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

Wenhui Sui *, Brendan Towell *, Zihan Qu, Eugene Min, and Richard D. Wesel
Department of Electrical and Computer Engineering
University of California, Los Angeles
Email: {wenhui.sui, brendan.towell, brucequ, eugenemin, wesel}@ucla.edu

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

I. Introduction

A. Background

For a feedforward encoder with ν memory elements implementing a tail-biting convolutional code (TBCC) [1], the tail-biting condition can be enforced by setting the initial encoder state with the final ν message bits. By avoiding the ν overhead zero-termination (ZT) bits, TBCCs can achieve higher coding rates and show improvement in frame error rate (FER) vs. E_b/N_0 performance compared to the corresponding ZTCCs. Designing efficient decoders for TBCCs has been a popular research topic [2]–[6] and the construction of minimal trellises for TBCCs has also been extensively researched [7]–[10].

In addition to tail-biting, the paradigm of concatenating a convolutional code with a cyclic redundancy check (CRC) code further improves the decoding performance and has been applied widely since its proposal in 1994 [11]. Classically, the CRC code functions as an outer error-detecting code and verifies if a codeword has been correctly received. However, when used in conjunction with list decoding [12], the CRC acts as an outer code that expurgates low-weight codewords of the inner code [5], [13]. These expurgating linear functions (ELFs) need not be cyclic, and we will follow [14] and refer to them as ELFs. Recent works using the perspective of a CC concatenated with an ELF include [15], [16], [17], and [18].

Compared to zero-terminated codes, TBCCs require a higher decoding complexity to find the maximum likelihood (ML) trellis path that also satisfies the tail-biting condition. The exhaustive approach of performing a separate Viterbi decoding for each possible beginning/ending state finds the maximum-likelihood TB codeword but requires high complexity. A preferred ML decoder is proposed in Shankar *et al.*, where the ML codeword is found through a Viterbi step followed by an A* search [3]. In contrast, the wrap-around Viterbi algorithm (WAVA) [4] has a complexity that is only three to five times that of standard Viterbi decoding and is often only slightly suboptimal. Parallel and serial list decoding are two additional approaches [5], [6], [12] that provide ML decoding of a TBCC with less complexity than the exhaustive approach or the A* search, but more complexity than WAVA. When decoding a TBCC that is concatenated with an ELF, the serial list Viterbi decoding algorithm (S-LVA) is a natural approach [5], [14], [16], where the terminating conditions require that the trellis path under consideration is both a TB codeword and a codeword of the expurgated code produced by the ELF.

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.

* The first two authors contributed equally to this work.

B. Contributions

This paper exploits the linear nature of convolutional codes [19] to reduce the complexity required to decode a TBCC concatenated with an ELF. For linear codes, the relative position of all neighboring codewords to a reference codeword is the same regardless of the reference codeword. Once a single trellis path has been found through Viterbi search, the neighborhood around that codeword can be easily accessed by simple XOR operations using a pre-computed list of offsets. This paper presents two low-complexity decoders that utilize pre-computed lists to quickly explore nearby TB codewords of a codeword found by either the Viterbi algorithm or S-LVA.

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

Our second decoder is the **list-of-spheres** decoder, which can achieve a near-ML total failure rate (TFR) while maintaining a low average list rank $\mathsf{E}[L]$ for ELF-TBCCs. TFR includes erasures as well as undetected errors. However, with

a sufficiently large list size, the erasure probability approaches zero and TFR becomes the same as the undetected error rate. This decoder performs S-LVA and searches a small sphere of tail-biting codewords around each trellis path found by the S-LVA. For decoding to terminate, the squared Euclidean metric of the codeword of the expurgated TB code must be within a threshold value. It is shown that the proposed decoder performs within 0.125 dB of the standard S-LVA decoder while maintaining a significantly smaller list size. At low E_b/N_0, the list-of-spheres decoder achieves the random coding union (RCU) bound developed by Polyanskiy *et al.* [20], which is an upper bound on the error probability of the best code given the blocklength and codeword length.

We follow [19, Chapter 11] to describe the generator matrix $G(D) = [g^{(0)}(D), g^{(1)}(D), \dots g^{(n-1)}(D)]$, where each $g^{(i)}(D)$ is a polynomial of degree up to ν in delay element D associated with the i-th code stream, i.e.,

$$g^{(i)}(D) = g_{\nu}^{(i)} D^{\nu} + g_{\nu-1}^{(i)} D^{\nu-1} + \cdots + g_0^{(i)}, \qquad (1)$$

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

C. Organization

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

II. Offset Sphere Decoding

This section considers an offset sphere decoder where the decoder examines a single large sphere of TB codewords centered on the trellis path found by the Viterbi decoder. Section II-A explains the process of generating lists of pre-computed offsets associated with the ESD of the trellis path. Section II-B shows how these pre-computed offset lists are used to find nearby TB codewords in the offset sphere decoding algorithm. Section II-C shows simulation results for offset sphere decoding on the rate-$1/2$ $(561, 753)$ ELF-TBCC with blocklength of $N = 142$ bits, and compares performance to a Viterbi decoder as well as a standard S-LVA decoder.

A. Generating Lists of Neighboring Tail-Biting Codewords

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

Consider our example ELF-TBCC with rate-$1/2$ TBCC generator polynomials $(561, 753)$ and ELF 0xFF. Using the list decoding sieve described in [14], we can perform S-LVA on the rate-$1/2$ trellis allowing any starting state and any ending state and using the noiseless all-zeroes ELF-TB codeword as the received word. The list decoder will enumerate trellis codewords in order of increasing Hamming distance from the all-zeros codeword. For each trellis codeword identified by the list decoder, we compute the ESD of that trellis codeword and append it to the list of neighboring trellis codewords associated with that ESD. The list decoding sieve continues until each list associated with an ESD has L_N neighbors.

Since the all-zeros codeword is used as the received word, the list of neighboring codewords with a particular ESD also represent the list of offsets that can be added to any trellis codeword with the same ESD to find the list of L_N closest TB codewords. As a result, the list of neighboring TB codewords can be quickly found with bit-wise XOR operations. The multi-trellis approach described in [16] to perform S-LVA would be preferred when the desired L_N is large.

B. Searching the TB Sphere for the Closest ELF-TB Codeword

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

C. Simulation Results and Discussion

Fig. 1 shows simulation results comparing a standard Viterbi decoder, the proposed offset sphere decoder with varying sizes of L_N, and an S-LVA decoder with a maximum list size of $L_{max} = 2048$ for our example rate-$1/2$ ELF-TBCC. Compared to the standard Viterbi decoder, the offset sphere decoder has a significant improvement on the TFR. The decoding performance of the offset sphere decoder improves

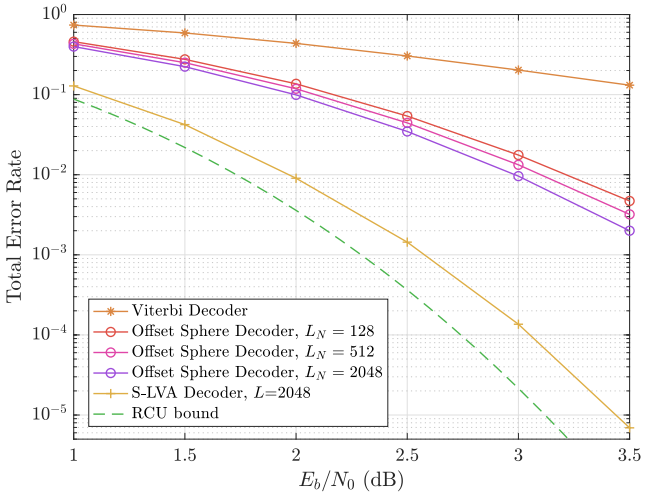

Fig. 1. TFR vs. E_b/N_0 simulation results for a Viterbi decoder, an offset sphere decoder with a varying number of neighboring codewords $L_N = 128$, 512 and 2048, as well as an S-LVA decoder with a restricted maximum list size ($L_{max} = 2048$) so we can compare across the two approaches. The RCU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

as L_N increases. For a sufficiently large L_N, performance should approach ML and therefore approach that of S-LVA with a large list size. However, the memory required to support extremely large values of L_N may make the offset sphere decoder unattractive for some applications. Fig. 1 shows that TFR for the offset sphere decoder is not comparable to S-LVA even when L_N is as large as the maximum list size of the S-LVA.

III. List-of-Spheres Decoder

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

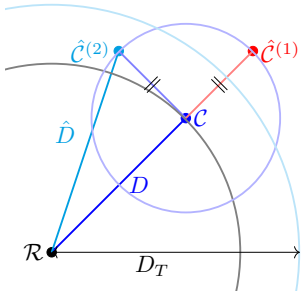

Fig. 2. Illustration of the relationship between the received word $\mathcal{R}$, a trellis codeword $\mathcal{C}$ identified by S-LVA, and two TB codewords $\hat{\mathcal{C}}^{(1)}$ and $\hat{\mathcal{C}}^{(2)}$ equidistant from $\mathcal{C}$ found through a pre-computed list of offsets corresponding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

A. Widening the Aperture of S-LVA with a List of Spheres

This section explains how the list-of-spheres decoder decreases complexity by widening the search aperture of S-LVA. Fig. 2 illustrates notation that will be useful in our discussion. The noisy received word is denoted by $\mathcal{R}$ and the trellis codeword most recently found by S-LVA is $\mathcal{C}$. As described in Sec. II-A, a list of offsets is pre-computed for each possible ESD. For each trellis codeword identified by S-LVA, the list of offsets corresponding to its ESD is used to find neighboring TB codewords.

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

$$\hat{\mathcal{C}}^{(i)} = \mathcal{C} \otimes \mathcal{N}_e^{(i)}, \tag{2}$$

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

The binary codewords are transmitted using BPSK; for each codeword bit b the level $(-E_b)^b$ is transmitted. Fig. 2 illustrates (in Euclidean space of BPSK transmission) the two TB codewords $\hat{\mathcal{C}}^{(1)}$ and $\hat{\mathcal{C}}^{(2)}$ closest to $\mathcal{C}$, which happen to be equidistant from $\mathcal{C}$.

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

In [12], Seshadri and Sundberg proposed the S-LVA, where at most L_{max} most likely codewords are found in order of increasing distance from $\mathcal{R}$. As each codeword on the list is found, it is checked to see if it meets the terminating condition, which in [12] was passing a CRC.

When S-LVA is applied to an ELF-TBCC, many of the trellis codewords on the list are not even TB codewords. The

Algorithm 1 List-of-Spheres Decoding Algorithm

Input: $\mathcal{R}$, $N_{neighbor}$, A, L_{max}
Output: Decoded codeword $\mathcal{C}^*$

```
while L < L_max do
    Identify C by S-LVA
    if C is an ELF-TB codeword then
        Return C* = C
    else
        Calculate ESD e of C
        Compute Ĉ using N_e
        if ∃ Ĉ^(i) ∈ Ĉ that is an ELF-TB codeword then
            Compute D̂^(i) ≜ ||R − Ĉ^(i)||_2
            if D̂^(i) ≤ D_T and D̂^(i) < D̂^(j) ∀j ≠ i then
                Return C* = Ĉ^(i)
            end if
        end if
    end if
end while
```

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

B. The Size of the Spheres

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

C. A Threshold to Avoid Decoding Errors

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

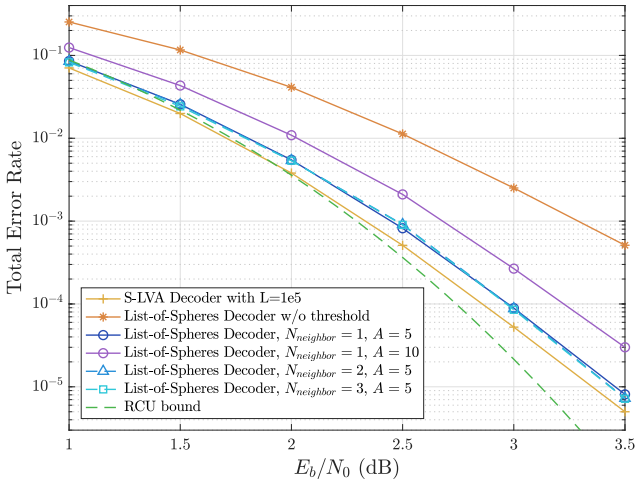

Fig. 3. TFR vs. E_b/N_0 simulation results for the same ELF-TBCC as Fig. 1 using the list-of-spheres decoder for aperture parameters $A = 10$ with $N_{neighbor} = 1$ and $A = 5$ with $N_{neighbor} \in \{1, 2, 3\}$. The performance of a list-of-spheres decoder without any threshold and an S-LVA decoder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

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

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

The aperture parameter A prevents the decoder from selecting neighboring TB codewords that are too far away from the received word $\mathcal{R}$. This restriction reduces the probability of selecting a non-ML decoding result. When $A = 0$, the list-of-spheres decoder is equivalent to a regular S-LVA decoder. As A is increased and the "aperture" of our decoder opens, more TB codewords in the sphere become admissible. Thus, the expected list rank and decoding complexity are reduced because decoding terminates sooner, but non-ML codewords are more likely to be selected.

Fig. 3 shows simulated TFR vs. E_b/N_0 result for a list-of-spheres decoder without a threshold requirement, which is far off the RCU bound. The TFR performance improves considerably when the threshold requirement is added with an aperture parameter of $A = 10$. Even better TFR performance, within 0.125 dB of S-LVA, is achieved with an aperture

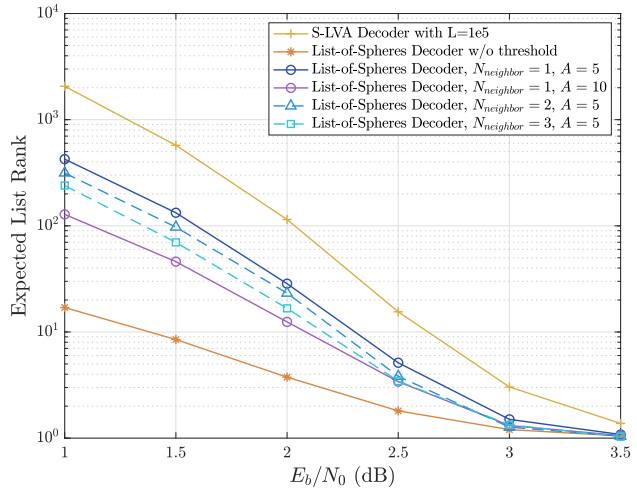

Fig. 4. $\mathsf{E}[L]$ vs. E_b/N_0 simulation results for the same ELF-TBCC as Fig. 1 using the list-of-spheres decoder for aperture parameters $A = 10$ and $A = 5$ and $N_{neighbor} \in \{1, 2, 3\}$. The $\mathsf{E}[L]$ of a list-of-spheres decoder without any threshold and an S-LVA decoder with a sufficiently large maximum list size ($L_{max} = 10^5$) that ensures ML decoding are shown in solid yellow and orange, respectively.

parameter of $A = 5$ at a target TFR of 10^{-5}. At low E_b/N_0, the performance of a list-of-spheres decoder with $A = 5$ achieves the RCU bound.

To consider possible choices for $N_{neighbor}$, Fig. 3 shows simulation results for a list-of-spheres decoder with a fixed aperture parameter $A = 5$ and varying $N_{neighbor}$ values of 1, 2, and 3, as well as an S-LVA decoder with a sufficiently large list size ($L_{max} = 10^5$) that ensures ML decoding. With the aperture parameter preventing most selections of non-ML codewords, increasing $N_{neighbor}$ does not degrade TFR performance. However, as shown in Fig. 4, increasing $N_{neighbor}$ significantly reduces the expected list rank. For $A = 5$, increasing to $N_{neighbor} = 3$ reduces the expected list rank to become close to that of $A = 10$ with $N_{neighbor} = 1$.

E. Expected List Rank and Complexity

In [5], the authors provided the complexity expression for S-LVA of rate-$1/\omega$ ELF-TBCCs, where the overall average complexity of S-LVA can be decomposed into three components:

$$C_{\text{SLVA}} = C_{\text{SSV}} + C_{\text{trace}} + C_{\text{list}}. \tag{3}$$

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

For ELF-TBCCs, these metrics are evaluated by Eq. 4 through Eq. 7, where $\mathsf{E}[I]$ is the expected number of insertions to maintain the sorted list of path metric differences.

$$C_{\text{SSV}} = 1.5(K+m)2^{v+1} + 2^v + 3.5(K+m) \tag{4}$$

$$C_{\text{trace}} = 3.5(\mathsf{E}[L]-1)(K+m) \tag{5}$$

$$C_{\text{list}} = \mathsf{E}[I]\log(\mathsf{E}[I]) \tag{6}$$

$$\mathsf{E}[I] \le (K+m)\mathsf{E}[L] + 2^v - 1 \tag{7}$$

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

In Fig. 4, the expected list ranks of a list-of-spheres decoder with varying $N_{neighbor}$ and aperture values are compared to that of a standard S-LVA decoder that achieves ML decoding. At low E_b/N_0, as the threshold increases, $\mathsf{E}[L]$ of the list-of-spheres decoder decreases substantially. The expected list rank of a list-of-spheres decoder without a threshold is around $\frac{1}{10}$ of those with thresholds, but this complexity reduction comes at the price of poor decoding performance. While $A = 10$ achieves the second lowest $\mathsf{E}[L]$, the list-of-spheres decoder with $A = 5$ still shows a much lower complexity than the S-LVA decoder, providing a considerable reduction in decoding complexity.

Fig. 4 also compares the expected list rank of list-of-spheres decoders with varying $N_{neighbor}$ and a fixed aperture $A = 5$. Despite their similar TFR performance, the decoders show how $\mathsf{E}[L]$ decreases with increasing values of $N_{neighbor}$, especially at low E_b/N_0. Including more neighboring codewords reduces $\mathsf{E}[L]$ because the decoder is able to find an ELF-TB codeword earlier by searching a larger sphere around a trellis codeword.

IV. Conclusion

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

REFERENCES

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

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

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

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

[5] 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*, Feb. 2022, early access. [Online]. Available: https://doi.org/10.1109/TIT.2022.3150717

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[20] 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.