Complementary Exclusion of Full Polynomials to Enable Dual List Decoding of Convolutional Codes

Zihan Qu, Amaael Antonini, Wenhui Sui, Eugene Min, Arthur Yang, and Richard D. Wesel
Department of Electrical and Computer Engineering
University of California, Los Angeles
Email: {brucequ, amaael, wenhui.sui, eugenemin, arthuryang27, wesel}@ucla.edu

***Abstract*—Convolutional codes have been widely studied and used in many systems. As the number of memory elements increases, frame error rate (FER) improves but computational complexity increases exponentially. Recently, decoders that achieve reduced average complexity through list decoding have been demonstrated when the convolutional encoder polynomials share a common factor that can be understood as a CRC or more generally an expurgating linear function (ELF). However, classical convolutional codes avoid such common factors because they result in a catastrophic encoder. This paper provides a way to access the complexity reduction possible with list decoding even when the convolutional encoder polynomials do not share a common factor. Decomposing the original code into component encoders that fully exclude some polynomials can allow an ELF to be factored from each component. Dual list decoding of the component encoders can often find the ML codeword. Including a fallback to regular Viterbi decoding yields excellent FER performance while requiring less average complexity than always performing Viterbi on the original trellis. A best effort dual list decoder that avoids Viterbi has performance similar to the ML decoder. Component encoders that have a shared polynomial allow for even greater complexity reduction.**

I. Introduction

Convolutional codes (CCs) are one of the most widely used codes in modern wireless communications. A widely used decoding method that minimizes the frame error rate [1] is with the Viterbi algorithm, by A. Viterbi [2].

A common way of terminating a CC is to drive the states back to the zero state after processing all message bits, i.e. appending ν zeros to the end of a message, where ν is the number of memory elements in a CC. In this paper, we explore the efficient decoding of such zero-terminated (ZT) CCs.

In recent works, cyclic redundancy checks have been concatenated with CCs and used to expurgate CC codewords with low Hamming weights, thus improving the minimum distance $D_{\min}$ and decreasing the number of nearest neighbors $A_{d_{\text{free}}}$ at $D_{\min}$. Lou *et al.* [3] presented an algorithm that designs optimal CRCs for ZTCCs. In [4], Yang *et al.* furthered the work by jointly designing distance spectrum optimal (DSO) CRCs and CCs at target frame error rates (FERs) of 10^{-2}, 10^{-3}, and 10^{-4}. Yang *et al.* showed that the concatenation of CCs with DSO CRCs can approach the random coding union (RCU) bound [5]. Since these expurgating functions do not have to be cyclic in nature, we will thereby follow [6] and refer to them as expurgating linear functions (ELFs). Other recent work on CCs concatenated with ELFs include [7], [8], and [9]. The concatenation of an ELF with a CC is itself a CC, although the ELF being a common factor of all of the encoder polynomials would make the encoder catastrophic.

The reason why the concatenation of an ELF with degree m and a CC with ν encoder memories is preferable to a CC with $\nu + m$ encoder memories is that the ELF-CC can be decoded with low average complexity using list decoding, as described by Seshadri and Sundberg [10]. Using list Viterbi decoders, the best L decoding estimates are produced either simultaneously using a parallel list Viterbi algorithm or sequentially using a serial list Viterbi algorithm (SLVA). For an ELF-CC, SLVA is performed on a trellis with 2^{ν} states rather than one with $2^{\nu+m}$ states. Often, at the operating point of interest the average list size of SLVA is one so that the average complexity of the decoder is essentially the complexity of performing Viterbi on a trellis with 2^{ν} states.

Since the seminal paper by Seshadri and Sundberg, substantial progress has been made to reduce the complexity of the list Viterbi algorithm (LVA). In [11], M. Roder and R. Hamzaoui proposed a new implementation of LVA that is linear in both time and space by maintaining an ordered list of path metric differences using a Red-Black tree. The same performance can be achieved using a Min Heap [12], which is proven to have the same lowest runtime of inserting elements as the Red-Black tree. Considerable effort was dedicated to analyzing the complexity of list decoders. In [13], [14], the decoding complexity of using LVA is analyzed and quantified, concluding that the expected list size determines the overall complexity of a list decoder.

While it's possible, decoding CCs with large ν is inefficient because the complexity of Viterbi decoding increases exponentially with ν. In our paper, we propose a low complexity decoder consisting of a pair of list decoders, which we refer to as the Dual List Decoder (DLD).

A. Contributions

This paper presents list decoding algorithms where the decoding complexity of a ZTCC with large ν is reduced to below the complexity of standard soft Viterbi (SSV). For ZTCCs with large ν, building the trellis and running a single

This research is supported by National Science Foundation (NSF) grant CCF-2008918. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect views of NSF.

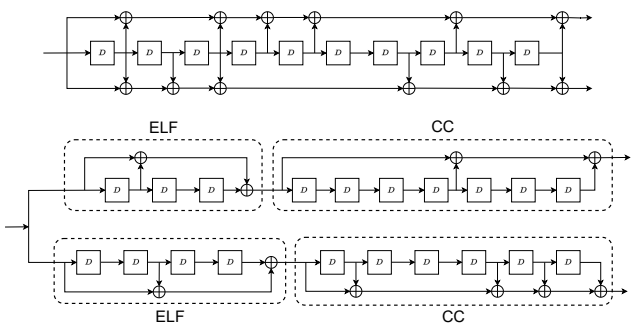

Fig. 1. Convolutional code encoding circuit with $\nu = 10$ from Lin & Costello [15] having generator polynomial $(p_1(x), p_2(x)) = (1 + x + x^3 + x^4 + x^5 + x^8 + x^{10}, 1 + x + x^2 + x^3 + x^7 + x^9 + x^{10})$. The lower graph shows that these two polynomials can be decomposed into $(1 + x + x^3) \cdot (1 + x^4 + x^7)$ and $(1 + x^2 + x^4) \cdot (1 + x + x^4 + x^5 + x^6)$, respectively.

traceback takes high computational complexity and memory space. The complexity reduction possible with list decoding can be achieved even for a CC that lacks the explicit ELF-CC structure. To accomplish this, we decompose the CC into two shorter component encoders, each of which has the ELF-CC structure. To reveal the ELF-CC structure, we factor the polynomial for each component encoder into two smaller ones, where one polynomial acts as a smaller CC polynomial and the other polynomial is seen as an ELF that expurgates low-weight codewords of the smaller CC. Dual list decoding of the two component ELF-CC codes offers a substantial decrease in decoding complexity.

As we will see, a DLD that performs list decoding of both component encoders in parallel can often find the ML codeword. Including a fallback to regular Viterbi decoding on the original trellis when needed produces an overall decoder with excellent FER performance and less average complexity than standard SSV. Designing an encoder where the component encoders share a common constituent polynomial increases the redundancy of the component encoders which drastically improves the dual list decoder's performance and leads to significantly more complexity reduction over SSV.

B. Organization

The rest of the paper proceeds as follows. Sec. II describes the system model, ZTCCs, standard Viterbi decoder, and serial list Viterbi decoder. We also briefly analyze the complexity of the SSV decoder given the number of memory elements of a ZTCC. In Sec. III, we present three DLD decoding approaches. Sec. III-C explores another strategy that reduces the decoding erasure rate by sharing information between the two list decoders. Sec. IV shows the FER vs. E_b/N_0 simulation results of using different decoding schemes and compares their decoding complexities. Finally, Sec. V concludes the paper.

II. Complementary Polynomial Exclusion (CPE)

A convolutional code encoder consists of linear finite-state shift registers and is designed to process inputs and generate outputs based on the current states of memory elements. In the context of CC, the encoding trellis is commonly represented by a generator polynomial. Let k be the number of message bits, ν be the number of encoder memories, n be the number of output symbols, and $r = k/n$ be the rate of a CC. m denotes the degree of the CRC or ELF. We use $\log(\cdot)$ to denote the log-2 logarithm. $\mathbb{E}(X)$ is denoted as the expected value of the random variable X. The polynomial representation we use is in GF(2).

For a ZTCC, we append ν zeros to k message bits to ensure the feed-forward encoder starts from and ends in the all-zero state. The trailing zeros lead to a total number of received symbols $n = \frac{1}{r}(k + \nu)$ so that the actual rate is lower than before. In Fig. 1, the upper plot shows a $\nu = 10$ feed-forward rate-1/2 convolutional encoder with generator polynomial $(2473, 3217)$ represented in octal. A rate-$1/2$ ZTCC can be decoded by performing SSV that starts and terminates at the 0_{th} state.

A ZTCC with a larger ν typically results in lower FER. The construction complexity of a trellis is directly proportional to the number of trellis states. For a CC with ν memory elements, the associated complexity is dictated by the number of states, which amounts to 2^ν. As shown in [14], we can achieve a reduction in average complexity when a "CRC polynomial" or ELF can be factored out of all of the constituent polynomials. While [14] designs ELF-CC codes that have this structure, our paper proposes an approach to reduce decoding complexity for cases where the constituent polynomials can each be factored but do not share a common factor.

Looking again at Fig. 1, we can use complementary polynomial exclusion (CPE) to decompose the rate-1/2 convolutional code into two rate-1/1 component encoders. Note that each component encoder still has some redundancy because of termination. One component encoder is created by excluding the polynomial $p_1(x) = 1 + x + x^3 + x^4 + x^5 + x^8 + x^{10}$. The other component encoder is created by excluding the polynomial $p_2(x) = 1 + x + x^2 + x^3 + x^7 + x^9 + x^{10}$. Each of these component encoder polynomials can be factored into an ELF and a CC so that each component encoder can be decoded by a list decoder. In our example, we can decompose $p_1(x)$ into two smaller polynomials, i.e., $1 + x + x^3 + x^4 + x^5 + x^8 + x^{10} = (1 + x^4 + x^7) \cdot (1 + x + x^3)$. Similarly, $p_2(x) = 1 + x + x^2 + x^3 + x^7 + x^9 + x^{10} = (1 + x + x^4 + x^5 + x^6) \cdot (1 + x^2 + x^4)$. We will denote the factorization of $p_1(x)$ as $g_{cc,1}$ and $g_{crc,1}$. Likewise, $p_2(x)$ can be factored into polynomials defined as $g_{cc,2}$ and $g_{crc,2}$. If a codeword generated by $g_{cc,1}$ satisfies $g_{crc,1}$, then it is a codeword of the code $p_1 = g_{cc,1} * g_{crc,1}$. The Euclidean distance path metric obtained from both component encoders should add up to the path metric obtained by decoding with SSV using the original rate-1/2 code with generator polynomial $(p_1(x), p_2(x))$.

III. Dual List Decoding (DLD)

In dual list decoding, two decoders collaborate to decode the message. For a given overall convolutional code C_0 with ν_0 memory elements, the SSV decoder has complexity of 2^{ν_0}. Dual list decoding replaces the SSV decoder with two list decoders, constructed for two distinct component convolutional encoders C_1 and C_2, each concatenated with outer

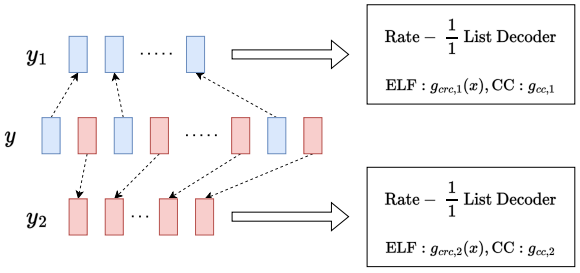

Fig. 2. The sequence y is demultiplexed into sequence y_1 from component encoder $p_1(x)$ and y_2 from component encoder $p_2(x)$. Sequences y_1 and y_2 are then decoded separately by two rate - 1/1 list decoders for the component encoders that have been factored into an ELF and CC.

ELFs. Taken together, the transmitted bits produced by the two component encoders C_1 and C_2 provide the identical bit stream as the transmitted bits produced by the overall convolutional code C_0. However, the component encoders C_1 and C_2, each has corresponding lower numbers of memory elements (not including the ELF) such that $\nu_1 < \nu_0$ and $\nu_2 < \nu_0$, which allows the dual list decoder to provide a complexity advantage over the SSV decoder for C_0, as long as the expected list size is not too large.

A. DLD for Complementary Polynomial Exclusion (CPE)

We now explain the dual list decoder for the code implemented by the circuits of Fig. 1. The code has generator polynomials of degree $\nu = 10$, while component $g_{cc,1}$ has degree $\nu_1 = 7$ and component $g_{cc,2}$ has degree $\nu_2 = 6$. Since we divide the original generator polynomial $p_1(x)$ and $p_2(x)$, each dual list decoder would only use half of the received symbols. For this, we divide the received symbols into two sequences based on the polynomials that generated them. All symbols generated by the first polynomial $p_1(x)$ are used by one decoder and the symbols generated by the second polynomial $p_2(x)$ are used by the other decoder. As shown in Fig. 2, this requires "unravelling" the received sequence, y, as if it was encoded by $p_1(x)$ and $p_2(x)$ separately and interleaved together. Let y_1 and y_2 denote the two sequences encoded by each component encoder.

The first decoder enumerates a possible path through a trellis constructed using y_1, starting from the path with the lowest metric, pausing once a codeword that satisfies $g_{crc,1}$ is found. We store this codeword and its corresponding message as a candidate in an associative array. Then we move on to the other decoder that runs the same steps on y_2. When the same codeword has been found both by the decoder for y_1 and the decoder for y_2, we have a potentially good estimate of the transmitted codeword.

B. Verification (or not) and falling back to SSV

Now we consider making the best use of the results from the two list decoders. For a general DLD system, the first codeword that appears on both the list for C_1 and the list for C_2 is not necessarily the maximum-likelihood (ML) estimate of the transmitted codeword. One decoding option is to declare

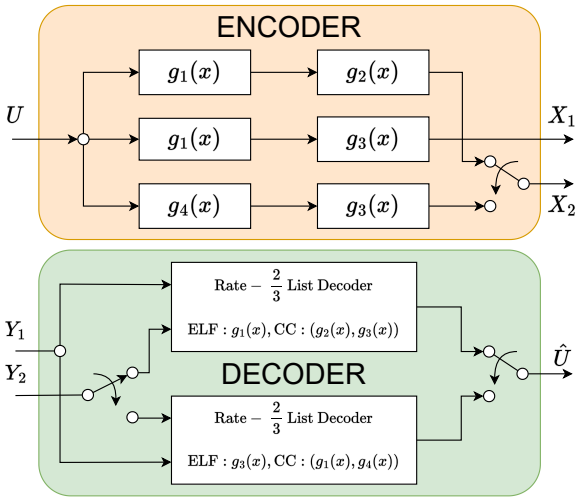

Fig. 3. Encoder and dual list decoder for a rate-1/2 code decomposed into two rate-2/3 component encoders created by complementary polynomial exclusion with a shared polynomial $g_1(x)g_3(x)$. The rate-1/3 mother code is punctured to rate 1/2 by puncturing every other bit from the two polynomials that are excluded from one of the component encoders.

an erasure unless the decoding results (the metric information from both lists) is sufficient to conclude that the codeword that appears on both lists with the best metric is definitely the ML codeword. We refer to this decoder, which returns either the ML codeword or an erasure, as the DLD-ML/E decoder.

Since in practice, the erasure rate can be an order of magnitude larger than the desired FER but still quite small (say 10^{-3}), a reasonable option is to fall back to running SSV when DLD-ML/E produces an erasure. We refer to this decoding approach as DLD-ML/SSV because DLD will either return a known ML codeword or fall back to using SSV.

Yet another approach is to consider a best-effort DLD that runs to a maximum list size and does the best that it can. Such a decoder does not always return the ML decisions but has the benefit of never employing a full SSV decoder. For this decoder, we note that once a codeword has appeared in one list, the decoder knows the message and can easily compute the output associated with that message for the other list and hence its overall decoding metric. This is important because even if a codeword fails to appear on the other list before the maximum list size is reached, it may still be the ML codeword. With this in mind, we run the two list decoders to a maximum list size and return the codeword with the best available metric (BAM) regardless of whether that codeword has appeared on both lists. We call this the DLD-BAM decoder.

C. DLD for CPE with a Shared Polynomial (SP)

We now introduce the paradigm of complementary polynomial exclusion with a shared polynomial (CPE-SP) as a way to increase the redundancy of the component encoders, which improves the performance of the dual list decoders by reducing the expected list size.

Fig. 3 illustrates a rate-1/2 convolutional encoder structure that provides increased redundancy to the dual list decoders.

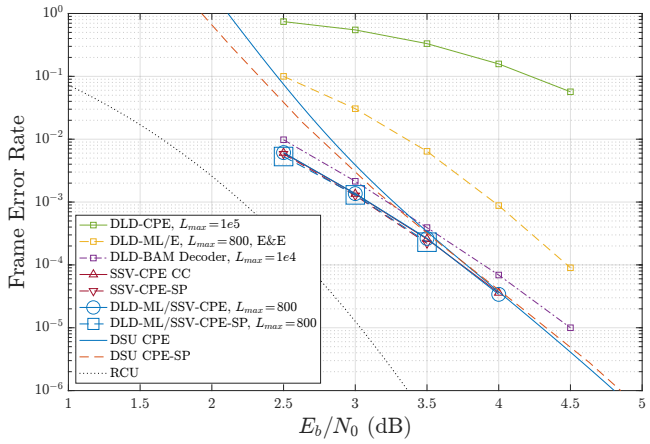

Fig. 4. FER vs. E_b/N_0 simulation results for DLD-ML/E, DLD-ML/SSV, DLD-BAM, and SSV decoding of two codes. The first code is the $\nu = 10$ rate-1/2 code from Lin & Costello [15]. The second code is a newly designed $\nu = 10$ rate-1/2 code obtained by puncturing a rate-1/3 code. This second code is decomposed via CPE-SP into two rate-2/3 codes that are decoded with DLD-ML/SSV and DLD-BAM decoder.

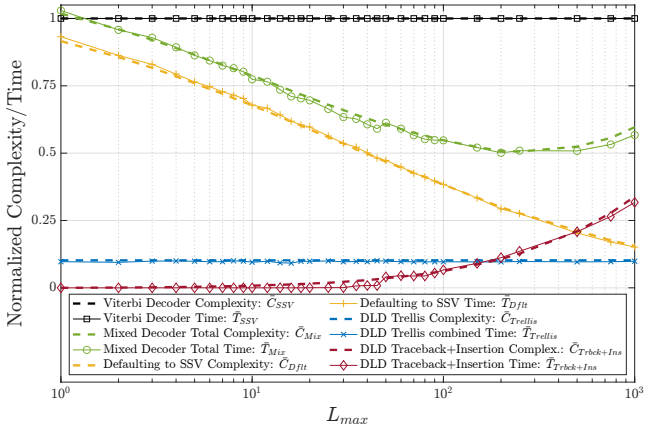

Fig. 5. Run time and complexity of DLD-ML/SSV decoder for CPE as a function of maximum list size for the $\nu = 10$ rate-1/2 code from Lin & Costello [15] decoded using dual list decoding at $\frac{E_b}{N_0} = 5.13$ dB. All variables are normalized by the time or complexity of the SSV decoding. In the simulation setting, $c_1 = 0.3$, $c_2 = 0.4$.

The rate-1/2 code is decomposed into two rate-2/3 component encoders created by complementary polynomial exclusion with a shared polynomial $g_1(x)g_3(x)$. The rate-1/3 mother code is punctured to rate 1/2 by puncturing every other bit from the two polynomials $g_1(x)g_2(x)$ and $g_4(x)g_3(x)$ that are excluded from one of the component encoders. The first component encoder is the rate-2/3 code with polynomials $g_1(x)g_2(x)$ and $g_1(x)g_3(x)$ so that $g_1(x)$ serves as the ELF and $[g_2(x), g_3(x)]$ is the CC. Note that every other bit is punctured from the output of $g_1(x)g_2(x)$. The second component encoder is the rate-2/3 code with polynomials $g_1(x)g_3(x)$ and $g_4(x)g_3(x)$ so that $g_3(x)$ serves as the ELF and $[g_1(x), g_4(x)]$ is the CC. Note that every other bit is punctured from the output of $g_4(x)g_3(x)$.

The polynomials $g_1(x)$, $g_2(x)$, $g_3(x)$, and $g_4(x)$ were selected from among all degree-5 polynomials to maximize the minimum distance of the rate-1/2 code. The selected polynomials are defined as follows:

$$g_1(x) = x^5 + x^2 + 1 \tag{1}$$
$$g_2(x) = x^5 + x^4 + x^3 + x^2 + 1 \tag{2}$$
$$g_3(x) = x^5 + x + 1 \tag{3}$$
$$g_4(x) = x^5 + x^4 + x^2 + x + 1\,. \tag{4}$$

Note that with CPE-SP, the sum of the Euclidean metrics obtained from the two dual list decoders for a specified message will now exceed the total metric for that message decoded by SSV since the portion of the metric corresponding to output of the shared polynomial is double counted. Therefore, designing a stopping criterion for the CPE-SP enhanced DLD with shared information is slightly more involved.

While CPE-SP places significant constraints on the encoder polynomials, our distance-spectrum union (DSU) bound and simulated FER analysis in Fig. 4 both confirm that these constraints do not significantly impact performance of the overall encoder C_0 as compared to an overall convolutional encoder that is completely unconstrained with the same ν_0.

IV. Results

We now show the simulation result in FER and the complexity required to achieve target FER using DLD for CPE and DLD for CPE-SP.

A. FER Performance

DLD decomposes convolutional encoders with large ν into smaller ones that would enable serial list decoding. The ELF codes serve to expurgate low weight codewords. However, as we can see in Fig. 4, the FER of DLD for CPE is significantly higher than that of SSV. The rate-1/1 codes have such low redundancy that an extremely large list size is needed to recover the ML codeword. However, we observed that the DLD for CPE does not make decoding errors. Rather, it simply fails to find an ELF-TB codeword and declares NACK rather than undetected errors, which means the FER curve can be interpreted as the percentage that are decoded by DLD. For example, around $E_b/N_0 = 4.15$ dB, DLD for CPE with a $L_{max} = 10^5$ would declare NACK one out of 10 times on average. As we show in Fig. 5, a mixed DLD/SSV for CPE, that falls back to run the full complexity SSV whenever DLD fails to find an ELF-TB codeword, offers both ML performance and significant complexity reduction if the E_b/N_0 is high enough for DLD to find an ELF-TB codeword most of the time.

We also improve the dual decoding algorithm by sharing received symbols as elaborated in the previous section. Looking at Fig. 4, DLD for CPE-SP dramatically improves the percentage decoded by DLD and shrinks the list size required to find a match. For example, at FER 10^{-4} and max list size = 800, DLD declares NACK every one out of 500 times, which means that the mixed DLD/SSV for CPE-SP decoder resorts to SSV one out of 500 times on average.

B. Complexity Decrease of DLD over SSV

The overall complexity of DLD is dominated by the complexity of maintaining separate ordered lists in both decoders.

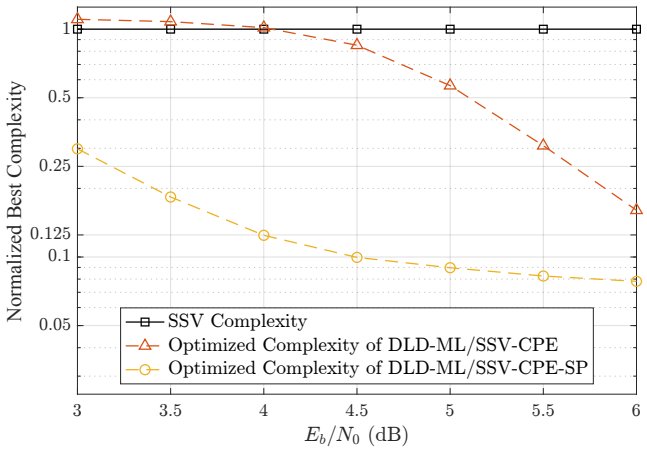

Fig. 6. Normalized complexity for optimized maximum list sizes vs. E_b/N_0 for two codes. The first code is the $\nu = 10$ rate-1/2 code from Lin & Costello [15]. The second code is the newly designed $\nu = 10$ rate-1/2 code obtained by puncturing a rate-1/3 code. Normalization is with respect to SSV.

Let L be a random variable defined as the "list rank" required by S-LVA to search for a codeword that passes the CRC. In a dual-list structure, two random variables L_1 and L_2 are defined for each decoder to be the corresponding "list ranks" of the first matching codeword found on both lists. We developed theoretical complexity expressions based on the methods by Yang *el al.* [14]. The average complexity C_{SLVD} of a rate-$1/\omega$ CRC-ZTCCs SLVD of is decomposed into three components:

$$C_{\text{SLVD}} = C_{\text{SSV}} + C_{\text{trace}} + C_{\text{list}}\,. \tag{5}$$

C_{SSV} denotes the complexity of summed add-compare-select (ACS) operations on a zero-terminated CC trellis, C_{trace} denotes the complexity of the *additional* traceback operations required by SLVD, and C_{list} denotes the average complexity of inserting new elements to maintain an ordered list of path metric differences.

In [14], these metrics are evaluated by Eq. 6 through Eq. 9, where $\mathsf{E}[I]$ is the expected number of insertions to maintain the sorted list of path metric differences.

$$\begin{aligned} C_{\text{SSV}} &= 2^{v+1} - 2 + 1.5(2^{v+1} - 2) + 1.5(k+m+v)2^{v+1} \\ &\quad + c1[2(k+m+v) + 1.5(k+m)] && (6)\\ C_{\text{trace}} &= 3.5c_1(\mathbb{E}[L] - 1)(K+m) && (7)\\ C_{\text{list}} &= c_2\mathbb{E}[I]\log(\mathbb{E}[I]) && (8)\\ \mathbb{E}[I] &\leq (K+m)\mathbb{E}[L] + 2^v - 1\,. && (9) \end{aligned}$$

Similarly, we defined C_{Trellis} as the combined complexity of constructing the lower order trellises for DLD. $C_{\text{Traceback+Insertion}}$ is defined as the combined complexity of the *additional* traceback operations and the maintenance of an ordered list of path metrics required by both DLDs. We use C_{DLD} to denote the total complexity of the dual list decoder without defaulting to SSV.

$$\begin{aligned} C_{\text{Trellis}} &= C_{\text{SSV},1} + C_{\text{SSV},2} && (10)\\ C_{\text{Traceback+Insertion}} &= C_{\text{trace},1} + C_{\text{list},1} + C_{\text{trace},2} + C_{\text{list},2} && (11)\\ C_{\text{DLD}} &= C_{\text{Trellis}} + C_{\text{Traceback+Insertion}}\,. && (12) \end{aligned}$$

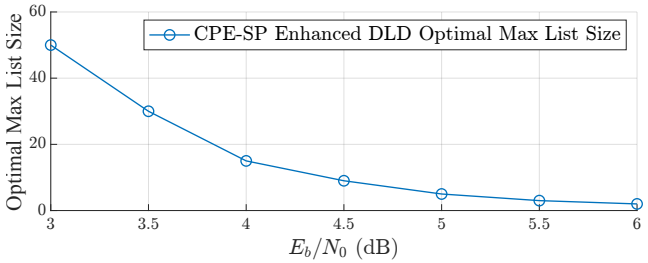

Fig. 7. The optimal constrained maximum list size vs. E_b/N_0 for DLD-ML/SSV that uses a rate - 1/2 code produced by puncturing a rate - 1/3 code given by (2267, 3613, 2353). At each E_b/N_0, a selection of list sizes (1 - 1000) is considered and the best list size is the one that results in the lowest runtime. As E_b/N_0 increases, the optimal max list size converges to 1.

In order to compare the complexity of DLD and that of SSV, the normalized complexity $\bar{C}$ is defined as the complexity divided by the complexity of performing SSV, i.e.,

$$\bar{C}_{\text{DLD}} = \bar{C}_{\text{Trellis}} + \bar{C}_{\text{Traceback+Insertion}}\,. \tag{13}$$

The complexity of a DLD-ML/SSV decoder is C_{DLD} plus the complexity of running SSV whenever DLD declares NACK. The time measured for each component in (13) were normalized by T_{SSV} to get $\bar{T}$. The run-time of the aforementioned components were recorded on an Intel E5 2670 using Visual Studio C++14. Fig. 5 shows the normalized complexity derived from (10) - (13) and the normalized runtime. Normalization is done by dividing by the complexity or runtime associated with SSV, which includes performing ACS operations on the trellis and a single traceback. Fig. 5 shows that the normalized complexity and normalized runtime matches. In addition, the complexity of CPE-DLD is shown to be dominated by the complexity of maintaining an ordered list of path metric differences with high constrained maximum list size.

In Fig. 6, we show that DLD-ML/SSV for CPE-SP has a much lower complexity at $E_b/N_0 = 3.0$ and further converges to ~ 0.07 at high E_b/N_0. The DLD-ML/SSV for CPE, on the other hand, started with a complexity higher than that of SSV decoding and decreased quickly after $E_b/N_0 = 4.5$. In Fig. 7, we show that the optimal constrained maximal list size converges to 1 as E_b/N_0 increases for DLD-ML/SSV for CPE-SP. This is because we tend to find the ML and correct decoding very early in both lists at high E_b/N_0.

V. Conclusion

This paper decomposes an overall convolutional encoders C_0 into component encoders C_1 and C_2 that allow an ELF to be factored from each component. Parallel list decoding of the component encoders allows a significant complexity reduction as compared to regular Viterbi decoding applied to the original encoder. New encoders with constraints that further reduce the complexity of DLD by increasing the redundancy of the component codes reduce complexity by an order of magnitude while still yielding excellent FER performance.

REFERENCES

[1] G. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, March 1973.

[2] 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.

[3] 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, October 2015.

[4] 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)*, December 2018, pp. 1–6.

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

[6] R. D. Wesel, A. Antonini, L. Wang, W. Sui, B. Towell, and H. Grissett, "ELF codes: Concatenated codes with an expurgating linear function as the outer code," *2023 12th International Symposium on Topics in Coding (ISTC)*, pp. 287–291, June 2023.

[7] A. Antonini, W. Sui, B. Towell, D. Divsalar, J. Hamkins, and R. D. Wesel, "Suppressing error floors in SCPPM via an efficient CRC-aided list viterbi decoding algorithm," *2023 12th International Symposium on Topics in Coding (ISTC)*, pp. 221–225, October 2023.

[8] W. Sui, B. Towell, A. Asmani, H. Yang, H. Grissett, and R. D. Wesel, "CRC-aided high-rate convolutional codes with short blocklengths for list decoding," *IEEE Transactions on Communications*, vol. 72, no. 1, pp. 63–74, December 2022.

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

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

[11] M. Roder and R. Hamzaoui, "Fast tree-trellis list viterbi decoding," *IEEE Transactions on Communications*, vol. 54, no. 3, pp. 453–461, March 2006.

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

[13] H. Yang, E. Liang, and R. D. Wesel, "Joint design of convolutional code and crc under serial list viterbi decoding," November 2018.

[14] H. Yang, E. Liang, M. Pan, and R. D. Wesel, "CRC-aided list decoding of convolutional codes in the short blocklength regime," *IEEE Trans. Inf. Theory*, vol. 68, no. 6, pp. 3744–3766, February 2022, early access. [Online]. Available: https://doi.org/10.1109/TIT.2022.3150717

[15] S. Lin and D. J. Costello, *Error Control Coding: fundamentals and applications*. New Jersey, USA: Pearson Prentice Hall, May 2004.