Efficient Maximum-Likelihood Decoding for TBCC and CRC-TBCC Codes via Parallel List Viterbi

Jacob King*, William Ryan, Chester Hulse*, and Richard D. Wesel*
*Department of Electrical and Computer Engineering, University of California, Los Angeles, Los Angeles, CA 90095, USA
Email: jacob.king@ucla.edu, ryan@ece.arizona.edu, chulse@ucla.edu, wesel@ucla.edu

***Abstract*—Maximum-likelihood (ML) decoding of tail-biting convolutional codes (TBCCs) with $S = 2^v$ states traditionally requires a separate S-state trellis for each of the S possible starting/ending states, resulting in complexity proportional to S^2. Lower-complexity ML decoders for TBCCs have complexity proportional to $S \log S$. This high complexity motivates the use of the wrap-around Viterbi algorithm, which sacrifices ML performance for complexity proportional to S.**

This paper presents an ML decoder for TBCCs that uses list decoding to achieve an average complexity proportional to S at operational signal-to-noise ratios where the expected list size is close to one. The new decoder uses parallel list Viterbi decoding with a progressively growing list size operating on a single S-state trellis. Decoding does not terminate until the most likely tail-biting codeword has been identified. This approach is extended to ML decoding of tail-biting convolutional codes concatenated with a cyclic redundancy check code as explored recently by Yang et al. and King et al. Constraining the maximum list size further reduces complexity but sacrifices guaranteed ML performance, increasing errors and introducing erasures.

***Index Terms*—Tail-Biting Convolutional Codes, Convolutional Codes, Cyclic Redundancy Check, List Viterbi Decoding, List Decoding**

I. Introduction

Tail-biting convolutional codes (TBCCs) [1] have been shown to improve over the performance of zero-terminated convolutional codes (ZTCCs) because they avoid the rate loss of zero-termination. A well-studied problem is the design of computationally efficient decoders for TBCCs that achieve good frame error rate (FER) performance. Many efficient decoders have been proposed for TBCCs with $O(S)$ complexity, including WAVA [2] and the approximate maximum-likelihood (ML) linear-time decoding algorithm [3]. However, while these algorithms are desirable from a complexity standpoint, they do not achieve ML performance.

For a TBCC with $S = 2^v$ states, a brute force ML decoder implementation includes S separate Viterbi decoders, one for each of the S tail-biting sub-trellises. This algorithm has a complexity of $O(S^2)$, which is too complex for practical use. Shankar *et al.* [4], [5] have proposed a more efficient ML decoder for TBCCs with complexity $O(S \log S)$, which consists of a Viterbi step followed by an A* search to find the ML codeword. This algorithm was later improved by Pai *et al.* [6]. However, these algorithms are still significantly more complex than the suboptimal decoders with $O(S)$ complexity.

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 authors and do not necessarily reflect views of NSF.

Seshadri and Sundberg proposed parallel and serial list Viterbi algorithm (LVA) decoders in [7]. This paper modifies the parallel LVA decoder to achieve ML decoding performance of TBCCs and concatenated CRC-TBCCs with a conjectured $O(S)$ average complexity at operational signal-to-noise ratios where the expected list size is close to one.

A. Contributions

In this paper, we present an algorithm for ML decoding of TBCCs with average complexity that scales linearly with the number of states, and we prove that it achieves ML decoding. We also show how adding an additional step to this algorithm can also achieve ML decoding performance for concatenated CRC-TBCCs. We compare the FER performance of our algorithm to WAVA [2] and the WAVA-inspired adaptive parallel list Viterbi decoder (WI-APLVD) in [8] for selected TBCCs and CRC-TBCCs, respectively. We also compare the average complexity of WAVA and the new algorithm.

B. Organization

Section II reviews TBCCs and the challenges in designing good decoders for them. Section III gives an overview of parallel list Viterbi decoding (PLVD), as well as how a modification to a standard PLVD algorithm can be made to achieve ML decoding performance for TBCCs. Section IV shows how to generalize this ML decoder to CRC-TBCC concatenated codes, and details a sub-optimal decoder for CRC-TBCCs that is shown to have good performance in [9] [8]. Finally, Section V shows simulation results comparing FER performance and decoding complexity for our proposed ML decoder and several sub-optimal decoders, for both TBCCs and CRC-TBCCs.

II. Tail-biting Convolutional Codes

Convolutional codes, first introduced by Elias [10], have found wide application. There are two main classes of feedforward convolutional block codes: zero-terminated convolutional codes (ZTCCs), and tail-biting convolutional codes (TBCCs) [1]. Tail-biting convolutional codes use the final v message bits to initialize the state of the convolutional encoder so that the beginning state is the same as the final state. This ensures that all message bits enjoy the same protection from channel errors while avoiding the rate penalty incurred by zero-termination.

Compared to ZTCCS, TBCCs require additional decoding complexity to ensure that the selected codeword satisfies the tail-biting (TB)-condition. The TBCC decoder must either separately consider candidate codewords for every possible starting/ending state or determine (or guess) the starting/ending state of the ML path. The rate penalty of zero-termination is significant for short messages, and TBCCs have been shown to outperform ZTCCs in FER vs. E_b/N_0 performance [11]. Thus, low-complexity decoders for TBCCs are of practical interest. The following section reviews PLVD for TBCCs and introduces the proposed ML decoder.

III. ML Decoding via Parallel List Viterbi

A. Parallel list Viterbi decoding

Seshadri and Sundberg proposed an efficient PLVD algorithm in [7]. For each state in the trellis, instead of selecting a single best path, the PLVD keeps a metric-ordered list of the L best paths arriving at each state. Once the end of the trellis is reached, each ending state has a list of L best trellis paths to each state. This produces an overall list of SL trellis paths. The valid codeword with the best metric is selected.

For a TBCC, PLVD selects from the list of SL survivors the survivor with the best metric that also satisfies the TB condition. Note that some paths with better metrics than the SL surviving paths may be excluded from the list of survivors. Thus, the best TB path found on the list of survivors might not be the ML decoding decision.

B. Adaptive Parallel List Viterbi Decoding

If L is large, PLVD has high average complexity due to computing SL trellis paths before checking if any of them are valid codewords. The SL paths are organized as a set $\{\mathcal{L}_\sigma\}$ of S lists $\mathcal{L}_\sigma$, one for each possbile ending state. We can reduce the average computational complexity of the PLVD by using an adaptive PLVD (APLVD). The APLVD begins by initializing the list size $L = L_{min}$, where L_{min} is usually 1. PLVD is run with list size L. If a TB codeword is found by the PLVD, then that codeword is selected and the decoder terminates. Otherwise, L is doubled and PLVD is run again.

Yang *et. al.* [11] shows that for serial LVA decoders, the expected list rank of the decoded codeword converges to 1 as SNR increases, and is typically close to one at operating SNRs where the FER is acceptable. A PLVD needs a list size of at most γ to find a codeword of serial list rank γ. Thus, for typical SNR operating points the APLVD terminates with an average list size close to one. At such operating points, the average time and space complexity of APLVD is comparable to SLVD since both have average complexity similar to the standard Viterbi algorithm.

As stated above, the (A)PLVD is not an ML decoder for TBCCs, since the list of SL trellis paths it generates are not guaranteed to be the SL most likely paths. In the next section, we show that a modification to the APLVD algorithm can achieve ML decoding of TBCCs and CRC-TBCCs without incurring much additional complexity. An APLVD with this modification was originally proposed in [12], where it is called a resolution-terminated adaptive parallel list Viterbi decoder (RT-APLVD). We will continue to use this name for the remainder of this paper.

C. Description of RT-APLVD

We will first define some terminology. A *trellis path* (or simply *path*) is any sequence of states from the beginning of the trellis to the end. A *tail-biting (TB) path* is a trellis path such that the starting state and ending state are the same, and a *codeword* is the encoded output generated by a TB path. We will use TB path and codeword interchangeably.

The *candidate codeword* $\hat{C}$ is the codeword with the best metric that the RT-APLVD decoder has seen up to that point and the *candidate metric* $\hat{M}$ is the metric of this codeword. There is always at most one candidate codeword. The *state metric* M_σ of a given end state σ is the metric of the best TB path for that state, or if no TB path has been found at this end state with list size L, it is the best possible metric that a TB path for that state could have, i.e. the worst metric observed so far at this end state. The *state path* P_σ is the path that has the state metric M_σ.

Each end state is classified as either *resolved* or *unresolved*. A state is resolved either when the most likely codeword that ends in that state is found or when it is known that the ML codeword cannot end in that state. Every state begins unresolved, and the decoder will switch each state to resolved during the decoding process. We prove in Section III.D that once all of the states are resolved, the candidate codeword must be the ML codeword and the decoder can terminate.

The RT-APLVD algorithm is detailed in Algorithms 1, 2, and 3. It uses the **PLVD**(.) function as described in [7], except that resolved states are removed from the set of initial states and end states for which paths are listed. In Alg. 1, the list size L doubles after every iteration. An alternative would be to maintain the same L for multiple iterations doubling L once no additional states become resolved. Comparing the complexity of these alternatives is an area of future interest.

D. Proof of ML Decoding and Complexity Analysis

Lemma 1. *The PLVD algorithm with list size L identifies the L most likely trellis paths that end at each state σ.*

Proof: See Seshadri and Sundberg [7]. ■

Theorem 1. *RT-APLVD is an ML decoder for TBCCs.*

Proof of Theorem 1: By Lemma 1, executing PLVD with a list size L identifies the L most likely (not necessarily tail-biting) trellis paths for each ending state. Thus, traversing the list for a given end state in order starting from the most likely path, the first TB path encountered is the most likely TB path that ends at this state. Thus, one ML decoding algorithm for TBCCs is to repeatedly run PLVD with increasing list sizes until the most likely TB path for each end state is found. The most likely TB path among this group must be the ML codeword. The RT-APLVD is a modification to the above algorithm that reduces complexity by allowing termination before every end state has identified a TB codeword.

Algorithm 1: RT-APLVD

```
Input: $L_{min}$, $L_{max}$, channel outputs
Result: Decoded Codeword $c$
$L \leftarrow L_{min}$
$(\hat{C}, \hat{M}) \leftarrow (\{\}, \infty)$
$S^* \leftarrow \{\}$
while $L \leq L_{max}$ and $S \backslash S^* \neq \{\}$ do
    $\{\mathcal{L}_\sigma\} \leftarrow$ PLVD$(L, S \backslash S^*)$
    $(S^*, \{(P_\sigma, M_\sigma)\}) \leftarrow$
      StateMetricCalculation$(S^*, \{\mathcal{L}_\sigma\})$
    $(\hat{C}, \hat{M}, S^*) \leftarrow$
      CandidateSelection$(\{(P_\sigma, M_\sigma)\}, \hat{C}, \hat{M}, S^*)$
    $L \leftarrow 2L$
end
$c \leftarrow \hat{C}$
```

Algorithm 2:
$(S^*, \{(P_\sigma, M_\sigma)\}) =$ **StateMetricCalculation**$(S^*, \{\mathcal{L}_\sigma\})$

```
Input: Set of resolved states $S^*$, Set of Lists $\{\mathcal{L}_\sigma\}$,
       one for each end state, of paths and metrics
       sorted by metric.
Output: $S^*$, $\{(P_\sigma, M_\sigma)\}$
for $s = \sigma_0, \ldots, \sigma_{S-1}$ do
    if $s \in S^*$ then
        $(P_s, M_s) \leftarrow (\{\}, \infty)$
    else
        for $p \in \mathcal{L}_s$ do
            if $p$ is TB then
                $(P_s, M_s) \leftarrow (p, \text{metric}(p))$
                $S^* \leftarrow S^* \cup s$
                break
            end
            else if $p$ is Lth path then
                $(P_s, M_s) \leftarrow (p, \text{metric}(p))$
            end
        end
    end
end
```

Algorithm 3:
$(\hat{C}, \hat{M}, S^*) =$ **CandidateSelection**$(\{(P_\sigma, M_\sigma)\}, \hat{C}, \hat{M}, S^*)$

```
Input: $\{(P_\sigma, M_\sigma)\}$, $\hat{C}, \hat{M}$, $S^*$
Output: $\hat{C}, \hat{M}$, $S^*$
for $s = \sigma_0, \ldots, \sigma_{S-1}$ do
    if $M_s < \hat{M}$ and $P_s$ is TB then
        $(\hat{C}, \hat{M}) \leftarrow (P_s, M_s)$
    end
end
for $s = \sigma_0, \ldots, \sigma_{S-1}$ do
    if $M_s > \hat{M}$ then
        $S^* \leftarrow S^* \cup s$
    end
end
```

For a specified list size L, RT-APLVD identifies a candidate codeword $\hat{C}$, which is the codeword that has the best metric from among the codewords that have been found. RT-APLVD concludes that $\hat{C}$ is the ML codeword when all end states are *resolved*. A state becomes resolved in either of two ways. Firstly, the state can become resolved if its best TB path is found. Secondly, the state can become resolved before its best TB path is found if it becomes clear that best TB path is less likely than $\hat{C}$. These two conditions combined imply that once all states are resolved, the candidate codeword is the most likely codeword among the set of most likely codewords for each end state, i.e., it is the ML codeword. ■

The proof above shows that if RT-APLVD doesn't terminate until all states are resolved, then it is an ML decoder for TBCCs and CRC-TBCCs. This raises the question of whether all states will eventually resolve.

Yang [11] shows that when decoding TBCCs or CRC-TBCCs with a serial list Viterbi decoder (SLVD), there exists a *supremum list rank* λ such that a codeword is guaranteed to be found with a list rank less than or equal λ. By an identical argument as in [11], each individual end state of the RT-APLVD algorithm has a supremum list rank λ_σ by which a TB path is guaranteed to be found. Thus, there must exist a finite supremum $L_{max} = \max \lambda_\sigma$ for which RT-APLVD will always terminate with every state resolved, and thus there exists a finite supremum L_{max} for which RT-APLVD is an ML decoder for TBCCs and CRC-TBCCs. Finding concrete bounds on this supremum L_{max} is a question of future interest.

IV. Decoding Concatenated CRC-TBCC codes

A. Maximum Likelihood Parallel List Viterbi Decoding

The RT-APLVD algorithm is easily adapted to concatenated CRC-TBCC codes as explored by, e.g., [9], [11]. Instead of only checking that a trellis path is a TB path, the adapted RT-APLVD algorithm also checks if it passes the CRC check. Only if a path passes both checks is it validated as a codeword. Given the extra CRC-passing requirement, RT-APLVD typically requires larger list sizes for ML decoding of CRC-TBCCs compared to TBCCs with the same number of states but no CRC.

B. WAVA Inspired Adaptive Parallel List Viterbi Decoding

RT-APLVD provides an ML decoder for a sufficiently large list size, but for smaller list sizes a WAVA-inspired modification of APLVD can be competitive in both performance and complexity. WAVA selects a tail-biting codeword with low complexity because multiple passes through the trellis rapidly coalesce toward a matching start/end state. We propose a WAVA-inspired (WI)-APLVD that performs a single pass through the trellis with a list size of 1 to find the metrics of all of the ending states. We then use these metrics as the initial metrics for the APLVD, terminating as soon as a codeword that passes both the tail-biting and CRC checks is found. This decoder has been used for the decoding of CRC-TBCCs by King et al. in [9] [8]. While not ML, it achieves near-ML performance with low complexity, as shown in Sec. V.

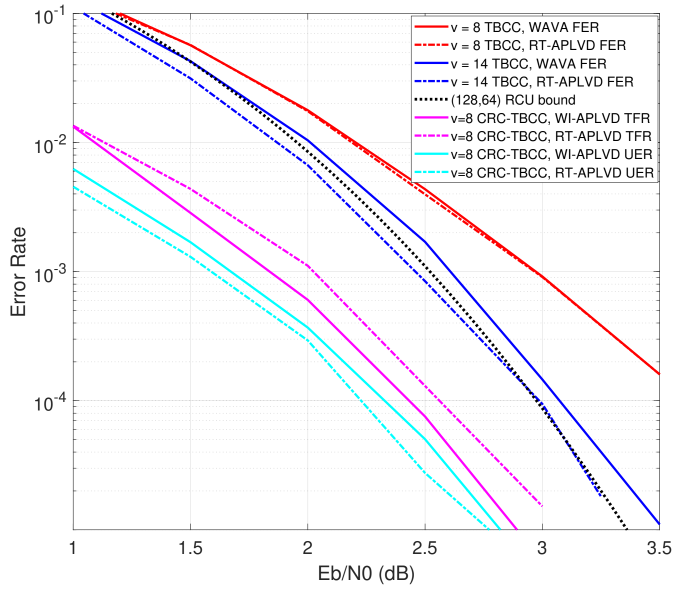

Fig. 1. FER curves of $v = 8$ and $v = 14$ TBCCs and $v = 8$ CRC-TBCC. RT-APLVD outperforms WAVA for $v = 11$ and $v = 14$ TBCCs, and beats the RCU bound for the $v = 14$ TBCC. RT-APLVD performs worse than WI-APLVD for the CRC-TBCC due to constraining L_{max} to 1024. However, RT-APLVD has a lower undetected error rate than WI-APLVD.

TABLE I
TABLE OF RATE-64/128 TBCCS FROM [13]. POLYNOMIALS ARE IN OCTAL.

v	$g_1(x)$	$g_2(x)$
8	515	677
11	5537	6131
14	75063	56711

We considered applying to the WI-APLVD decoder the more stringent stopping criterion of RT-APLVD to guarantee that the codeword with the best metric is found. However, this resulted in a degradation in FER for this decoder, perhaps because the WAVA step results in a non-ML metric.

V. SIMULATION RESULTS

We now show simulation results for FER of selected TBCC and CRC-TBCC codes, as well as simulation-assisted average complexity calculations of our decoders. This section compares the performance of RT-APLVD and WAVA on three selected rate-1/2 TBCCs from [13]. The generator polynomials for these TBCCs are given in Table I. We also compare the performance of RT-APLVD and WI-APLVD on a rate-32/512 CRC-TBCC from [14]. This code is an 11-bit outer CRC code with polynomial 0xF69, concatenated with a rate-1/12 inner convolutional code with polynomials $\{533, 727, 765, 445, 715, 635, 563, 555, 737, 557, 677, 511\}$ and punctured bits in positions $\{47, 60, 129, 504\}$. This code was originally studied in [14] to compare against a CRC-polar code compatible with the 5G standard.

A. Frame Error Rate Performance

For TBCCs, Fig. 1 shows the FER performance vs. E_b/N_0 for WAVA and RT-PLVD decoders for $v = 8$ and $v = 14$ as well as the RCU bound. For the TBCCs, WAVA and RT-APLVD have nearly identical performance for the $v = 8$ TBCC. However, RT-APLVD has noticeable improvement over WAVA for the $v = 14$ TBCCs. For the $v = 11$ code (not shown), RT-APLVD also provided a noticeable improvement over WAVA. The $v = 14$ TBCC decided with RT-APLVD has an FER below the RCU bound until around 3 dB.

For the CRC-TBCCs, Fig. 1 shows the total failure rate (TFR) (including both undetected errors and erasures) and undetected error rate (UER) (not including erasures) vs. E_b/N_0 for RT-APLVD and WI-APLVD. For these simulations the maximum list size was restricted to $L_{max} = 1024$, which is not large enough for ML decoding of this CRC-TBCC. As a result, the WI-APLVD has better FER performance than the RT-APLVD. However, the RT-APLVD has a lower UER compared to WI-APLVD. Once L_{max} is sufficiently large to provide ML decoding, RT-APLVD would have no erasures and a lower TFR than WI-APLVD, but the list size necessary to achieve this is likely prohibitively large.

B. Expected Complexity

This section analyzes the complexities of each of the decoders, found though a combination of mathematical analysis and simulation. Following the complexity calculations in [11], we assign one unit of complexity for each branch addition, and one unit of complexity for each comparison of two values.

The calculated complexity values depend on the average number of trellis passes for WAVA, the distribution of how often WI-APLVD and RT-APLVD terminate at each list size, and the proportion of unresolved states at each list size for RT-APLVD. All of these values are intractable to calculate analytically, so we found them through simulation. We restricted L_{max} to 8192 for these simulations due to technical limitations.

Even with using simulation to calculate these values, there are still a couple simplifying assumptions used in the calculation of these average complexities. However, our results for average complexity generate curves consistent with average decoding runtimes found through simulation.

The full derivation of expected complexity for each decoder cannot be included in this paper due to space limitations. We give the final expected complexities of each of the decoders. In these equations, $\mathsf{E}[n_{pass}]$ represents the expected number of trellis passes performed by WAVA before terminating, L represents the list size of the adaptive decoders when terminating, p_{2^j} represents the proportion of starting states that are unresolved with a list size 2^j, $\log$ refers to $\log_2$, and $\ln$ refers to $\log_e$ (i.e. the natural logarithm).

$$C_{WAVA} = 3KS\mathsf{E}[n_{pass}] + S + (\mathsf{E}[n_{pass}] - 1)(S + 1) + 2K$$

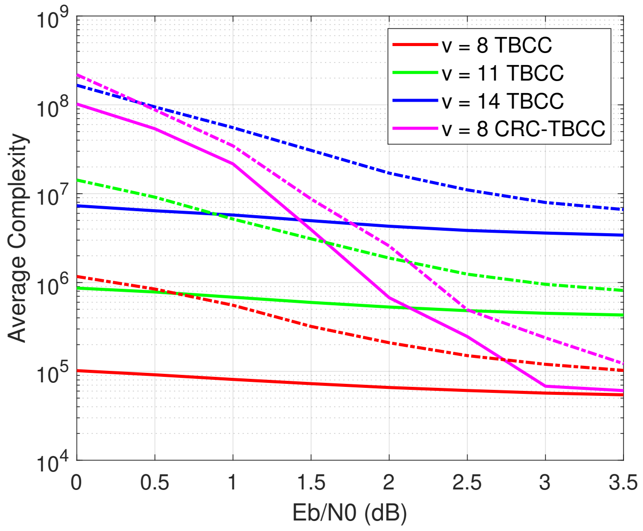

Fig. 2. Complexities C_{RT} (dashed) and C_{WAVA} (solid) vs. E_b/N_0 for TBCCs and Complexities C_{RT} (dashed) and C_{WI} (solid) vs. E_b/N_0 for the $v = 8$ CRC-TBCC.

$$\begin{aligned}C_{WI} = \sum_{k=0}^{\log L_{max}} & Pr[L = 2^k] * \Big(2S(2L - \log L - 2) \\ & + 3KS + 2K + (S+1)(2L-1) \\ & + (2L-1)(2KS - 4S) \\ & + (8L \log L + 4)\left(\frac{KS - 2S}{\log e}\right)\Big)\end{aligned}$$

$$\begin{aligned}C_{RT} \approx \sum_{k=0}^{\log L_{max}} & Pr[L = 2^k] * \Big(2S(2L-1) \\ & + 2K + (S+1)\left[\sum_{j=0}^{k} (2^j * p_{2^j})\right] \\ & + S\left[\sum_{j=0}^{k} 2^{j+1}\left(K - \left\lceil \log\left(\frac{2^j}{p_{2^j}}\right)\right\rceil\right)(1 + 2\ln(2^{j+1}))\right]\Big)\end{aligned}$$

Fig. 2 shows the expected complexity of each decoder on each code as a function of E_b/N_0. We see that RT-APLVD tends to be more complex than WAVA and WI-APLVD, as would be expected. However, as E_b/N_0 increases, the gap between RT-APLVD and the corresponding sub-optimal decoder tends to decrease. In addition, the complexity curves appear to scale linearly with S.

VI. Conclusion

In this paper, we presented an efficient maximum likelihood decoder for TBCCs and concatenated CRC-TBCCs. We motivate the design of this algorithm and show both its ML nature and its desirable complexity.

Simulations show that the RT-APLVD outperforms the WAVA decoder in terms of FER for TBCCs with large memory, although WAVA is still competitive. We also show that RT-APLVD decoding of a $v = 14$ TBCC in [13] slightly beats the RCU bound up to around $E_b/N_0 = 3dB$. WAVA is less complex than RT-APLVD due to its less stringent stopping condition and its list size of one for the Viterbi step. However, the RT-APLVD decoder is not prohibitively complex and can be used for practical situations in the case where ML decoding is desired. Further optimization can likely reduce the complexity of RT-APLVD.

For CRC-TBCC concatenated codes, the RT-APLVD is much better at filtering out undetected errors compared to the WI-APLVD; however, it often requires a very large list size to avoid erasures. Further analysis in this area is needed.

Our calculations for complexity on RT-APLVD imply that the complexity is $O(S)$ supposing that $\mathsf{E}[L]$ and p_{2^j} do not increase with increasing S. Our simulations show a very slight increase in these values as we varied S for our TBCCs, but the sample size of our simulations is not large enough to definitively disprove statistical variance given how small the change is. Even if the complexity is slightly worse than $O(S)$ asymptotically, the complexity is effectively linear for blocklengths, trellis sizes, and SNR ranges of practical interest.

References

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

[2] R. Y. Shao, S. Lin, and M. P. C. Fossorier, “Two decoding algorithms for tail-biting codes,” *IEEE Trans. Commun.*, vol. 51, no. 10, pp. 1658–1665, Oct 2003.

[3] K. M. Krishnan and P. Shankar, “Approximate linear time ML decoding on tail-biting trellises in two rounds,” in *2006 IEEE Inter. Sym. on Info. Theory (ISIT)*, Jul 2006, pp. 1–5.

[4] P. Shankar, P. N. A. Kumar, K. Sasidharan, and B. S. Rajan, “ML decoding of block codes on their tailbiting trellises,” in *Proc. 2001 IEEE Inter. Sym. on Info. Theory (ISIT)*, Jun 2001, p. 1.

[5] P. Shankar, P. N. A. Kumar, K. Sasidharan, B. S. Rajan, and A. S. Madhu, “Efficient convergent maximum likelihood decoding on tail-biting trellises.” [Online]. Available: https://arxiv.org/abs/cs/0601023

[6] H.-T. Pai, Y. S. Han, T.-Y. Wu, P. ning Chen, and S.-L. Shieh, “Low-complexity ML decoding for convolutional tail-biting codes,” *IEEE Commun. Letters*, vol. 12, no. 12, pp. 883–885, Dec 2008.

[7] 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.

[8] J. King, “CRC-aided list decoding of short convolutional and polar codes for binary and nonbinary signaling,” Master’s thesis, University of California, Los Angeles (UCLA), 2022.

[9] J. King, A. Kwon, H. Yang, W. Ryan, and R. D. Wesel, “CRC-aided list decoding of convolutional and polar codes for short messages in 5G,” in *2022 IEEE Int. Conf. on Commun. (ICC)*, May 2022, pp. 1–6.

[10] P. Elias, “Coding for noisy channels,” *Proc. IRE Conv. Rec. part 4*, pp. 37–46, 1955.

[11] 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, Jun 2022.

[12] C. Hulse, “FPGA implementation of decoders for CRC-aided tail-biting convolutional codes,” Master’s thesis, University of California, Los Angeles (UCLA), 2022.

[13] M. C. Coşkun, G. Durisi, T. Jerkovits, G. Liva, W. Ryan, B. Stein, and F. Steiner, “Efficient error-correcting codes in the short blocklength regime.” [Online]. Available: https://arxiv.org/abs/1812.08562

[14] J. King, H. Yao, W. Ryan, and R. D. Wesel, “Design, performance, and complexity of CRC-aided list decoding of convolutional and polar codes for short messages.” [Online]. Available: https://arxiv.org/abs/2302.07513