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

A. Baldauf, A. Belhouchat, S. Kalantarmoradian, A. Sung-Miller, D. Song, N. Wong, R. D. Wesel
University of California, Los Angeles, CA
{ambaldauf19,abelhouchat,dansong0729}@gmail.com
{shakehk,aletheasm,nsc.wong,wesel}@ucla.edu

***Abstract*—This paper 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 lower complexity approximation of ROVA. It turns out that AID is far less accurate than ROVA in practice. This paper proposes codeword information density (CID), which modifies AID to improve its accuracy and leads to a lower-complexity implementation of ROVA. The paper 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 paper examines both the complexity and the simulation time of ROVA, CID, AID, and the Fricke and Hoeher approximation to ROVA. This paper 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 paper concludes by comparing the performance of an incremental retransmission scheme using ROVA as a stopping condition to one that uses a CRC as a stopping condition.**

I. Introduction

Cyclic redundancy checks (CRCs) are often used to detect errors in convolutional codewords [2]–[6]. CRCs play an important role in many incremental redundancy hybrid automatic repeat requests (ARQ) [6]–[10] but add overhead that can be significant for short block-lengths. As pointed out by [10], an alternative to using a CRC is to directly consider the reliability of the Viterbi decoder output as was proposed by Yamamoto and Itoh [11]. Ragavan and Baum [12] proposed the reliability-output Viterbi algorithm (ROVA) as an improvement to [11]. 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 [13]–[15] to decide whether to request additional redundancy in a hybrid ARQ without the need for a CRC. For [13], ROVA was adapted as described in [16] for tail-biting convolutional codes. Although ROVA calculates codeword error probability exactly, it suffers from high complexity. Fricke and Hoeher [15] developed an approximation of ROVA that reduces complexity.

This work was presented in part at the 52nd Asilomar Conference on Signals, Systems, and Computers [1]. This research is supported by National Science Foundation (NSF) grant CCF-1955660. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect views of the NSF.

A. Overview

For zero-terminated convolutional codes (ZTCCs), this paper explores an alternative to ROVA for controlling an incremental redundancy hybrid ARQ that is based on information density [17]. Symbol-wise accumulated information density (AID) as proposed by Polyanskiy et al. [17] 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 paper 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, we propose 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 has similar accuracy to ROVA, but lacks the exactness of CID.

We develop 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 paper are then compared.

We derive 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 [18]–[20]. We use this method to optimize the order of additional symbol transmission in incremental retransmission schemes. We then extend the work by Williamson [13] 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, we demonstrate that using ROVA as a reliability metric for incremental retransmission outperforms using a CRC as a reliability metric at short blocklengths.

B. Contributions and Organization

The contributions in this paper are summarized as follows: We note that the last three bullets are new material compared with our precursor conference paper. Our current paper also provides a more in-depth complexity comparison between each reliability metric.

- This paper offers a method of exactly computing the probability that a Viterbi decoder has selected the correct codeword. The new method has reduced complexity compared to the method of ROVA introduced by [12].
- This paper examines symbol and codeword information density metrics, respectively AID and CID, for deciding whether to accept a Viterbi decoder decision.
- This paper derives an analytical expression for ROVA over an AWGN channel with a fixed SNR and demonstrates how to compute the probability of correct decoding, probability of incorrect decoding, and probability of NACK for a given ROVA threshold.
- This paper compares the complexity and accuracy of ROVA, the Fricke and Hoeher ROVA approximation, AID, and the CID implementation of ROVA.
- This paper derives an expression for the union bound on the frame error rate for ZTCCs with punctured symbols.
- This paper details a method of optimizing the order of additional symbol transmissions in retransmission schemes by performing a greedy search utilizing a union bound on the frame error rate for ZTCCs.
- This paper shows that using ROVA in a reliability-based retransmission scheme can achieve higher throughput than both the variable length coding with feedback achieveability and the CRC-based method at short blocklengths using 8PSK modulation.

Sec. II reviews the ROVA algorithm of [12]. Sec. III reviews AID of [17] and shows that it is much less predictive of reliable decoding than ROVA. Sec. IV 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 [12]. Sec. V 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. Sec. VI analyzes and compares the complexity of each of the four reliability metrics presented in this paper. Sec. VII 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. VIII. Sec. VIII produces an analogous plot for short-blocklength performance of a reliability-based retransmission scheme using either a CRC or ROVA as shown in [13], but for a higher order modulation compared to BPSK. Sec. IX concludes the paper.

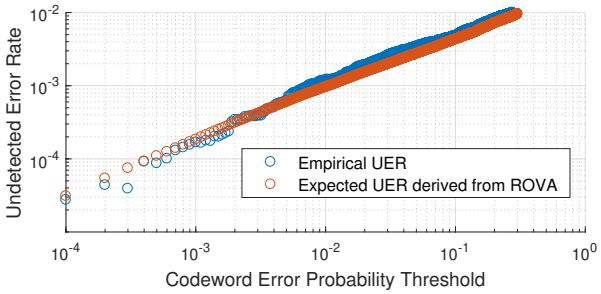

Fig. 1: Graph of empirical and expected undetected codeword error rate (UER) as a function of the ROVA threshold for 100,000 decodings of the 4-state, rate-1/2 convolutional code with $k = 128$ message bits at SNR 4.5 dB.

II. The Reliability Output Viterbi Algorithm

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 paper does not use the logarithmic domain for this reason. There are approximation methods that could be employed for the logarithmic domain as described in [21], but this remains a possible topic for future research.

As described in [12], 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 paper 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:

$$P(\hat{x}^{n_c} = x_t^{n_c} | y^{n_c}) = \frac{P(\hat{x}^{n_c}) f_{Y|X}(y^{n_c}|\hat{x}^{n_c})}{\sum_{x^{n_c} \in \mathcal{C}} P(x^{n_c}) f_{Y|X}(y^{n_c}|x^{n_c})} \quad (1)$$

$$= \frac{f_{Y|X}(y^{n_c}|\hat{x}^{n_c})}{\sum_{x^{n_c} \in \mathcal{C}} f_{Y|X}(y^{n_c}|x^{n_c})} \quad (2)$$

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 (1) to (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 (2) computed as in [12] and consider codewords with a ROVA value below the threshold as erasures because they are not sufficiently reliable. Fig. 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 [22]. 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.

A. Computing ROVA as in [12]

Algorithm 1 describes the procedure for computing the ROVA value (2) as described in [12]. 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 [12]: P_i^m and $\bar{P}_i^m$. We now define these two probabilities.

Algorithm 1: An algorithmic adaptation of the ROVA Algorithm described in [12].

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 (2) in [12] proceeds as follows:

1) $m = m + 1$
2) For each valid branch $(i, j) \in \mathcal{T}_m$ compute metrics
$$\gamma_m(i, j) = f\big(y_m|x_m(i, j)\big) \tag{3}$$
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) 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
$$P_j^m = \Delta_m^{-1}\gamma_m(i^*, j)P_{i^*}^{m-1} \tag{5}$$
$$\bar{P}_j^m = \Delta_m^{-1} \sum_{\{i:(i,j)\in\mathcal{T}_m\}} \gamma_m(i, j)(P_i^{m-1} + \bar{P}_i^{m-1}) - P_j^m \tag{6}$$
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.

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

$$\Delta_m = \frac{\sum_{\mathcal{T}} \gamma_1\gamma_2 \cdots \gamma_m}{\sum_{\mathcal{T}} \gamma_1\gamma_2 \cdots \gamma_{m-1}} \,, \tag{7}$$

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:

$$\sum_{i=0}^{2^v-1} \big(P_i^m + \bar{P}_i^m\big) = 1 \,. \tag{8}$$

Let $N_s = |\mathcal{S}|$ be the number of states, and let N_b be the number of branches entering each state. Table II 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 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 $\bar{P}_{j^*}^m$.

B. Fricke and Hoeher Approximation

Fricke and Hoeher [15] 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 III 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 [15] as compared to the original ROVA

Algorithm 2: An algorithmic adaptation of the Approximate ROVA Algorithm from [15].

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
$$\gamma_m(i,j) = f\big(y_m | x_m(i,j)\big) \tag{9}$$
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
$$\Gamma_m^j = \Gamma_{m-1}^{i^*}\gamma_m(i^*,j)\,, \tag{10}$$
$$\hat{P}_m(i^*,j) = \frac{\Gamma_m^j}{\sum_{(i,j)\in\mathcal{T}_m}\Gamma_{m-1}^i\gamma_m(i,j)}\,. \tag{11}$$
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.

proposed in [12], consider the simple example shown in Fig. 2. Applying *Algorithm 1*, produces the result

$$P_0^4 = \frac{\gamma_1(0,0)\gamma_2(0,0)\gamma_3(0,0)\gamma_4(0,0)}{\sum_{\mathcal{T}}\gamma_1\gamma_2\gamma_3\gamma_4}\,, \tag{12}$$

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 (12) is precisely the probability given in (2) that the codeword has been correctly decoded.

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

$$\prod_{m=1}^{n_c}\hat{P}_m(i^*,j^*) = \frac{\gamma_1(0,0)\gamma_2(0,0)\gamma_3(0,0)\gamma_4(0,0)}{D}\,, \tag{13}$$

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

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

$$\alpha = \frac{\gamma_1(0,1)\gamma_2(1,2)\gamma_3(2,0)}{\gamma_1(0,0)\gamma_2(0,0)\gamma_3(0,0)}\,. \tag{14}$$

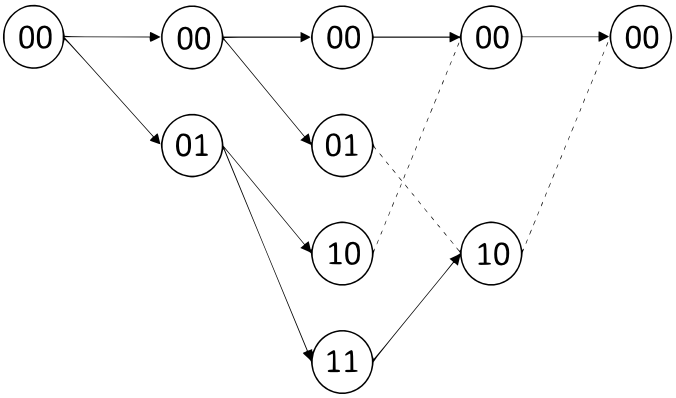

Fig. 2: A four-state trellis that begins at state $s_0 = 0$ and terminates at state $s_4 = 0$ after four symbol transmissions. Solid branches represent survivor paths while dashed branches were rejected by the Viterbi algorithm

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 (11) 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.

III. Accumulated Information Density

Polyanskiy et al. [17] 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

$$i(y_j,\hat{x}_j) = \log_2\frac{f_{Y|X}(y_j|\hat{x}_j)}{f_Y(y_j)}\,. \tag{15}$$

In (15), $f_Y(y_j)$ 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 [23]), each constellation point is equally likely so that $P_X(x) = |\mathcal{X}|^{-1}$.

Following the termination approach of [17], accumulated information density (AID) sums (15) for each symbol in the

codeword to produce $i_{\text{AID}}(y^{n_c}, \hat{x}^{n_c})$ as follows:

$$i_{\text{AID}}(y^{n_c}, \hat{x}^{n_c}) = \sum_{j=1}^{n_c} i(y_j, \hat{x}_j) \tag{16}$$

$$= \sum_{j=1}^{n_c} \log_2 \left(\frac{f_{Y|X}(y_j|\hat{x}_j)}{f_Y(y_j)} \right) \tag{17}$$

$$= \log_2 \left(\frac{\prod_{j=1}^{n_c} f_{Y|X}(y_j|\hat{x}_j)}{\prod_{j=1}^{n_c} f_Y(y_j)} \right) \tag{18}$$

$$= \log_2 \left(\frac{f_{Y|X}(y^{n_c}|\hat{x}^{n_c})}{\sum_{x^{n_c} \in \mathcal{X}^{n_c}} |\mathcal{X}|^{-n} f_{Y|X}(y^{n_c}|x^{n_c})} \right) \tag{19}$$

where $\mathcal{X}^{n_c}$ is the set of all sequences of n_c symbols. For AID, the denominator in (19) 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

$$\Gamma_m^j = \Gamma_{m-1}^{i^*} \gamma_m(i^*, j)\,. \tag{20}$$

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

$$\Pi(m) = \Pi(m-1) f_Y(y_m) \tag{21}$$

5) if $m = n_c$ conclude by reporting

$$i_{\text{AID}}(y^{n_c}, \hat{x}^{n_c}) = \log_2 \frac{\Gamma_n^0}{\Pi(n)}\,, \tag{22}$$

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. 3 and 4 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. 3 and 4, a smaller overlap between the density functions for correctly and incorrectly decoded sequences indicates a better ability for the metric to indicate when a sequence should

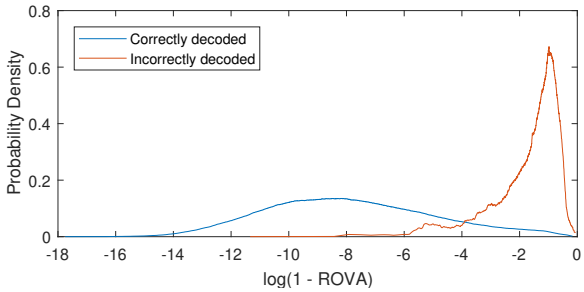

Fig. 3: Probability density of ROVA error values for correct and incorrect decodings for the same conditions as in Fig. 1. The minimal overlap between the incorrect and correct decodings indicates that setting a decision threshold using ROVA is an effective way to reduce undetected errors.

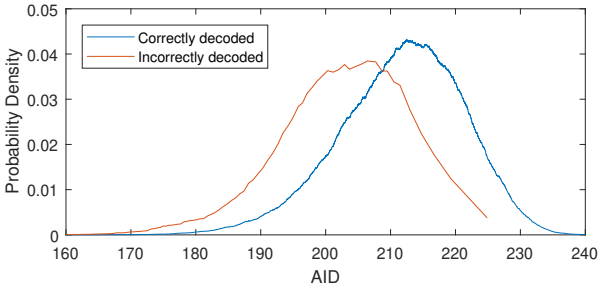

Fig. 4: Probability density of AID values for correct and incorrect decodings for the same scenario as Fig. 1. There is considerable overlap between the incorrect and correct decodings, which suggests that using AID is an ineffective way to reduce undetected errors.

be deemed unreliable. The better separation (smaller overlap area) seen in Fig. 3 as compared to Fig. 4 shows how ROVA is more effective than AID in this example.

Fig. 5 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. 5 confirms the relatively poor performance of AID that

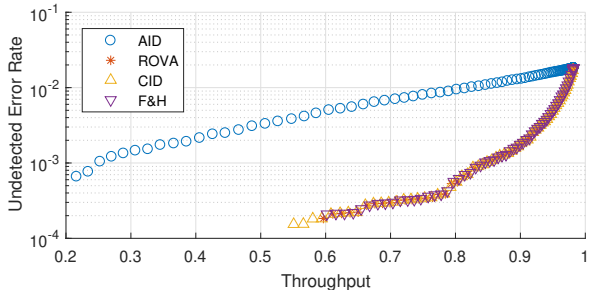

Fig. 5: Undetected error rate (UER) as a function of throughput for ROVA, approximate ROVA, AID, and CID showing the operating points of (throughput, UER) achievable with thresholds on ROVA, approximate ROVA AID, and CID metrics for the same scenario as Fig. 1. ROVA and CID give identical performance as expected by (32). Approximate ROVA has near identical performance to ROVA and CID.

was suggested in Figs. 3 and 4 . 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. 3 and 4, 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 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. 6 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

$$||g|| = \sqrt{(g_1)^2 + ... + (g_N)^2} = \sqrt{\sum_{k=1}^{N} (g_k)^2} \tag{23}$$

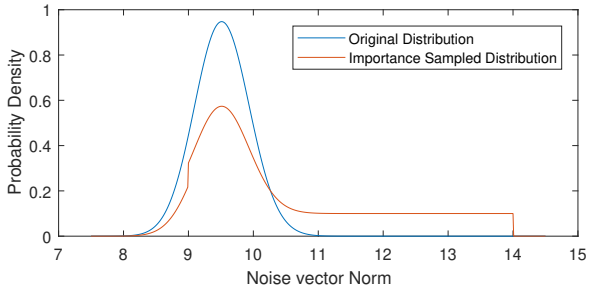

Fig. 6: The original Nakagami distribution which describes the distribution of the noise vector norms compared to the bias distribution for the same scenario as Fig. 1. The bias distribution is a mixture distribution composed of one half of the original Nakagami distribution with one half of a uniform distribution on the interval $[9, 14]$.

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

$$g_i \sim \mathcal{N}_i(0, \sigma_i^2) \rightarrow \frac{g_i}{\sigma_i} \sim \mathcal{N}_i(0, 1) \tag{24}$$

$$\left(\frac{g_i}{\sigma_i}\right)^2 \sim (\mathcal{N}_i(0, 1))^2 = \mathcal{X}^2(1) \equiv \Gamma\left(\frac{1}{2}, 2\right) \tag{25}$$

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.

$$g_i^2 \sim \sigma_i^2 \Gamma\left(\frac{1}{2}, 2\right) = \Gamma\left(\frac{1}{2}, 2\sigma_i^2\right) \tag{26}$$

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

$$\Gamma\left(\frac{1}{2}, 2\sigma_i^2\right) = \Gamma\left(\frac{1}{2}, 2\sigma^2\right) \tag{27}$$

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.

IV. Codeword Information Density: ROVA Redux

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:

$$i_{\text{CID}}(y^{n_c}, \hat{x}^{n_c}) = \log_2 \frac{f_{Y|X}(y^{n_c}|\hat{x}^{n_c})}{\sum_{x^{n_c} \in \mathcal{C}} P(x^{n_c}) f_{Y|X}(y^{n_c}|x^{n_c})} \tag{28}$$

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 (2) and (28), we find that ROVA and CID have almost the same formula. Starting with (2), including a $P(x^{n_c})$ term in the denominator and taking a logarithm produces (28). Consequently, CID and ROVA have a one-to-one transformation given by

$$i_{\text{CID}}(y^{n_c}, \hat{x}^{n_c}) = \log_2 \left(\frac{P_0^{n_c}}{P(x^{n_c})}\right). \tag{32}$$

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

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

$$Z_m^j = \sum_{(i,j)\in\mathcal{T}_m} Z_{m-1}^i \gamma_m(i, j)\,. \tag{29}$$

5) if $m = n_c$ conclude by reporting either

$$i_{\text{CID}}(y^{n_c}, \hat{x}^{n_c}) = \log_2\left(\frac{\Gamma_n^0}{P(x^{n_c})Z_n^0}\right), \text{or} \tag{30}$$

$$P_0^{n_c} = \frac{\Gamma_n^0}{Z_n^0} \tag{31}$$

otherwise, go to Step 1.

code where $N_b = 2$, excluding the number of multiplications necessary 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 $\gamma_m(i, j)$.

V. Analytical Expression for the $P_0^{n_c}$ Distribution

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. 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 (2) as a random variable F and recall that for an AWGN channel it is computed as

$$F = \prod_{i=1}^{n_b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \hat{x}_i)^2}{2\sigma^2}} \tag{33}$$

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 $d_H(\hat{x}^{n_c}, x_t^{n_c}) = 0$,

$$F_0 = \prod_{i=1}^{n_b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z_i^2}{2\sigma^2}} \tag{34}$$

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

$$F_1 = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z_j+2)^2}{2\sigma^2}} \prod_{i=1, i\neq j}^{n_b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z_i^2}{2\sigma^2}} \tag{35}$$

$$= e^{-\frac{4+4z_j}{2\sigma^2}} \prod_{i=1}^{n_b} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z_i^2}{2\sigma^2}} \tag{36}$$

$$= e^{-\frac{4+4z_j}{2\sigma^2}} F_0 \tag{37}$$

where the mean of Gaussian describing the j^{th} symbol in (35) 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 $d_H(\hat{x}^{n_c}, x_t^{n_c}) = m$,

$$F_m = e^{-\frac{4m+\sum_{\ell=1}^{m} 4z_\ell}{2\sigma^2}} F_0 \tag{38}$$

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:

$$A_u = \left|\{\hat{x}^{n_c} \in \mathcal{T} : d_H(\hat{x}^{n_c}, x_0^{n_c}) = u\}\right|, \tag{39}$$

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

$$P_0^{n_c} \approx \frac{F_0}{F_0 \sum_{u=0}^{U} A_u e^{-\frac{4u+\sum_{\ell=1}^{u} 4z_\ell}{2\sigma^2}}}, \tag{40}$$

which can also be expressed as

$$P_0^{n_c} \approx \left(1 + \sum_{u=1}^{U} A_u e^{-\frac{4u+\sum_{\ell=1}^{k} 4z_\ell}{2\sigma^2}}\right)^{-1}. \tag{41}$$

The expression for $P_0^{n_c}$ given in (41) 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 [24] as well as Table 12.1 of [25], Fig. 7 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 (41) 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. 7.

For the following analysis, $P(\text{C})$ is the probability of Viterbi selecting the correct codeword a.k.a. the throughput, $P(\text{E})$ is the probability of Viterbi selecting an incorrect codeword, i.e.,

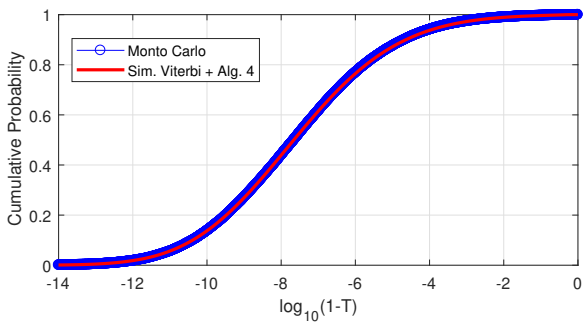

Fig. 7: Cumulative histogram of ROVA metric computed by simulation of Viterbi/Algorithm 4 and by Monte Carlo of (41) with U truncated to 21 for 64-state, rate-1/3 convolutional code with $n = 32$ at SNR 1.0 dB.

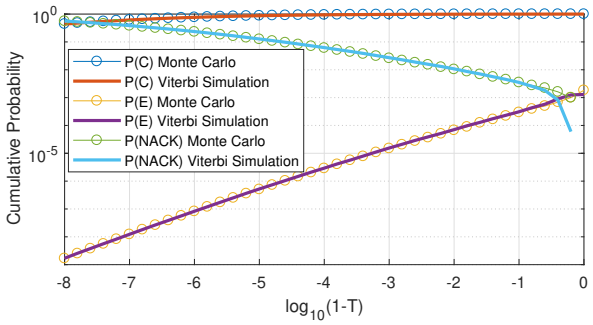

Fig. 8: Comparison of throughput $P(\mathrm{C})$, $P(\mathrm{E})$, and $P(\mathrm{NACK})$ between ROVA probabilities obtained by simulation for the code and channel of Fig. 7 and Monte Carlo using (41) with $U = 21$.

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 (41) 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:

$$P(\mathrm{C}) = \int_{p=T}^{1} p f_P(p) dp \tag{42}$$

$$P(\mathrm{E}) = \int_{p=T}^{1} (1-p) f_P(p) dp \tag{43}$$

$$P(\mathrm{NACK}) = \int_{p=0}^{T} f_P(p) dp\,. \tag{44}$$

Fig. 8 compares the application of (42), (43), and (44) using the cumulative histogram of $P_0^{n_c}$ generated by Monte Carlo using $U = 21$ (shown in Fig. 7) to the values of $P(\mathrm{C})$, $P(\mathrm{E})$, and $P(\mathrm{NACK})$ 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 $P(\mathrm{NACK})$ for values of $\log(1-T)$ above -0.5. As shown in Fig. 3 (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.

VI. Complexity Analysis of Reliability Metrics

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

$$\sum_{i=1}^{\log_{N_b}(N_s)} (N_b)^i = \frac{N_b^{\log_{N_b}(N_s)+1} - N_b}{N_b - 1} \tag{45}$$

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 (45).

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

$$N_b N_s (k - 2\log_{N_b}(N_s)) + \frac{2(N_b^{\log_{N_b}(N_s)+1} - N_b)}{N_b - 1} \tag{46}$$

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 (46) 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:

$$\gamma_m = \prod_{i=1}^{n_d} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \hat{x}_i)^2}{2\sigma^2}} \tag{47}$$

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 trellis stage $\mathcal{T}_m$, or along the surviving branches chosen by the Viterbi algorithm. For the Raghavam and Baum implementation of ROVA [12], the computation of Δ_m in (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 (5)-(6) only needs to be performed along the surviving branches. The Frick and Hoeher Approximation of ROVA [15] requires the computation of Γ_m^j in (10) along every valid branch as those values are necessary for the computation of the denominator in (11) for the winning branches. The AID algorithm only requires the computation of Γ_m^j in (20) on the surviving branches chosen by Viterbi, and $\Pi(m)$ in (21) can be computed once for every stage. The CID implementation of ROVA requires the computation of Z_m^j in (29) for each state and Γ_m^j for each surviving path.

Table I shows the number of operations per stage necessary to compute each reliability metric using the most efficient implementation we could devise. Fig. 9 illustrates how the additional computation required for each reliability metric compares with the computations required for Viterbi decoding

TABLE I: 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

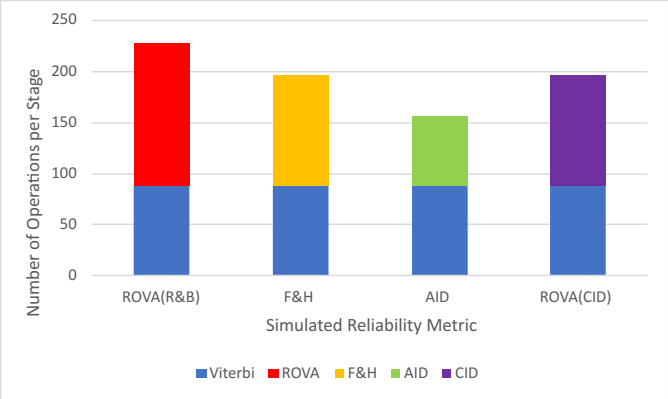

Fig. 9: Number of operations needed per stage to implement Viterbi alongside the chosen reliability metric. Number of operations is based on a 4 state, 128 information bit, rate 1/2 decoder.

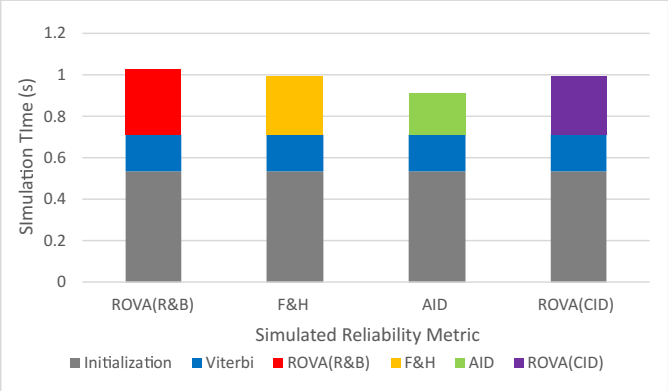

Fig. 10: Simulation time per 100 decodings with Viterbi and the chosen reliability metric. Initialization includes everything that was simulated that was not a part of the Viterbi algorithm or the chosen reliability metric. Initialization includes tasks such as creation of the trellis structure, encoding the input sequence, and computing every state transition bit sequence in the trellis.

for the example of $\alpha = 2$, $N_s = 4$, $N_b = 2$, and $n_d = 2$. Fig. 10 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 (47) 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 (47) once per stage can significantly reduce the computation. For each unique symbol value in each dimension of the mapped symbol alphabet, the term

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \hat{x}_i)^2}{2\sigma^2}} \tag{48}$$

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 $7n_d N_s N_b$ computations per stage to $n_d N_s N_b + 6n_d\alpha$ 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, (48) can be computed once for each of these five values, and the results can be cached and accessed when they are multiplied together in (47). 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 [12] has an additional $3N_bN_s$ and $4N_s$ term necessary to compute (4)-(6) of Algorithm 1. The ROVA approximation [15] and ROVA computed as CID have an additional N_bN_s term necessary to compute (11) and (29) respectively. After these optimizations, AID [17] is the fastest metric, ROVA computed as CID and the ROVA approximation share the same complexity, and ROVA computed as in [12] 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.

VII. 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. VIII.

Let $\varepsilon_{x,x'}$ 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

$$\text{FER} = P\left(\bigcup_{\substack{x,x'\in\mathcal{C}\\ x\neq x'}} \varepsilon_{x,x'}\right) = P\left(\bigcup_{\substack{x,x'\in\mathcal{C}\\ x\neq x'}} \mathbf{e}_{x,x'}\right) \tag{49}$$

$$\leq \sum_{\substack{x,x'\in\mathcal{C}\\ x\neq x'}} P\left(\mathbf{e}_{x,x'}\right). \tag{50}$$

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

$$P\left(\mathbf{e}_{x,x'}\right) = Q\left(\sqrt{\frac{d^2\left(x,x'\right)}{2N_0}}\right) p(x) \tag{51}$$

$$\leq Q\left(\sqrt{\frac{d_{free}^2}{2N_0}}\right) e^{\frac{d_{\text{free}}^2}{4N_0}} e^{-\frac{d^2(x,x')}{4N_0}} p(x), \tag{52}$$

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

$$p(x) = \prod_{\ell=1}^{n^c} q(x_\ell) \tag{53}$$

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

Let

$$W = e^{-\frac{1}{4N_0}}. \tag{54}$$

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

$$\sum_{\substack{x,x'\in\mathcal{C}\\ x\neq x'}} \exp\left(-\frac{d^2\left(x,x'\right)}{4N_0}\right) p(x) \tag{55}$$

$$= \sum_{\substack{x,x'\in\mathcal{C}\\ x\neq x'}} \exp\left(-\frac{1}{4N_0}\sum_{\ell=1}^{n^c} d^2\left(x_\ell,x'_\ell\right)\right) p(x) \tag{56}$$

$$= \sum_{\substack{x,x'\in\mathcal{C}\\ x\neq x'}} \prod_{\ell=1}^{n^c} \left[W^{d^2\left(x_\ell,x'_\ell\right)} q\left(x_\ell\right)\right] \tag{57}$$

$$= \sum_{x,x'\in\mathcal{C}} \prod_{\ell=1}^{n^c} \left[W^{d^2\left(x_\ell,x'_\ell\right)} q\left(x_\ell\right)\right] - \sum_{x} \prod_{\ell=1}^{n^c} \left[W^0 q\left(x_\ell\right)\right] \tag{58}$$

$$= \sum_{x,x'\in\mathcal{C}} \prod_{\ell=1}^{n^c} \left[W^{d^2\left(x_\ell,x'_\ell\right)} q\left(x_\ell\right)\right] - 1, \tag{59}$$

We convert the sum over codewords into a sum over paths through the trellis. In order to compute (59), 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 [26] 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

$$\begin{bmatrix} \mathbf{d} & \mathbf{c} \\ \mathbf{b} & \mathbf{A} \end{bmatrix} \tag{60}$$

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 [26]. We can rewrite (59) as the following transfer function

$$T(W) = \begin{bmatrix} \mathbf{d} & \mathbf{c} \end{bmatrix} \begin{bmatrix} \mathbf{d} & \mathbf{c} \\ \mathbf{b} & \mathbf{A} \end{bmatrix}^{n^c-2} \begin{bmatrix} \mathbf{d} \\ \mathbf{b} \end{bmatrix} - 1, \tag{61}$$

Noting that the Hamming weight does not appear in (59), we can combine (52) and (61) as the bound

$$\text{FER} \leq Q\left(\sqrt{\frac{d_{free}^2}{2N_0}}\right) e^{\frac{d_{free}^2}{4N_0}} T\left(W = e^{-\frac{1}{4N_0}}\right), \tag{62}$$

where d_{free} is the free distance of the code. For punctured trellis codes, [27] states that the transfer function in (62)

should not be evaluated with a single W. Each matrix in (61) 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, (62) becomes the following:

$$\text{FER} \leq Q\left(\sqrt{\frac{r^2}{2N_0}}\right) e^{\frac{r^2}{4N_0}} T\left(W^{\ell}\right), \tag{63}$$

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

$$T(W^{\ell}) = \begin{bmatrix}\mathbf{d} & \mathbf{c}\end{bmatrix}\Big|_{W_1} \prod_{j=2}^{n^c-1}\left(\begin{bmatrix}\mathbf{d} & \mathbf{c}\\ \mathbf{b} & \mathbf{A}\end{bmatrix}\Big|_{W_j}\right)\begin{bmatrix}\mathbf{d}\\ \mathbf{b}\end{bmatrix}\Big|_{W_\ell} - 1 \tag{64}$$

A greedy search algorithm detailed in Algorithm 5 is performed in Section VIII 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 (63) 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.

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

Initialization: Compute the minimal state transition diagram as in eq. (60). 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 (63) 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.

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

Previous work by Williamson [13] 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.

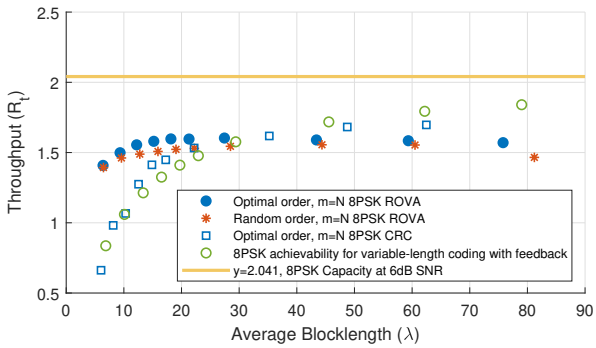

Fig. 11: Short-blocklength performance of the m = N ROVA-based retransmission scheme over the AWGN channel with SNR 6.00 dB and target probability of error $\epsilon = 10^{-3}$ using a 64 state rate 1/3 code with 8-PSK modulation. The ROVA for terminated convolutional codes is used. The performance of 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.

Fig. 11 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 [28] 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 [29] such that each data point in Fig. 11 achieves a frame error rate less than target ϵ. Fig. 12 shows the observed frame error rates for each of the simulated data points in Fig. 11. It can be seen that ROVA is able to target specific error rates much more precisely than the CRC. Both schemes utilize the techniques in Section VII 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. 11. 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:

$$\lambda \leq \frac{1 + \sum_{i=1}^{N-1} P_{\text{NACK}}(i)}{1 - P_{\text{NACK}}(N)} \tag{65}$$

$$R_t = \frac{k}{\lambda}(1 - P_{\text{UE}}) \tag{66}$$

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

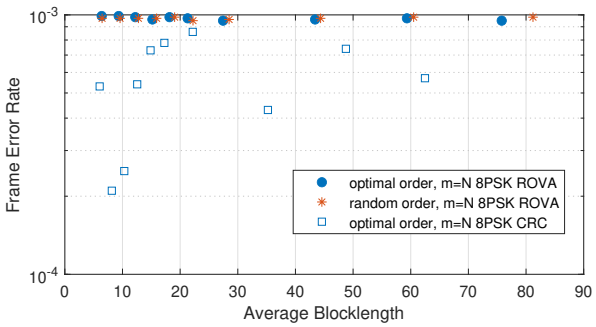

Fig. 12: Frame error rate plotted against the average blocklength for each of the simulated points in Fig. 11. Each ROVA threshold and CRC polynomial is chosen such that the frame error rate is below target error rate ϵ. ROVA is able to target desired error rates more precisely than the CRC.

coding with feedback represents the random coding lower bound as defined in [13] according to [17]. Similar to the results shown in [13], the simulations in Fig. 11 demonstrate that the throughput of this convolutional code exceeds the random-coding lower bound at short blocklengths. Fig. 11 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 paper can aid in determining if the decoding is sufficiently reliable.

IX. Conclusion

This paper compares the ROVA algorithm of [12] to AID from [17] and found that AID is less accurate 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 [12]. The Fricke and Hoeher approximation was shown to have comparable accuracy to ROVA, but the CID implementation of ROVA achieves similar complexity reduction as Fricke and Hoeher while computing the exact probability rather than an approximation.

This paper also derives an expression for the random variable that describes the codeword reliability according to ROVA. The distribution of the ROVA metric can be used to accurately model how the probabilities of correct decoding, undetected error, and negative acknowledgement depend on the choice of ROVA threshold.

Additionally, this paper derives an expression for the union bound on the frame error rate for ZTCCs with punctured symbols. An optimal ordering of additional symbols to transmit in an incremental retransmission scheme is achieved by utilizing this union bound in a greedy search algorithm.

Finally, this paper shows that ROVA can be used in a reliability-based retransmission scheme using 8PSK modulation to achieve higher throughput than the achievability lower bound of [17] at short average blocklengths. Based on the data in Fig. 11, ROVA allows us to significantly exceed Polyanskiy's random coding union bound for very short blocklengths. However, the CRC performs better than ROVA after a certain average blocklength. The reason for this is that the CRC provides an additional coding benefit, as well as the fact that the penalty of the additional bits required by the CRC decreases with increased blocklength.

References

[1] A. Baldauf, A. Belhouchat, N. Wong, and R. D. Wesel, "Efficient computation of convolutional decoder reliability without a crc," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 508–512.

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

[3] 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.

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

[5] *"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.

[6] European Telecommunications Standards Institute 3GPP TS 25.212 version 7.0.0 release 7. [Online]. Available: https://portal.3gpp.org

[7] 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.

[8] D. J. Costello, J. Hagenauer, H. Imai, and S. B. Wicker, "Applications of error control coding," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2531–2560, Oct. 1998.

[9] 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.

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

[11] H. Yamamoto and K. Itoh, "Viterbi decoding algorithm for convolutional codes with repeat request," *IEEE Transactions on Information Theory*, vol. 26, no. 5, pp. 540–547, 1980.

[12] A. Raghavan and C. Baum, "A reliability output Viterbi algorithm with applications to hybrid ARQ," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 1214–1216, May 1998.

[13] A. R. Williamson, T.-Y. Chen, and R. D. Wesel, "Variable-length convolutional coding for short blocklengths with decision feedback," *IEEE Trans. on Comm.*, vol. 63, no. 7, pp. 2389–2403, Jul 2015.

[14] J. Fricke and P. Hoeher, "Reliability-based retransmission criteria for hybrid ARQ," *IEEE Transactions on Communications*, vol. 57, no. 8, pp. 2181–2184, Aug 2009.

[15] J. C. Fricke and P. A. Hoeher, "Word error probability estimation by means of a modified Viterbi decoder," in *2007 IEEE 66th Vehicular Technology Conference*, Sep 2007.

[16] A. R. Williamson, M. J. Marshall, and R. D. Wesel, "Reliability-output decoding of tail-biting convolutional codes," *IEEE Transactions on Communications*, vol. 62, no. 6, pp. 1768–1778, Jun 2014.

[17] Y. Polyanskiy, H. V. Poor, and S. Verdu, "Feedback in the non-asymptotic regime," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 4903 – 4925, August 2011.

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

[19] J. Hagenauer, "Rate-compatible punctured convolutional codes (rcpc codes) and their applications," *IEEE Transactions on Communications*, vol. 36, no. 4, p. 389–400, 1988.

[20] A. Bernard, X. Liu, R. Wesel, and A. Alwan, "Speech transmission using rate-compatible trellis codes and embedded source coding," *IEEE Transactions on Communications*, vol. 50, no. 2, p. 309–320, 2002.
[21] 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*, vol. 20, no. 2, p. 284–287, 1974.
[22] W. E. Ryan and S. Lin, *Channel codes: classical and modern*. Cambridge University Press, 2009, p. 148.
[23] T. Wiegart, F. Da Ros, M. P. Yankov, F. Steiner, S. Gaiarin, and R. D. Wesel, "Probabilistically shaped 4-pam for short-reach im/dd links with a peak power constraint," *Journal of Lightwave Technology*, vol. 39, no. 2, pp. 400–405, 2021.
[24] A. R. Williamson, T.-Y. Chen, and R. D. Wesel, "Variable-length convolutional coding for short blocklengths with decision feedback," *IEEE Transactions on Communications*, vol. 63, no. 7, p. 2395, 2015.
[25] S. J. Lin and D. J. Costello, *Error control coding: fundamentals and applications*, 2nd ed. Prentice-Hall, 2004.
[26] R. D. Wesel, "Reduced-state representations for trellis codes using constellation symmetry," *IEEE Transactions on Communications*, vol. 52, no. 8, p. 1302–1310, Aug 2004.
[27] R. D. Wesel and X. Liu, *Analytic Techniques for Periodic Trellis Codes**. University of Illinois Urbana-Champaign, 1998.
[28] R. Wesel, X. Liu, and W. Shi, "Trellis codes for periodic erasures," *IEEE Transactions on Communications*, vol. 48, no. 6, pp. 938–947, 2000.
[29] P. Koopman and T. Chakravarty, "Cyclic redundancy code (crc) polynomial selection for embedded networks," *International Conference on Dependable Systems and Networks, 2004*, 2004.

APPENDIX

TABLE II: 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 III: 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 (11) N_s divisions to compute (11) for each state
Step 4	n_c multiplies to compute $\prod_{m=1}^{n_c} \tilde{P}_m(i^*, j^*)$.

TABLE IV: 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 V: 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 VI: Implementation details of the comparison between incremental retransmission schemes utilizing either ROVA or the CRC in Fig. 11. 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