, **70**(9):5711–5723, 2022.

[BCJ74] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)." *IEEE Transactions on Information Theory*, **20**(2):284–287, 1974.

[Bla12] Richard Blahut. *Algebraic codes for data transmission*, p. 62–62. Cambridge Univ. Press, 2012.

[BLW02] A. Bernard, Xueting Liu, R.D. Wesel, and A. Alwan. "Speech transmission using rate-compatible trellis codes and embedded source coding." *IEEE Transactions on Communications*, **50**(2):309–320, 2002.

[Bur76] Marat V Burnashev. "Data transmission over a discrete channel with feedback. Random transmission time." *Problems of Information Transmission*, **12**(4):10–30, 1976.

[CHI98] D. J. Costello, J. Hagenauer, H. Imai, and S. B. Wicker. "Applications of Error Control Coding." *IEEE Trans. Inform. Theory*, **44**(6):2531–2560, Oct. 1998.

[EST18] *"ETSI TS 136 212 V15.3.0 LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (3GPP TS 36.212 version 15.3.0 Release 15)"*, 2018.

[FH07] Justus Ch. Fricke and Peter A. Hoeher. "Word Error Probability Estimation by Means of a Modified Viterbi Decoder." In *2007 IEEE 66th Vehicular Technology Conference*, Sep 2007.

[FH09] Justus Fricke and Peter Hoeher. "Reliability-based retransmission criteria for hybrid ARQ." *IEEE Transactions on Communications*, **57**(8):2181–2184, Aug 2009.

[Hag88] J. Hagenauer. "Rate-compatible punctured convolutional codes (RCPC codes) and their applications." *IEEE Transactions on Communications*, **36**(4):389–400, 1988.

[HS14] Michael Heindlmaier and Emina Soljanin. "Isn't hybrid arq sufficient?" *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2014.

[KC04a] P. Koopman and T. Chakravarty. "Cyclic redundancy code (CRC) polynomial selection for embedded networks." In *Int. Conf. Dependable Systems and Networks*, pp. 145–154, June 2004.

[KC04b] P. Koopman and T. Chakravarty. "Cyclic redundancy code (CRC) polynomial selection for embedded networks." *International Conference on Dependable Systems and Networks, 2004*, 2004.

[LC04] Shu J. Lin and Daniel J. Costello. *Error control coding: fundamentals and applications*. Prentice-Hall, 2nd edition, 2004.

[LDW15] Chung-Yu Lou, Babak Daneshrad, and Richard D. Wesel. "Convolutional-Code-Specific CRC Code Design." *IEEE Transactions on Communications*, **63**(10):3459–3470, Oct 2015.

[LMS07] C. Lott, O. Milenkovic, and E. Soljanin. "Hybrid ARQ: Theory, state of the art and future directions." In *IEEE Inf. Theory Workshop (ITW)*, Bergen, Norway, Jul. 2007.

[PPV10] Yury Polyanskiy, H. Vincent Poor, and Sergio Verdu. "Channel coding rate in the finite blocklength regime." *IEEE Transactions on Information Theory*, **56**(5):2307–2359, 2010.

[PPV11] Y. Polyanskiy, H. V. Poor, and S. Verdu. "Feedback in the non-asymptotic regime." *IEEE Trans. Inf. Theory*, **57**(8):4903 – 4925, August 2011.

[RB98] A. Raghavan and C. Baum. "A reliability output Viterbi algorithm with applications to hybrid ARQ." *IEEE Trans. Inf. Theory*, **44**(3):1214–1216, May 1998.

[Ric94] M. Rice. "Comparative analysis of two realizations for hybrid-ARQ error control." In *1994 IEEE GLOBECOM. Communications: Communications Theory Mini-Conference Record,*, pp. 115–119, 1994.

[RL09] William E. Ryan and Shu Lin. *Channel codes: classical and modern*, p. 148. Cambridge University Press, 2009.

[Sha56] C. Shannon. "The zero error capacity of a noisy channel." *IEEE Transactions on Information Theory*, **2**(3):8–19, 1956.

[Vit67] A. Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." *IEEE Transactions on Information Theory*, **13**(2):260–269, 1967.

[VRD16] Kasra Vakilinia, Sudarsan V. Ranganathan, Dariush Divsalar, and Richard D. Wesel. "Optimizing transmission lengths for limited feedback with Nonbinary LDPC examples." *IEEE Transactions on Communications*, **64**(6):2245–2257, 2016.

[WCW15a] Adam R. Williamson, Tsung-Yi Chen, and Richard D. Wesel. "Variable-Length Convolutional Coding for Short Blocklengths With Decision Feedback." *IEEE Trans. on Comm.*, **63**(7):2389–2403, Jul 2015.

[WCW15b] Adam R. Williamson, Tsung-Yi Chen, and Richard D. Wesel. "Variable-Length Convolutional Coding for Short Blocklengths With Decision Feedback." *IEEE Transactions on Communications*, **63**(7):2395, 2015.

[WDY21] Thomas Wiegart, Francesco Da Ros, Metodi Plamenov Yankov, Fabian Steiner, Simone Gaiarin, and Richard D. Wesel. "Probabilistically Shaped 4-PAM for Short-Reach IM/DD Links With a Peak Power Constraint." *Journal of Lightwave Technology*, **39**(2):400–405, 2021.

[Wes04] Richard D. Wesel. "Reduced-state representations for trellis codes using constellation Symmetry." *IEEE Transactions on Communications*, **52**(8):1302–1310, Aug 2004.

[WL98] Richard D Wesel and Xueting Liu. *Analytic Techniques for Periodic Trellis Codes**. University of Illinois Urbana-Champaign, 1998.

[WLS00] R.D. Wesel, Xueting Liu, and Wei Shi. "Trellis codes for periodic erasures." *IEEE Transactions on Communications*, **48**(6):938–947, 2000.

[WMW14] Adam R. Williamson, Matthew J. Marshall, and Richard D. Wesel. "Reliability-Output Decoding of Tail-Biting Convolutional Codes." *IEEE Transactions on Communications*, **62**(6):1768–1778, Jun 2014.

[WVW17] Nathan Wong, Kasra Vakilinia, Haobo Wang, Sudarsan V. Ranganathan, and Richard D. Wesel. "Sequential differential optimization of incremental redundancy transmission lengths: An example with tail-biting convolutional codes." *2017 Information Theory and Applications Workshop (ITA)*, 2017.

[WWB17] Haobo Wang, Nathan Wong, Alexander M. Baldauf, Christopher K. Bachelor, Sudarsan V. Ranganathan, Dariush Divsalar, and Richard D. Wesel. "An information density approach to analyzing and optimizing incremental redundancy with feedback." *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017.

[WWB18] R. D. Wesel, N. Wong, A. Baldauf, A. Belhouchat, A. Heidarzadeh, and J. Chamberland. "Transmission Lengths that Maximize Throughput of Variable-Length Coding & ACK/NACK Feedback." In *IEEE Global Communications Conference*, Abu Dhabi, UAE., Dec 2018.

[YI80] H. Yamamoto and K. Itoh. "Viterbi decoding algorithm for convolutional codes with repeat request." *IEEE Transactions on Information Theory*, **26**(5):540–547, 1980.

[YRW18] H. Yang, S. V. S. Ranganathan, and R. D. Wesel. "Serial List Viterbi Decoding with CRC: Managing Errors, Erasures, and Complexity." In *IEEE Global Communications Conference*, Abu Dhabi, UAE., Dec 2018.