Improved Decoding of Convolutional and Turbo Codes via List Decoding

Ethan Liang

Stanford University, Stanford, CA 94305 USA

I. Introduction

The third generation partnership project (3GPP) defined three service categories for the fifth-generation (5G) standard [1]. Each service category defines requirements on bandwidth, reliability, latency, and complexity to facilitate the development of future applications not yet feasible using current wireless standards. The already adopted enhanced mobile broadband (eMBB) standard aims to provide higher data rates than previous generations. The other two standards place less emphasis on throughput and emphasize reliability, latency and low complexity. The massive machine-type communication (mMTC) standard promises to support communication between large numbers of ultra-low-power devices whereas the ultra-reliable and low latency communication (URLLC) standard aims to support highly delay-sensitive applications. While each of these standards will require some form of forward error correction, mMTC and URLLC have the additional requirement of low complexity and low latency in order to achieve their respective service category requirements. Convolutional codes and turbo codes under list decoding may be capable of providing the low complexity and powerful error correction required by these standards.

Convolutional codes (CC) have had a profound impact in many communication standards, including 3rd generation cellular standards and deep-space communications. In addition, tail-biting convolutional codes (TBCCs) [2] and turbo codes [3] were used for forward error correction in the long-term evolution (LTE) standard.

Decoding methods for CCs, likewise, have a long, storied history. Andrew Viterbi developed a practical maximum-likelihood (ML) decoding algorithm for CCs based on the trellis representation for cyclic codes [4]. Bahl *et al.* developed a bit-wise maximum a posteriori (MAP) decoding algorithm for convolutional codes which later played an important role in turbo codes [5]. Seshadri *et al.* developed a method for performing CRC-aided decoding of CCs using an outer cyclic redundancy check (CRC) code [6]. Further, Lou *et al.* introduced an algorithm that maximizes the distance spectrum of a concatenated CC-CRC code where the encoder for the CC is in a feedforward implementation [7].

Methods for decoding TBCCs and turbo codes have also developed over a long period of time. The theory of TBCCs was further developed in [2] and near ML decoding algorithms for TBCCs were introduced in [8], [9]. Maximum likelihood decoding of a concatenated inner TBCC with an outer CRC code was further explored in [10], [11]. Turbo codes [3] are often considered the first, low-complexity, capacity approaching codes. Serial-list Viterbi decoding of concatenated turbo codes with CRCs was first introduced in [12] and further studied in [13]–[17].

A. Project Outline

The goal of this project is to begin the development of the software and the analysis required to design, decode, and simulate the performance of CCs, TBCCs, and turbo codes concatenated with CRCs decoded using serial-list Viterbi decoding [6]. A brief recapitulation of the development schedule for the code base is given in the following:

- Prior to the Feb. 7th proposal deadline, I implemented an encoder for a rate-$1/n$ systematic feedback (SFB) and non-systematic feedforward (NSFF) CC. On the decoder side, I implemented a distance-based soft decoder using the Viterbi algorithm and the wrap-around Viterbi algorithm [9].
- Over the next two weeks from Feb. 8th to Feb. 22nd, I implemented a generalized list-decoding algorithm for trellis-based decoders following the work in [6], [11]. The decoding algorithm was used in decoding zero-terminated convolutional codes (ZTCC) with both SFB and NSFF trellis structures. I also implemented a new decoding algorithm for TBCCs in the feed-forward configuration.
- In the two weeks from Feb. 23rd to Mar. 8th, I implemented a bit-wise MAP decoder based on the BCJR algorithm. The implementation of the code is valid for CCs in both the SFB and NSFF structures and for both terminated and "open-ended" trellis structures.

Upon the completion of each of the three implementation phases above, I performed simulations leading to a collection of observations and results that are analyzed later in this report. These results also lend some insight into the design of concatenated turbo codes with a CRCs that would lead to improved decoding performance.

Ultimately, this project aims to use the techniques described among the papers cited in the introduction to design concatenated TBCCs with CRCs and concatenated turbo codes with CRCs which, under serial-list Viterbi decoding, exhibit performance near the finite-blocklength coding bounds in [18]. Although the decoder for concatenated turbo codes with

CRCs is not yet completed, the results presented in this paper show that the development of the constituent elements of the decoder has been completed.

II. Preliminaries

This section will provide an introduction to the notation used throughout the paper and will restate important results in a way that is amenable to derivations in later sections. To balance brevity with accessibility, relevant major results will be stated without proof and will refer the reader to [19] for further discussion.

A. Convolutional codes and the Viterbi algorithm

Let $f(x)$ denote a k bit information sequence and let $p(x)$ denote an $m+1$ bit CRC polynomial of degree m. Let $r(x)$ define the remainder after message polynomial $x^m f(x)$ is divided by $p(x)$. $m(x)$ is the resulting $k+m$ bit sequence which is the input to the convolutional encoder. All polynomials are assumed to be big-endian, i.e., $f(x) = f_{k-1}x^{k-1} + f_{k-2}x^{k-2} + \cdots + f_1 x + f_0$, and all operations are over binary field $\mathbb{F}_2$.

The convolutional encoding function can be represented as a polynomial product of $m(x)$ with vector polynomial $\boldsymbol{g(x)}$, which is assumed to be rate-$1/n$. Each constituent polynomial of $\boldsymbol{g(x)}$ is assumed to have degree v. An alternative representation for the encoding function for convolutional codes is via a shift-register with v elements. ZTCCs refer to convolutional codes in which, during the encoding with $\boldsymbol{g(x)}$, input a trailing v zeros into the shift-register to return the encoder to the all zero state. TBCCs refer to convolutional codes in which the shift-register is instantiated with the v highest degree bits from $m(x)$. Two different rate-$1/2$ CC generator polynomials are used throughout this paper; the $v = 3$ CC generator polynomial is (0x13, 0x17) and the $v = 6$ generator polynomial is (0x133, 0x171).

Because of the termination, all valid codewords are represented by paths through the trellis that start and end in the zero-state. For TBCCs, all codewords are the standard trellis are uniquely represented by paths through the trellis that start and end in the same state, but not necessarily the all-zero state. The resulting length of the sequence N from encoding is $n(k+m+v)$ and $n(k+m)$ for ZTCCs and TBCCs, respectively. Although including the termination bits simplifies the decoding process, the rate loss due to termination can be prohibitive at the short blocklengths considered in this paper.

The binary input additive-white Gaussian noise (AWGN) channel with one-dimensional noise variance $\frac{N_0 B}{2}$ is used throughout this paper assuming QPSK modulation. Each dimension of the transmitted symbol is assumed to be in $\{+1, -1\}$ and the one-dimensional noise variance is represented by i.i.d. Gaussian random variables with variance $\sigma^2 = \frac{N_0 B}{2}$ resulting in a signal-to-noise power ratio (SNR) of $\frac{1}{\sigma^2}$. Let x be a sequence that denotes the length N modulated signal over the channel assuming the mapping that 0 is mapped to $+1$ and 1 is mapped to -1 and y denote a sequence which is the length N received signal after the AWGN is added.

The Viterbi algorithm is implemented using the standard add-compare-select operations on the natural trellis representation of the convolutional code. Cumulative path metric differences are stored at each trellis state along with the path metric difference between the two incoming paths. An important property of the Viterbi algorithm is that it is a maximum-likelihood sequence detection algorithm, i.e., it finds the sequence that solves the following optimization problem:

$$\hat{c}(x) = \operatorname{argmin}_{c(x)} d_E(y, x),$$

where $d_E(y, x)$ is the Euclidean distance between the received vector y and possible x and x is the transmitted symbol corresponding to $\hat{c}(x)$. Equivalently, the optimization problem can be formulated as

$$\begin{aligned}
\hat{c}(x) &= \operatorname{argmax}_{c(x)} f(y|x) \\
&= \operatorname{argmax}_{c(x)} \prod_{i=1}^{N} f(y_i|x_i) \\
&= \operatorname{argmax}_{c(x)} \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\|y_i - x_i\|^2}{2\sigma^2}} \\
&= \operatorname{argmax}_{c(x)} \prod_{i=1}^{N} e^{-\frac{\|y_i - x_i\|^2}{2\sigma^2}} \\
&= \operatorname{argmax}_{c(x)} \sum_{i=1}^{N} -\frac{\|y_i - x_i\|^2}{2\sigma^2} \\
&= \operatorname{argmin}_{c(x)} \sum_{i=1}^{N} \|y_i - x_i\|^2 .
\end{aligned}$$

B. The serial-list Viterbi algorithm, bit-wise MAP decoding of convolutional codes, and the wrap-around Viterbi algorithm

In [6], the authors made the observation that a list of codeword candidates in order of increasing metric path metric (or decreasing likelihood) could be found serially using the serial-list Viterbi algorithm (S-LVA). S-LVA operates under the principle that the 2nd most likely path must be a path that detours from the maximum-likelihood path computed using the Viterbi algorithm exactly once. The third most likely codeword must be a path that detours from any possible valid state among the first or second paths. In general, the l^{th} most likely path can be computed by considering all $l-1$ candidate paths previously computed by the S-LVA. The outer CRC is used to check whether a decoded codeword is a valid codeword of the concatenated code. This process continues until a pre-determined maximum list size L_{max} is reached. L is the random variable whose value is the total number of candidate codewords checked against the CRC for one S-LVA decoding.

There are three possible categories that the output of the S-LVA can fall under: a successful decoding, an erasure,

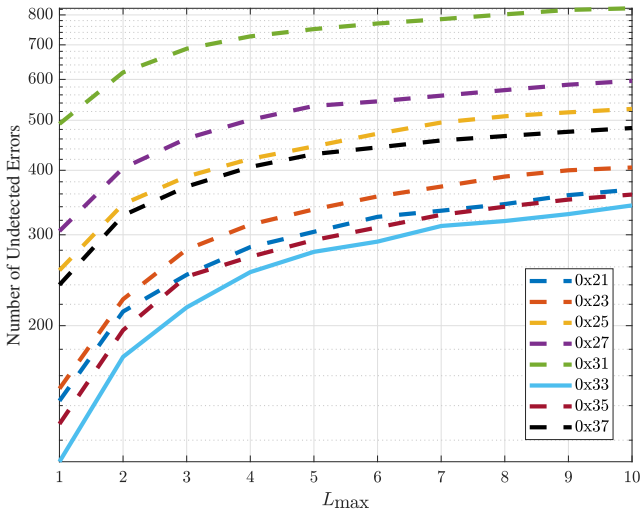

Fig. 1. Shown above is the number of undetected errors vs. the maximum list size under S-LVA decoding for a concatenated ZTCC with all possible CRCs of degree $m = 4$. The solid line indicates the CRC chosen by the techniques outlined in [7]. Other simulation parameters are $v = 6$, $k = 64$, SNR $= 2$ dB, and the total number of simulations is 10^5 per CRC. The numbers in the legend are represented in octal form from MSB to LSB. For example, 0x35 corresponds to the polynomial $x^4 + x^3 + x^2 + 1$.

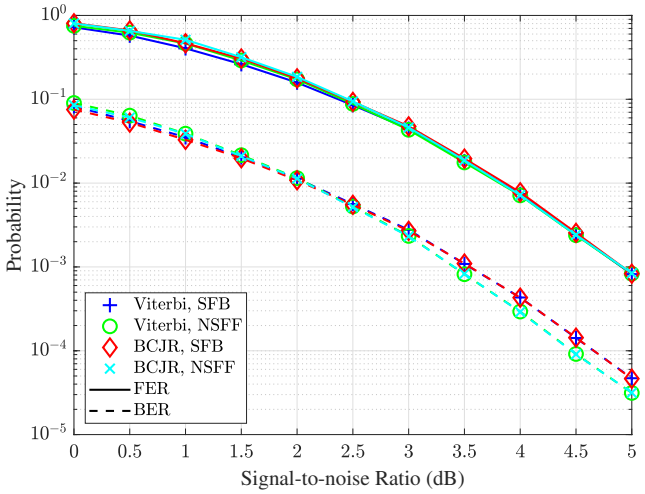

Fig. 2. Performance comparison between the Viterbi algorithm and the BCJR algorithm using full floating-point precision. The convolutional code used is a ZTCC and the simulation parameters are $k = 64$, $v = 3$, $m = 0$, and $L_{\max} = \infty$. The noise sequence was generated using a monotonically increasing random seed to fix ensure both decoders received the same noise vectors.

or an undetected error. If S-LVA finds a possible codeword before the maximum list size is reached, then the decoding is successful. If L equals $L_{\max}$ and a codeword that passes the CRC has not been identified, an erasure is declared by the decoder. An undetected error occurs when S-LVA outputs a codeword that passes the CRC that does not equal the originally transmitted codeword. Fig. 1 shows that choosing the CRC that maximizes the distance spectrum of the concatenated CC/CRC code minimizes the undetected error probability for the specified simulation parameters. In [11], the authors showed that at the additional complexity of S-LVA over the standard soft Viterbi algorithm can be small.

The Bahl Cocke Jelinek Raviv (BCJR) algorithm is an alternative method of decoding a convolutional code and serves as an important sub-component of turbo codes. The BCJR algorithm is a soft output, bit-wise MAP decoder which solves the following optimization problem:

$$\hat{m}_i = \text{argmax}_{m_i} P(m_i|y),$$

where m_i is the i^{th} message bit and $P(m_i|y)$ is the posterior probability of m_i given the received vector y. The logarithmic a posteriori probability (log-APP) is defined to be

$$L(m_i) = \log \frac{P(m_i = 0|y)}{P(m_i = 1|y)}.$$

To partially verify the correctness of my implementation of the BCJR algorithm, I simulated the two algorithms for both SFB and NSFF ZTCCs. Fig. 2 shows the decoding performance difference between the two Viterbi and BCJR algorithms as measure by frame error rate (FER) and bit error rate (BER). Although the FER and BER rate curves appear indistinguishable for the SFB and NSFF implementations of the convolutional encoder, the exact simulated tables show that the Viterbi algorithm results has a lower FER and the BCJR algorithm has a lower BER as expected. The most prominent difference between the two decoders is the BER difference between the SFB and NSFF encoder structure for the algorithm. This difference arises from the fact that the bit-to-codeword mapping is more randomized for the SFB CC, so that nearby codewords (by hamming distance) differ in more message bits as compared to NSFF CCs on average.

III. List Decoding of Tail-biting Convolutional Codes using the Wrap-around Viterbi Algorithm

In [11], the authors extended S-LVA to perform maximum likelihood decoding of TBCCs. For the sake of brevity, the interested reader can refer to the paper for further information about the decoding algorithm. The wrap-around Viterbi algorithm (WAVA) was introduced in [9] as a near-maximum likelihood decoding algorithm for TBCCs. Although many variations of WAVA exist, I will give a high-level overview of my specific implementation.

WAVA performs the standard add-compare-select operations along a trellis with all starting states initialized to zero. When the decoder reaches the last state in the trellis, the ending state with the lowest cumulative metric is found and checked for the tail-biting condition, i.e., the decoder checks if the winning path that ends at the lowest metric end state has a starting state equal to itself. If this condition is met, the decoder declares the message corresponding to the winning path as the decoded message. If the lowest-metric ending state does not meet this condition, the initial states of the trellis are initialized with the cumulative-metric values of the final state and the add-compare-select operations repeat again for the entire length of the trellis. This process repeats until a

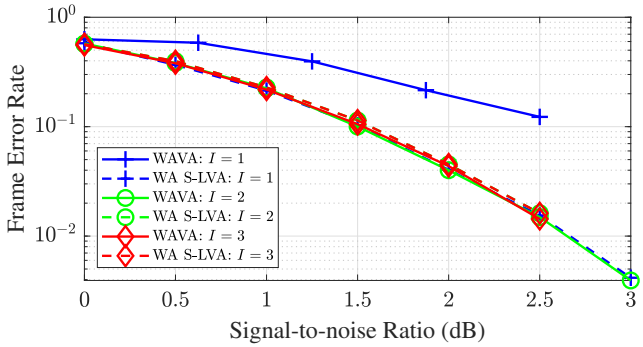

Fig. 3. FER comparison between WAVA and WA S-LVA for $I = 1, 2,$ and 3 for a tail-biting convolutional code without a CRC. The simulation parameters are $k = 64$, $v = 6$, $m = 0$, and $L_{\max} = \infty$.

maximum number of iterations is reached. Once this happens, the decoder checks in monotonically increasing order of cumulative metric whether any end state meets the tail-biting condition. The decoder terminates when it finds the first end state to meet this condition and declares an erasure if no end states meet the tail-biting condition.

Here, I propose a new decoding algorithm that combines the wrap-around behavior of WAVA with the S-LVA for TBCCs and is denoted the *wrap-around serial-list Viterbi algorithm* (WA S-LVA). The motivation of this work is to decrease the expected list size of the list decoding algorithm analyzed in [11] while incurring only a minor penalty in decoding performance. The algorithm takes as a hyperparameter a pre-defined number of trellis iterations I and the received vector y from the channel. The algorithm proceeds in two distinct stages: (1) the wrap-around stage and (2) the list-decoding stage. In the wrap-around stage, the algorithm performs I iterations of add-compare-select along the trellis; each time the end of the trellis is encountered, the initial states of the trellis are initialized to the cumulative metrics in the final states. In stage (2), the algorithm performs the list decoding on the path metric differences as described in [11].

A. Wrap-around serial-list Viterbi decoding of TBCCs without a CRC

Fig. 3 shows the FER vs. SNR for WAVA and WAVA with S-LVA for a tail-biting convolutional code without a CRC. The simulation parameter WAVA with $I = 1$ corresponds to the algorithm analyzed in [11]. Although this parameter setting corresponds to a provably ML decoder for TBCCs, the variance due to the Monte-Carlo simulation was unable to demonstrate this. Simulations not included in this report with fixed noise sequences confirm that WAVA with $I = 1$ has the lowest frame error rate.

An interesting but counter-intuitive result from Fig. 3 is that WAVA with $I = 2$ and $I = 3$ actually has better performance than WA S-LVA and $I = 2$ and $I = 3$. Before this result, my intuition suggested that WA S-LVA and $I = 2$ and $I = 3$ should only perform better than standard WAVA with the same I values, but this turns out to be wrong. When analyzing error

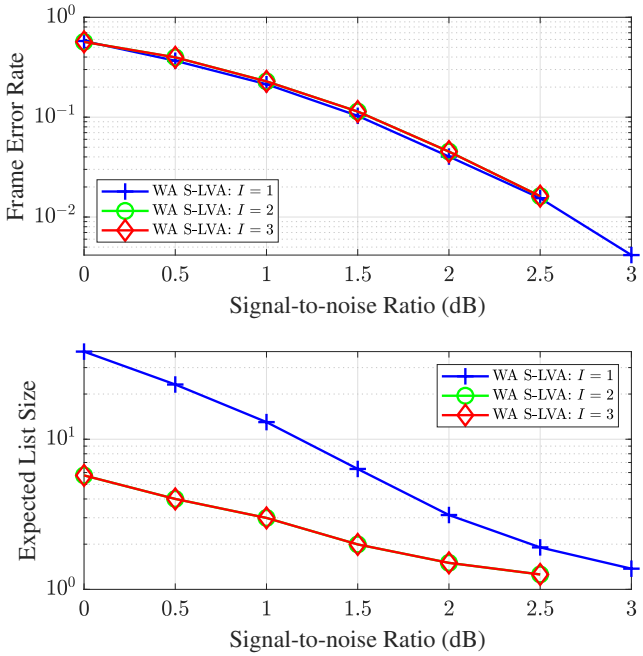

Fig. 4. Top: FER vs. SNR for WA S-LVA for $I = 1, 2,$ and 3 for a TBCC without a CRC. These curves are identical to the corresponding curves from Fig. 3 and are repeated here for to highlight the performance degradation under list-decoding due to using WAVA. Bottom: Expected list size vs. SNR for list decoding of TBCC without a CRC. The simulation parameters are $k = 64$, $v = 6$, $m = 0$ and $L_{\max} = \infty$.

vectors in which the WA S-LVA algorithm decoded incorrectly while the standard WAVA decoder decoded correctly, I found that the codeword computed by the standard WAVA algorithm was typically only a few candidate codewords higher as compared to the codeword computed by standard WAVA. This suggests that the edge metric distortion due to re-initialization of the starting states causes WA S-LVA to perform worse than standard WAVA when no CRC is used.

The top plot in Fig. 4 shows the relative performance of WA S-LVA for different numbers of WAVA iterations. The relative performance degradation due to re-initialization is small on average. The bottom plot depicts the expected list size vs. SNR for each of the three parameter settings. The second iteration of WAVA results in a significant decrease in the expected list size while the third iteration provides no additional benefit. Additional WAVA iterations reduce the expected list size of the list decoder because the first v stages of a tail-biting trellis after only one iteration of WAVA do not experience the full error protection provided by the memory of the convolutional code. As a result, WA S-LVA with $I = 1$ disproportionately checks detour paths along the first v stages of the trellis.

B. Wrap-around serial-list Viterbi decoding of concatenated TBCCs with a CRC

WA S-LVA allows a decoder to improve its error correction capability by performing joint error correction with the CRC. Fig. 5 adds an $m = 6$ CRC to the $v = 3$ convolutional code

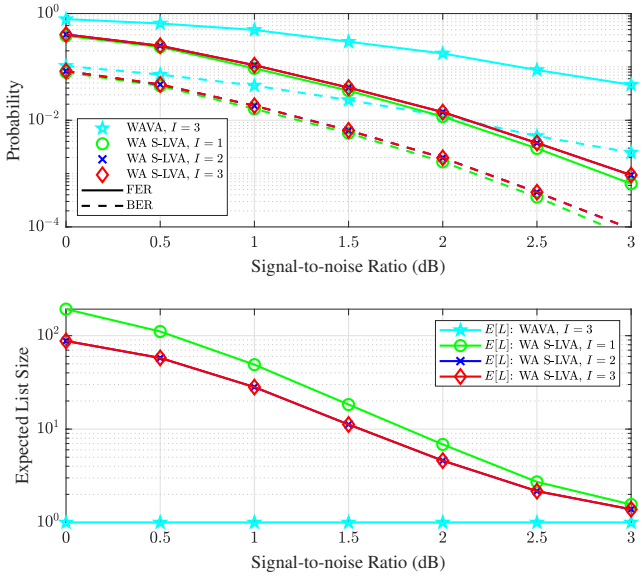

Fig. 5. Top: FER/BER vs. SNR for a TBCC under four different decoding methods. Bottom: Expected list size vs. SNR for each of the four decoding methods. The simulation parameters are $k = 64$, $v = 3$, $m = 6$ and $L_{\max} = \infty$.

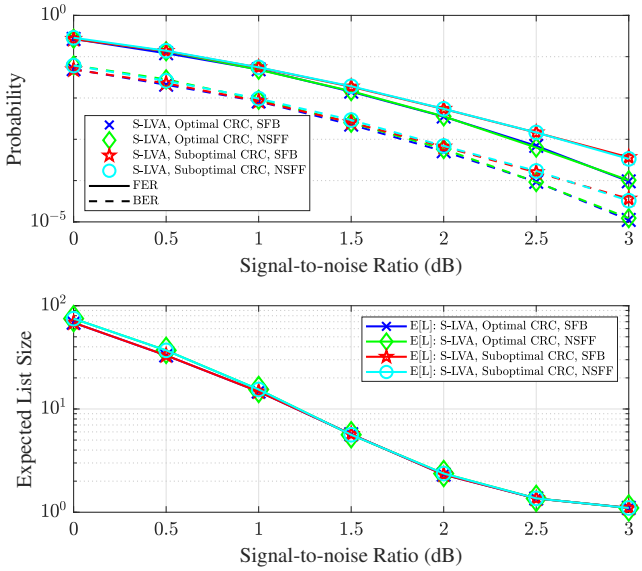

Fig. 6. Top: FER/BER vs. SNR for a ZTCC under S-LVA decoding. 'Optimal CRC' indicates the CRC that was designed using the algorithms described in [7]. The suboptimal CRC was chosen to have a smaller concatenated distance than the optimal CRC. The concatenated hamming distance of the CC with the optimal CRC is 14 whereas the concatenated hamming distance of the CC with a suboptimal CRC is 10. The simulation parameters are $k = 64$, $v = 3$, $m = 8$ and $L_{\max} = \infty$. The optimal CRC polynomial is 0x405 and the suboptimal CRC polynomial is 0x431.

and performs decoding for four different decoder settings. WAVA with $I = 3$ does not utilize the CRC for error correction whereas WA S-LVA uses the CRC for error correction. WA S-LVA clearly outperforms WAVA at all SNRs simulated, due to the improved minimum distance provided by the CRC. The relative improvement due to list decoding increases with SNR over WAVA and the expected list size decreases with increasing SNR. At SNR$=$ 3dB, the expected list size is below 2. As was observed in 4, $I = 3$ provides no further improvement over $I = 2$ in terms of FER, BER, or $E[L]$.

The FER performance in Fig. 5, in which the TBCC is concatenated with a CRC, is superior to that of Fig. 4, where the TBCC is not concatenated with a CRC. The performance degradation due to using WAVA with $I = 2$ and $I = 3$ is larger for $m = 6$ than it is for $m = 0$ but still relatively small. Further analysis is necessary to conclude whether this behavior holds for other combinations of k, m, and v. Again contrasting the two plots, the relative decrease in $E[L]$ due to using WAVA is much greater in Fig. 4 than it is in Fig. 5. I conjecture that this is due to the fact that the relative sizes of v and m are different between the two plots. It is my guess that, as v increases for constant m, the relative decrease in $E[L]$ from $I = 1$ to $I = 2$ becomes larger.

IV. Optimal CRC Design for Systematic Feedback Terminated Convolutional Codes

SFB convolutional codes are often necessary to obtain large coding gains under turbo decoding. Thus, it is likely that designing a CRC to maximize the concatenated distance spectrum of the SFB ZTCC with a CRC would provide larger coding gains than would be obtained with a CRC of the same degree with a smaller concatenated distance.

The CRC design technique in [7] finds a CRC for a fixed k and m to maximize the distance spectrum of a concatenated NSFF CC and CRC code. It is unknown, however, whether this technique also maximizes the distance spectrum of the concatenated code when the inner CC encoder is an SFB CC.

Fig. 6 shows the decoding performance under S-LVA under four CC/CRC designs. In order from top to bottom, the first code is an SFB ZTCC with the distance-spectrum maximizing CRC, the second is an NSFF ZTCC with the distance-spectrum maximizing CRC, the third is an SFB ZTCC with a suboptimal CRC, and the fourth is an NSFF ZTCC with a suboptimal CRC. Because the performance of the S-LVA decoder becomes increasingly dependent on the minimum distance of the code, the NSFF ZTCC with the optimal CRC outperforms both CCs with the non-optimal CRC for higher SNRs. Interestingly, the bottom plot shows little difference in $E[L]$ for any of the four code structures analyzed despite differences in decoding performance at higher SNRs.

One surprising result here is that the FER performance between the NSFF and SFB CCs for both optimal and suboptimal CRC designs are very similar. However, this simulation result could easily just have occurred by chance as the design technique does not guarantee optimal distance properties for SFB CCs. A possible avenue of future work is to show whether or not the CRC design from [7] also indirectly yields the optimal CRC for SFB ZTCCs as well.

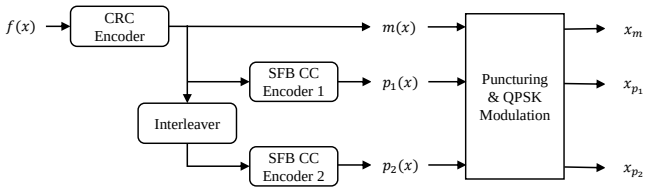

Fig. 7. System model for the parallel concatenated turbo code.

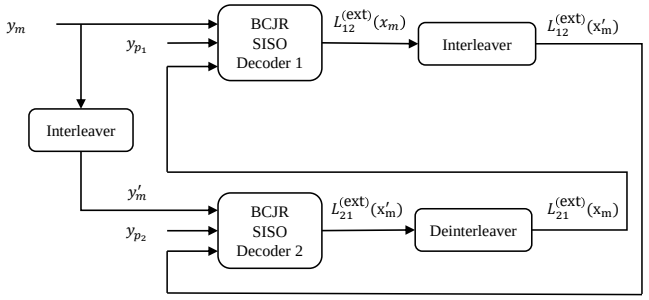

Fig. 8. System model for the parallel concatenated turbo decoder.

V. List Decoding of Parallel Concatenated Turbo Codes

In this section, I first introduce the background and notation used for the encoder and decoder for turbo codes. I then review the algorithm and results in two papers which modified the S-LVA to decode turbo codes. I posit that a CRC designed to maximize the concatenated distance spectrum of the upper SFB CC should improve decoding performance both in terms of the error floor and coding gain. I conclude by explaining my planned implementation of the weighted S-LVA algorithm.

A. Background and Notation

Because the model and implementation of the turbo code closely follow [19], I will only introduce these concepts at a high level. A turbo encoder can be viewed as two parallel SFB CCs connected by an interleaver. Fig. 7 shows the system model for a parallel-concatenated turbo code. Because the systematic bits the output of SFB CC Encoder 1 and SFB CC Encoder 2 are both fully punctured, $p_1(x)$ and $p_2(x)$ consist only of parity bits between generated by these two encoders. x_m, x_{p_1}, and x_{p_2} are the sequences of the transmitted ± 1 symbols after further puncturing is applied and the symbols are modulated onto the channel.

In Fig. 8, y_m, y_{p_1}, and y_{p_2} are the symbols received from the channel after AWGN is added to x_m, x_{p_1}, and x_{p_2}, respectively. The first BCJR soft-input soft-output (SISO) decoder takes as input the observed symbols from the channel and extrinsic information from the second decoder and produces as output extrinsic information $L_{12}^{\text{ext}}(x_m)$ that is passed to the second decoder. The second BCJR SISO takes analogous inputs and provides extrinsic information $L_{21}^{\text{ext}}(x_m^{'})$ that is passed to the first decoder. The $'$ symbol is used to indicate whether the sequence corresponds to the original or interleaved order of the transmitted symbols.

Each BCJR SISO decoder accepts soft information from the channel and extrinsic information from its counterpart decoder and compute extrinsic information that is then passed to its counterpart decoder. The turbo decoder operates in an iterative fashion, using $L_{12}^{\text{ext}}(x'_m)$ to compute $L_{21}^{\text{ext}}(x'_m)$ and then using $L_{21}^{\text{ext}}(x_m)$ to recompute $L_{12}^{\text{ext}}(x_m)$. For the sake of consistency, the turbo decoder terminates when BCJR SISO decoder 1 outputs a decoded sequence that passes the CRC check or a maximum number M iterations is reached.

B. Prior Work

Serial-list Viterbi decoding of turbo codes was first proposed in [12] and is denoted as *tailing* list-sequence (LS) turbo decoding. At a high level, their algorithm performed up to M iterations of standard iterative BCJR decoding, checking their decoded sequence without invoking list decoding against the CRC after each iteration. If the decoder fails to converge to a decoded sequence that passes a CRC check before M iterations, BCJR SISO decoder 1 uses $L_{21}^{\text{ext}}(x_m)$ as prior probabilities to perform weighted serial list decoding. The method of using extrinsic information as prior information in the S-LVA is denoted here as *weighted* S-LVA. In [12], under tailing LS turbo decoding, increasing L_{max} from 3 to 64 yielded very small improvements in coding gain.

In [14], the authors made the observation that further coding gains could be obtained if weighted S-LVA is invoked after each iteration of BCJR rather than only after the M^{th} BCJR iteration failed to converge; this method of decoding is denoted here as *repeated* LS turbo decoding. Repeated LS turbo decoding exhibits superior performance over tailing LS turbo decoding due to the fact that the turbo decoder may have diverged to a decision region far from the ML codeword after M iterations of BCJR decoding. The authors limited their analysis to a maximum list size of $L_{\text{max}} = 3$.

The limited coding gain in [12] from increasing L_{max} from 3 to 64 is a fundamentally different behavior than observed in [10], [11]. I conjecture this is due to the fact that the list decoder was only invoked after M iterations of BCJR decoding, which greatly distorts the metrics upon which the list decoder operates. This conjecture is supported by [14], in which simply invoking weighed S-LVA after each iteration of BCJR yielded an additional coding gain of approximately 0.7 dB.

As shown in [10], [11] and this paper thus far, designing a CRC to maximize the concatenated distance of a CC improves the decoding performance under S-LVA decoding. In addition, large coding gains are observed for values of $L_{\text{max}} \gg 3$. By extending the techniques described in [7] to SFB CCs and decoding the transmitted symbol under repeated LS turbo decoding for larger values of L_{max}, I expect to design a turbo code that approaches the finite length coding bounds of [18] for short to moderate blocklengths.

C. Proposed Model

This implementation of weighted S-LVA will leverage the Euclidean distance based S-LVA decoder implemented for list

decoding of ZTCCs and TBCCs. $L_{21}^{\text{ext}}(x_m)$ will be used as prior information by BCJR SISO decoder 1 to adjust the metrics on each edge of the trellis prior to performing S-LVA. The log-APP value $L_{21}^{\text{ext}}(x_m)$ can be converted [20] to linear probability units via

$$\mathbb{P}[x_m(i) = \pm 1] = \frac{e^{\pm L_{21}^{\text{ext}}(x_m(i))}}{1 + e^{\pm L_{21}^{\text{ext}}(x_m(i))}},$$

where $x_m(i)$ and $L_{21}^{\text{ext}}(x_m(i))$ correspond to the i^{th} transmitted symbol in the sequences x_m and $L_{21}^{\text{ext}}(x_m)$, respectively. Using the extrinsic information, we can solve the following optimization problem:

$$\begin{aligned}
\hat{c}(x) &= \text{argmax}_{c(x)}\, f(x|y) \\
&= \text{argmax}_{c(x)}\, \mathbb{P}[x] f(y|x) \\
&= \text{argmax}_{c(x)} \prod_{i=1}^{N} \mathbb{P}[x_i] f(y_i|x_i) \\
&= \text{argmax}_{c(x)} \prod_{i=1}^{N} \frac{\mathbb{P}[x_i]}{\sqrt{2\pi\sigma^2}} e^{-\frac{\|y_i - x_i\|^2}{2\sigma^2}} \\
&= \text{argmax}_{c(x)} \prod_{i=1}^{N} \mathbb{P}[x_i] e^{-\frac{\|y_i - x_i\|^2}{2\sigma^2}} \\
&= \text{argmax}_{c(x)} \sum_{i=1}^{N} - \left(\frac{\|y_i - x_i\|^2}{2\sigma^2} - \log(\mathbb{P}[x_i]) \right) \\
&= \text{argmin}_{c(x)} \sum_{i=1}^{N} \left(\frac{\|y_i - x_i\|^2}{2\sigma^2} - \log(\mathbb{P}[x_i]) \right).
\end{aligned}$$

Notice that $L_{21}^{\text{ext}}(x_m)$ provides information only for transmitted symbols $x_i \in x_m$; if $x_i \notin x_m$, $\log(\mathbb{P}[x_i]) = 0$. For an SFB rate-$1/n$ CC, there is exactly one symbol that corresponds to a systematic bit per edge in the trellis. Therefore, weighted S-LVA incorporates the extrinsic information by adjusting each edge in the trellis by $\log(\mathbb{P}[x_i])$. Earlier, we showed that standard S-LVA operates on the linear Euclidean distance metrics on each branch to produce a list of possible decoded sequences in order of increasing metric distance sequentially. Weighted S-LVA also generates a list of possible decoded sequences but operates on the adjusted edge metrics instead.

VI. Conclusions and Future Works

In this paper, I extended the work in [10], [11] to perform S-LVA on a SFB CC. I also proposed a new decoding algorithm that incorporated WAVA to reduce the expected list size of S-LVA for TBCCs. I provided numerical simulations that showed that a CRC designed to maximize the concatenated distance of an NSFF CC also provided similar decoding performance when concatenated with the equivalent SFB CC. I also implemented the BCJR algorithm for NSFF and SFB CCs, which will serve as the basis for a weighted S-LVA decoder for turbo codes. Finally, I examined the tailing LS turbo decoding and repeated LS turbo decoding algorithms, identified deficiencies in their respective implementations, and proposed a modified design that should provide additional coding gains and result in a reduced error floor.

Based on the current results, I propose the following directions for future work:

- Complete the weighted S-LVA turbo decoder and simulate the performance at short to moderate blocklengths using CRCs designed by the techniques described in [7].
- Extend the work in [7] to SFB CCs.
- Explore different turbo decoder architectures, including serially concatenated turbo codes with either NSFF and SFB outer CCs.
- Simulate the performance of S-LVA and weighted S-LVA on channels other than the AWGN channel.
- Explore whether machine learning techniques can provide better extrinsic information for use by the weighted S-LVA algorithm.

References

[1] M. Shirvanimoghaddam, M. S. Mohammadi, R. Abbas, A. Minja, C. Yue, B. Matuz, G. Han, Z. Lin, W. Liu, Y. Li, S. Johnson, and B. Vucetic, "Short block-length codes for ultra-reliable low latency communications," *IEEE Communications Magazine*, vol. 57, no. 2, pp. 130–137, February 2019.

[2] H. Ma and J. Wolf, "On tail biting convolutional codes," *IEEE Transactions on Communications*, vol. 34, no. 2, pp. 104–111, February 1986.

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

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

[5] 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, pp. 284–287, March 1974.

[6] N. Seshadri and C.-E. Sundberg, "List viterbi decoding algorithms with applications," *IEEE transactions on communications*, vol. 42, no. 234, pp. 313–323, 1994.

[7] C. 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.

[8] R. V. Cox and C. E. W. Sundberg, "An efficient adaptive circular viterbi algorithm for decoding generalized tailbiting convolutional codes," *IEEE Transactions on Vehicular Technology*, vol. 43, no. 1, pp. 57–68, Feb 1994.

[9] R. Y. Shao, Shu Lin, and M. P. C. Fossorier, "Two decoding algorithms for tailbiting codes," *IEEE Transactions on Communications*, vol. 51, no. 10, pp. 1658–1665, Oct 2003.

[10] H. Yang, S. V. S. Ranganathan, and R. D. Wesel, "Serial list viterbi decoding with crc: Managing errors, erasures, and complexity," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec 2018, pp. 1–6.

[11] E. Liang, H. Yang, D. Divsalar, and R. D. Wesel, "List-decoded tailbiting convolutional codes with distance-spectrum optimal crcs for 5g," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec 2019, pp. 1–6.

[12] K. R. Narayanan and G. L. Stuber, "List decoding of turbo codes," *IEEE Transactions on Communications*, vol. 46, no. 6, pp. 754–762, June 1998.

[13] C. F. Leanderson and C. W. Sundberg, "The max-log list algorithm (mlla)-a list-sequence decoding algorithm that provides soft-symbol output," *IEEE Transactions on Communications*, vol. 53, no. 3, pp. 433–444, March 2005.

[14] ——, "On list sequence turbo decoding," *IEEE Transactions on Communications*, vol. 53, no. 5, pp. 760–763, May 2005.

[15] ——, "Performance evaluation of list sequence map decoding," *IEEE Transactions on Communications*, vol. 53, no. 3, pp. 422–432, March 2005.

[16] Chunlong Bai, B. Mielczarek, W. A. Krzymien, and I. J. Fair, "Efficient list decoding for parallel concatenated convolutional codes," in *2004 IEEE 15th International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE Cat. No.04TH8754)*, vol. 4, Sep. 2004, pp. 2586–2590 Vol.4.

[17] C. Bai, B. Mielczarek, W. A. Krzymien, and I. J. Fair, "Improved analysis of list decoding and its application to convolutional codes and turbo codes," *IEEE Transactions on Information Theory*, vol. 53, no. 2, pp. 615–627, Feb 2007.

[18] Y. Polyanskiy, H. V. Poor, and S. Verdu, "Channel coding rate in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.

[19] W. Ryan and S. Lin, *Channel codes: classical and modern*. Cambridge university press, 2009.

[20] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, March 1996.