Efficient decoder-free minimum distance estimation for concatenated convolutional codes

Mohammad Bazzal, Jérémy Nadal, Stefan Weithoffer, Charbel Abdel Nour, Catherine Douillard

Efficient decoder-free minimum distance estimation for concatenated convolutional codes

Mohammad Bazzal, Jeremy Nadal, Stefan Weithoffer, Charbel Abdel Nour, Catherine Douillard
IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France

***Abstract*—This work presents a novel approach for estimating the minimum distance of concatenated convolutional codes. Unlike most prior methods, the proposed approach relies solely on encoder-side properties, eliminating the need for a decoder in the estimation process. To this end, we introduce an alternative method for identifying return-to-zero sequences, enabling a low-complexity, reliable minimum distance estimation. The proposed method is flexible and applicable to both serial and parallel concatenated structures. The results confirm that this approach produces exact minimum distance values for several sizes of the LTE turbo code family while significantly reducing processing time and computational complexity compared to existing methods. For example, in a challenging case study, applying the method to an LTE turbo code with tailbiting termination of length $K = 6144$ bits yields an estimated minimum distance of 50 and a multiplicity of 12288 in 100 minutes on a standard personal computer - a reduction by several orders of magnitude in comparison to state of the art methods.**

***Index Terms*—Concatenated convolutional codes, minimum distance, return-to-zero sequences.**

I. Introduction

Designing efficient error-correcting codes requires a thorough understanding of both design and analysis tools. Despite their introduction over three decades ago, coding structures based on the concatenation of convolutional codes (CCs) remain challenging to fully analyze and optimize. A primary challenge lies in understanding the effects of interleaving, a key component that significantly impacts error-correction performance by determining the minimum Hamming distance (mHd). Calculating the mHd in these structures is particularly complex due to the pseudo-random connections introduced by the interleaver, which make brute-force computation infeasible for practical frame sizes. This issue has been addressed extensively in the context of turbo codes (TCs). This code family continues to be widely used, including in evolutions of LTE standards such as LTE Advanced Pro [1]. TCs offer good error-correction performance along with inherent rate flexibility [2]. To this regard, an algorithm proposed in [3] provides a means to calculate the initial terms of a TC's distance spectrum for a given interleaver. While efficient for short interleavers and low-mHD cases, this algorithm becomes impractical for longer interleavers due to its high computational complexity.

Approximate distance estimation methods using iterative decoding techniques have also been proposed. Among these, the error impulse method (EIM) [4] provides reasonably efficient and accurate estimates for low-mHD cases. However, as demonstrated in [5], [6], the method becomes unreliable for interleavers that yield high mHDs, often leading to significant underestimations of the actual distance. Other iterative approaches, including the single impulse method (SIM), double impulse method (DIM), triple impulse method (TIM), and event impulse method (EVIM) [5]–[7], also face similar issues. These techniques rely on iterative decoding processes that not only introduce computational overhead but provide inaccurate estimates. This inaccuracy arises for decoding steps that are typically performed through sub-optimal iterative decoders, which can fail to produce actual codewords.

A key insight is that the mHD of concatenated codes is independent of the decoding process, which opens the door for alternative estimation techniques based directly on code structure or encoder properties. This work focuses on developing such techniques.

The remainder of this paper is organized as follows: Section II provides an overview of the main estimation methods based on the insertion of one or more error impulses, including a detailed analysis of their computational complexity. Section III explores key mathematical properties of return-to-zero (RTZ) sequences of CCs and introduces a novel graph-based approach for estimating the mHD of concatenated CCs. Section IV presents a comparative evaluation of our algorithm against TIM on quadratic permutation polynomial (QPP) interleavers. Finally, conclusions are drawn in Section V.

II. Related Work

The first distance estimation method, called EIM, was introduced by Berrou et al. [4]. The EIM method assesses the error correction capability of a turbo decoder by inserting an error impulse into an all-zero codeword at a specific data index. The amplitude of this impulse is gradually increased until the decoder fails to correct it, which happens when the mHD is reached, assuming maximum-likelihood (ML) decoding.

Garello et al. [5] then proposed the SIM method, also known as the "all-zero iterative decoding algorithm". Similar to EIM, it injects a single impulse at a specific data index in the all-zero codeword. However, unlike EIM, the impulse amplitude is sufficiently high that the decoder cannot correct it and converges to a non-zero codeword. This data pattern is then re-encoded to generate a corresponding non-zero codeword, which has been shown to yield the true mHD, $d_{\min}$, under the assumption of ML decoding. The complexity of SIM is $O(K^2\xi)$, where K is the information frame size, and ξ represents the number of decoding iterations.

In [6], the DIM method was introduced, which places a first impulse at the tested data index and a second impulse within

a specified range, τ, of the first one. Although more complex than SIM, with a complexity of $O(K^2\xi\tau)$, DIM provides greater accuracy in estimating mHD $d_{\min}$ when $d_{\min}$ is large.

The TIM method, also presented in [6], applies three strong impulses to encourage the turbo decoder to converge towards a non-zero data pattern. TIM's complexity is $O(K^2\xi\tau^2)$, offering slightly improved accuracy over DIM.

In contrast to the previous impulse methods, the EVIM method [7] targets specific low-weight error events rather than individual data bits. EVIM injects impulses at all occurrences of data '1' within a selected error event, allowing for a dynamic number of impulses depending on the event characteristics. The efficiency of this approach relies on the ability to populate a limited-size list with events that lead to the mHD, which is particularly challenging as parameter K increases. The computational complexity of EVIM is $O(K^2\xi V)$, where V represents the total number of considered events.

Although these methods were developed primarily for parallel concatenated structures, they are directly extendable to the serial concatenation of CCs.

III. Towards Decoder-Free Distance Estimation

Prior-art methods for minimum distance estimation commonly rely on one or more decoding steps, typically involving an iterative decoder that incurs significant computational costs, especially for large values of parameters ξ and K. Moreover, the accuracy of impulse-based methods in estimating the minimum distance of a code largely depends on the effectiveness of the sub-optimal decoding steps compared to ML.

However, the mHD is an intrinsic property of the code itself, which motivates the quest for alternative solutions that do not rely on decoding. This section introduces a novel framework for the identification of RTZ sequences of CCs. The herein defined notations will be employed subsequently in Section III-C for the estimation of the mHD of concatenated codes.

A. RTZ sequences and recursive convolutional codes

Due to the linearity of CCs, the distance spectrum of a code can be analyzed by examining the Hamming weight of non-zero codewords. In practice, this can be achieved by going through the code trellis to enumerate sequences that leave the zero state and then return to it, referred to as RTZ sequences. For recursive CCs, any non-RTZ error event, even if it affects only a limited number of information bits, results in parity bits being erroneous (i.e., set to '1') until the end of the frame. In contrast, an RTZ sequence generates erroneous parity bits only over the duration of the sequence.

For CCs, the mHD is also known as the free distance of the code [8] and corresponds to the RTZ sequence with the smallest Hamming weight [8].

Let R_w denote an RTZ sequence with input (systematic) weight (IW) w, defined as:

$$R_w = \{x_0, x_1, \ldots, x_{w-1}\}, \tag{1}$$

where $x_0, x_1, \ldots, x_{w-1}$ indicate the bit indices of the '1's relative to the start of the RTZ sequence. For instance, in an 8-state recursive systematic convolutional (RSC) code with generator polynomial $(1, 15/13)$ in octal representation, an RTZ sequence with $w = 3$, represented as $[000101100\ldots00]$, is denoted by $\{3, 5, 6\}$. Similarly, a sequence with $w = 4$, represented as $[1110100\ldots00]$, is denoted by $\{0, 1, 2, 4\}$.

For an RSC code, any RTZ sequence of IW $w = 2$ can be described as:

$$\{x_0, x_1\} \quad \text{with} \quad |x_0 - x_1| \bmod p = 0, \tag{2}$$

where p is the length of the shortest RTZ sequence of the code with $w = 2$, also called the period of the generator polynomial.

We define an atomic set RTZ_a as the collection of all possible RTZ sequences within one period p:

$$\mathit{RTZ}_a = \{R_w : 0 \le w < p,\ 0 \le x_i < p\}. \tag{3}$$

Additionally, we introduce a *Reduction* operation F_R that takes the modulo p of each element in an initial sequence $S = \{s_i\}_{i \in [\![0, w-1]\!]}$, and removes all pairwise repetitions in the resulting list to obtain a reduced sequence of at most length p. Defining $S_1 \uplus S_2 = (S_1 \cup S_2) \setminus (S_1 \cap S_2)$ for a couple of sequences (S_1, S_2), the reduction function can be expressed as:

$$F_R(S) = \biguplus_{i=1}^{|S|} \{s_i \bmod p\}, \tag{4}$$

where $|.|$ indicates the cardinality of a set, and $|S|$ is therefore the number of bits equal to '1' in the data sequence.

Applying the reduction operation to a sequence S results in a transformed sequence that can be used to determine whether S is an RTZ sequence or not, as stated in the following.

Theorem 1: If the reduction of a sequence S under the operation F_R yields a sequence belonging to the atomic set RTZ_a or the empty set $\emptyset$, then S is an RTZ sequence, and conversely:

$$F_R(S) \in (\mathit{RTZ}_a \cup \{\emptyset\}) \Leftrightarrow S \text{ is an RTZ sequence.} \tag{5}$$

Proof: We can decompose S as follows: $S = \bigcup_k S_k$, where

$$S_k = \big\{s_i : s_i \bmod p = k, i \in [1, |S|]\big\}, \tag{6}$$

with each S_k representing a sequence with bits equal to '1' positioned at multiples of p in S. If $F_R(S) = \emptyset$, it follows that $|S_k|$ is even for all k. Defining $l_k = |S_k|/2$, we can further decompose S into $\sum_k l_k$ sequences of IW $w = 2$, each denoted by $U_{j,k}$ for $j \in [1, l_k]$, such that $S = \bigcup_{j,k} U_{j,k}$. By construction, each $U_{j,k}$ is an IW-2 RTZ sequence. Since the bitwise sum of multiple RTZ sequences is itself an RTZ sequence, we can infer:

$$F_R(S) = \emptyset \Rightarrow S \text{ is an RTZ sequence.} \tag{7}$$

Now consider the case where $F_R(S) = V \neq \emptyset$. We define a new sequence S' such that $S = S' \uplus V$. Since the reduction function F_R is linear over the $\uplus$ operation, we have:

$$\begin{aligned} F_R(S') &= F_R\big(S \uplus V\big) \\ &= F_R(S) \uplus F_R(V) = V \uplus V = \emptyset. \end{aligned} \tag{8}$$

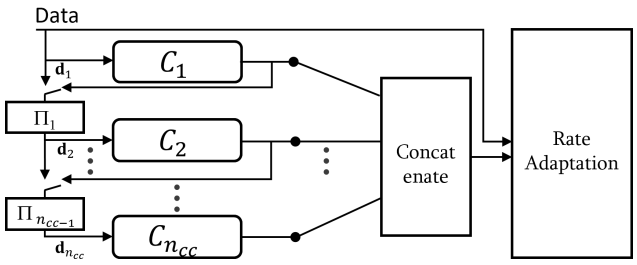

Fig. 1: An example of multiple hybrid concatenated codes.

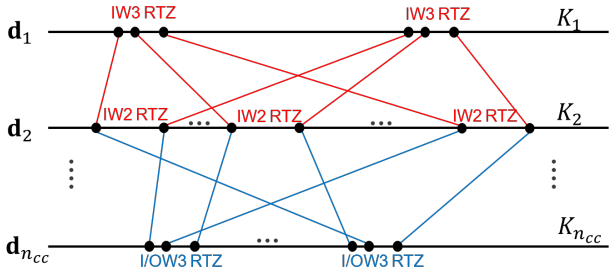

Fig. 2: An RTZ sequence interleaved to form other RTZ sequences within a series of linked hybrid codes. d_1 and d_2 sequences illustrate a parallel concatenation.

According to Eq. (7), S' is an RTZ sequence. Therefore, the sequence $S = S' \cup V$ is an RTZ sequence if and only if V is either an RTZ sequence or the all-zero vector.

The reverse implication follows similarly:

$$S \text{ is RTZ, } |S_k| \bmod 2 = 0 \; \forall k \Rightarrow F_R(S) = \emptyset. \tag{9}$$

If $|S_k|$ is not even for some values of k, there exists a unique sequence $V = \{v_k : |S_{v_k}| \bmod 2 = 1\}$ of p bits such that $F_R(V \cup S) = \emptyset$, implying that V can only be an RTZ sequence. ■

As an illustrative example, consider the 8-state RCS code $(1, 15/13)$, which has a period $p = 7$. We examine two sequences, $S_1 = \{3, 4, 5, 6, 7, 11, 14\}$ and $S_2 = \{1, 2, 3, 8, 9, 10\}$. The reduction operation $F_R(S)$ yields $\{3, 5, 6\}$, which belongs to RTZ_a, and $\emptyset$, respectively. Both S_1 and S_2 are therefore RTZ sequences.

B. RTZ sequences and concatenated convolutional codes

RTZ sequences for a single RSC code were discussed in Section III-A. In concatenated coding schemes, an interleaver π_q permutes the input sequence d_{q+1} of length K_{q+1} of each component encoder C_{q+1} ($q \in [1, n_{cc} - 1]$) prior to encoding, ensuring that each component encoder processes a distinct version of the input data as shown in Fig. 1. Depending on the concatenation structure – whether parallel, serial, or hybrid – the input sequence of each encoder may incorporate parts of the input and/or output sequences of preceding encoders.

We first extend the concept of RTZ sequences to the parallel concatenation of two RSC codes, commonly known as turbo codes. A turbo RTZ (TRTZ) sequence is defined as a sequence that simultaneously satisfies the RTZ conditions for both component encoders: the one processing data in its natural order and the other in the interleaved order. As a result, the IW w of a TRTZ sequence matches the IW of each component encoder's RTZ sequence. At high signal-to-noise ratio, the error correction performance of turbo codes is determined by TRTZ sequences with low total Hamming weight, also referred to as the output weight (OW).

The RTZ concept can similarly be extended to serial concatenations of codes. In this scenario, the output associated with an RTZ sequence from the outer encoder, after interleaving, should also form an RTZ sequence for the inner encoder. This structure produces an OW at the outer encoder's output that serves as the IW for the sequence processed by the inner encoder. Furthermore, the RTZ concept can be generalized to hybrid concatenated codes involving more than two encoders by combining parallel and serial schemes. This is illustrated in Fig. 1, where outputs from multiple encoders are concatenated and rate adaptation is applied.

In the following, we describe the proposed method for identifying RTZ sequences in the specific context of turbo codes, chosen due to their widespread adoption in standardized communication systems. However, the presented approach can be easily generalized to other types of concatenation by respecting the input/output relationships and RTZ sequences of component codes.

A key observation is that an RTZ sequence for a concatenated code can be decomposed into multiple elementary RTZ sequences within each component code, which may overlap.

For a given global RTZ sequence, the structure and IW of the corresponding elementary RTZ sequences may differ across the component codes. For example, in Fig. 2, the input sequences d_1 and d_2, each of length $K_1 = K_2$, represent a TRTZ sequence with an IW of $w = 6$ (denoted as IW-6). This sequence can be decomposed into two distinct IW-3 RTZ sequences in the natural order, which correspond to three separate IW-2 RTZ sequences in the interleaved order.

C. Graph-based mHD estimation for concatenated convolutional codes

For a conventional turbo code, the OW of a codeword is the sum of three terms: the IW of the codeword, determined by its systematic bits; the weight of the parity bits generated by the first component encoder, C_1, in the natural order; and the weight of the parity bits generated by the second component encoder, C_2, in the interleaved order. Finding the mHD of a turbo code involves identifying the TRTZ sequence that minimizes the OW.

To identify TRTZ sequences, we model the relationship between the input bit positions of both component encoders using an undirected graph structure, defined as

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \tag{10}$$

where $\mathcal{V}$ denotes a set of vertices and $\mathcal{E}$ represents a set of edges connecting pairs of vertices $(v, v') \in \mathcal{V}^2$. The vertex set $\mathcal{V}$ is partitioned into two subsets $\mathcal{V}_1$ and $\mathcal{V}_2$, where each

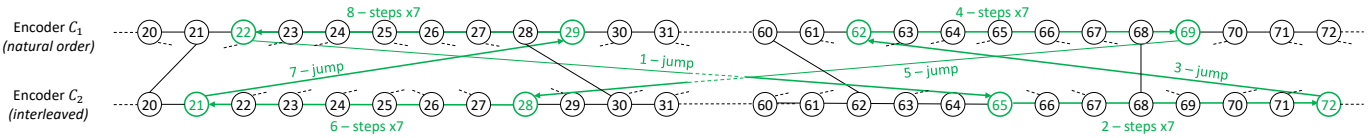

Fig. 3: Example illustrating the structure of the graph $\mathcal{G}$ for $K \geq 72$ bits. The green lines identify a TRTZ cycle in this graph.

vertex in these sets, $v_{i,q} \in \mathcal{V}_q, q \in \{1,2\}$, labels the i^{th} input bit position of component encoder C_q, with $i \in [\![0, K-1]\!]$. The edge set $\mathcal{E}$ is further divided into subsets, denoted as *steps* and *jumps*: $\mathcal{E} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{J}$. A step edge $S_{i,q} = (v_{i,q}, v_{i\pm1,q}) \in \mathcal{S}_q$, $q \in \{1,2\}$, connects two vertices related to the same component encoder q, corresponding to consecutive input bit positions $(i, i \pm 1)$. When tailbiting termination is applied [9], the last and first bits of the input frame are treated as adjacent. A jump edge $J_i = (v_{i,1}, v_{\pi^{-1}(i),2}) \in \mathcal{J}$ connects vertices corresponding to the same data bit, considered respectively in the natural and in the interleaved data sequences. Therefore, the configuration of the jump edges is only determined by the characteristics of the interleaver.

Fig. 3 shows an example of the graph structure $\mathcal{G}$, featuring two component encoders and an input sequence with at least 72 information bits ($K \geq 72$). To enhance clarity, only a subset of jump edges is displayed. The vertices are labeled according to their respective bit positions, with the upper and lower sequences in Fig. 3 corresponding to the input frames of component encoders C_1 and C_2, respectively. For example, vertex $v_{24,1} \in \mathcal{V}_1$ is labelled 24 in the upper sequence of $\mathcal{G}$.

Moreover, the subgraph associated with the TRTZ sequence $\{22, 29, 62, 69\}$ for encoder C_1 is shown in Fig. 3, where edges and vertices corresponding to bit value '1' are highlighted in green. This TRTZ sequence is composed of two IW-2 RTZ sequences, $\{22, 29\}$ and $\{62, 69\}$, for encoder C_1 and two IW-2 RTZ sequences, $\{21, 28\}$ and $\{65, 72\}$, for encoder C_2. The vertices of each IW-2 sequence are separated by $p = 7$ trellis stages. Connecting the specified vertices through step and jump edges forms a cycle within the graph. Due to their inherently periodic structure, all IW-2 TRTZ sequences represent cycles in $\mathcal{G}$. However, depending on how each component code decomposes into elementary sequences, a TRTZ sequence may instead be represented as a walk on $\mathcal{G}$ that does not form a cycle. We refer to such walks as *TRTZ walks* within the graph $\mathcal{G}$.

A TRTZ walk connects the vertices of one or more elementary RTZ sequences for encoders C_1 and C_2, alternating between jump and step edges. Note that while jump edges are determined by the interleaver connections, the walk itself dictates the choice of step edges, either in increasing or decreasing order, within the input sequences of C_1 and C_2.

According to these definitions, if $\mathcal{E}_W$ represents the set of edges forming a walk in $\mathcal{G}$, and $\mathcal{W} = \{v_{i,1}, v_{\Pi^{-1}(i),2} : J_i \in \mathcal{E}_W \cap \mathcal{J}\}$ denotes the corresponding set of vertices in $\mathcal{V}_1$ and $\mathcal{V}_2$ traversed by a jump edge, then $\mathcal{W}$ is a TRTZ walk if and only if, $\forall q \in \{1,2\}$:

$$F_R\Big(\{i : v_{i,q} \in \mathcal{W} \cap \mathcal{V}_q\}\Big) \in (\text{RTZ}_a \cup \{\emptyset\}). \tag{11}$$

Applying these conditions to the TRTZ sequence example illustrated in Fig. 3 allows us to represent the TRTZ walk as: $\mathcal{W} = \{v_{22,1}, v_{65,2}, v_{72,2}, v_{62,1}, v_{69,1}, v_{28,2}, v_{21,2}, v_{29,1}\}$. The indices denote the sequence of alternating jumps and steps required to complete the walk, demonstrating a possible traversal order starting from vertex $v_{22,1}$. In a parallel concatenation, both natural and interleaved order vertices are included in $\mathcal{W}$ for each systematic bit. For example, $v_{22,1}$ and $v_{65,2}$ refer to the same data bit and are both elements of $\mathcal{W}$ in this example.

Since the jump edges connecting the vertices are predetermined by the interleaver, we propose simplifying notation by reducing $\mathcal{W}$ to include only the vertices in their natural order to the following analysis. Thus, we define:

$$\mathcal{W}_{\text{TRTZ}} = \{v_{i,1} : v_{i,1} \in \mathcal{W} \cap \mathcal{V}_1, \mathcal{W} \text{ satisfies (11)}\}. \tag{12}$$

The mHD of a turbo code is determined by identifying TRTZ sequences with the smallest Hamming weight. To estimate it, we evaluate the OW of all TRTZ walks in $\mathcal{G}$:

$$d_{min} = \min_{\mathcal{W}_{\text{TRTZ}} \in \mathcal{T}_{\text{walk}}} W\Big(R_w^{\mathcal{W}_{\text{TRTZ}}}\Big), \tag{13}$$

where $\mathcal{T}_{\text{walk}}$ denotes the set of all TRTZ walks in $\mathcal{G}$, and $W(.)$ represents the OW obtained after turbo encoding the sequence $R_w^{\mathcal{W}_{\text{TRTZ}}} = \{i : v_{i,1} \in \mathcal{W}_{\text{TRTZ}}\}$, with IW $w = |\mathcal{W}_{\text{TRTZ}}|$. The bit positions set to '1' correspond to the vertex indices in $\mathcal{W}_{\text{TRTZ}}$. The list of TRTZ walks with OW d_{min}, denoted by $\mathcal{T}_{\text{walk}}^{\text{mHD}}$, can be obtained through

$$\mathcal{T}_{\text{walk}}^{\text{mHD}} = \underset{\mathcal{W}_{\text{TRTZ}} \in \mathcal{T}_{\text{walk}}}{\text{argmin}} W\Big(R_w^{\mathcal{W}_{\text{TRTZ}}}\Big), \tag{14}$$

and the number of mHD codewords (the multiplicity) is directly given by $|\mathcal{T}_{\text{walk}}^{\text{mHD}}|$. However, as the IW of TRTZ sequences increases, the number of possible walks grows exponentially, making this approach computationally impractical.

To address the computational challenge, we propose an algorithm that reduces the number of TRTZ walks considered in (13) while preserving the accuracy of the mHD estimation. Considering that TRTZ sequences with lower IW values are more likely to yield codewords with low mHD, the search is restricted to TRTZ sequences with an IW of at most $w_{\max}$. In addition, the OW associated with a TRTZ walk increases with the number of successive step edges between two jump edges. Thus, any TRTZ walk containing a pair of vertices connected by more than $s_{\max}$ step edges is excluded from the search space. The structure of the walk also impacts the algorithm's complexity. Recall that all IW-2 TRTZ walks form TRTZ cycles due to the periodic nature of the generator polynomial.

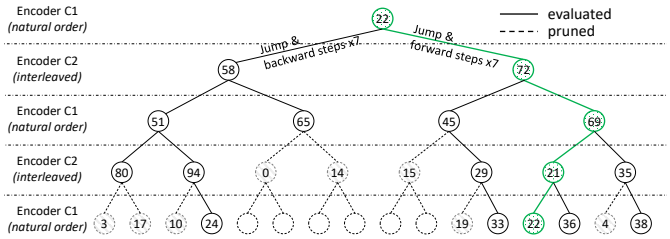

Fig. 4: Example of a tree search approach for finding TRTZ walks with parameters $\boldsymbol{E}_{\text{step}} = [-7, 7]$ and $w_{\max} = 4$.

D. Simplified algorithm for identifying TRTZ walks

To identify the TRTZ walks, we propose exploring the graph using a tree-search approach. Starting from a root vertex $v_{r,1}$ (position r), the exploration first traverses the graph through a jump edge to reach the interleaved vertex index $v_{\pi^{-1}(r),2}$. From $v_{\pi^{-1}(r),2}$, the graph is further explored by testing up to $s_{\max}$ distinct paths, each comprising multiple step edges. The number of steps in each path is specified by a vector $\boldsymbol{E}_{\text{step}}$ of size $s_{\max} \times 1$, where, for example, $\boldsymbol{E}_{\text{step}} = [-7, 7]$ represents 7 steps in the backward direction for the first path and 7 steps in the forward direction for the second path.

This process of alternating jumps and exploring up to $s_{\max}$ steps is repeated up to $w_{\max}$ times. If a cycle is detected, the current walk is evaluated following (12) and added to the list of identified TRTZ walks $\mathcal{T}_{\text{walk}}$ if it meets the definition.

The tree-search method can be implemented as follows: Each evaluated path in the search tree is represented by a matrix $\boldsymbol{T}$, where each row corresponds to an explored path in the graph, and each column represents the vertex indices in either C_1 or C_2 forming that path. Starting with $\boldsymbol{T} = r$ and $\mathcal{T}_{\text{walk}} = \{\emptyset\}$, the matrix $\boldsymbol{T}$ is updated iteratively at each step $\ell \in [\![1, w_{\max}]\!]$ of the algorithm described below and summarized in Alg. 1.

1) Procedure for "jump, expand, step, and update": In each iteration, the last column of $\boldsymbol{T}$ is extracted to form a vector $\boldsymbol{V}$ of size $L_T \times 1$, containing the last vertex index of each path evaluated in the previous iteration. The vertex indices after traversing a jump edge are computed as $\boldsymbol{V}_{\text{jump}} = \pi_\ell(\boldsymbol{V})$, where $\pi_\ell = \pi$ if $\text{mod}_2(\ell) = 1$ and $\pi_\ell = \pi^{-1}$ otherwise. The vector $\boldsymbol{V}_{\text{jump}}$ is then expanded by repeating each row $s_{\max}$ times: $\boldsymbol{V}_{\text{exp}} = \boldsymbol{V}_{\text{jump}} \otimes \mathbf{1}_{s_{\max}}$, where $\mathbf{1}_x$ denotes an all-ones vector of size $x \times 1$, and $\otimes$ represents the Kronecker product. For each newly tested path in the tree search corresponds a number of traversed step edges and a list of vertex indices $\boldsymbol{V}_{\text{step}}$ that is computed. Finally, $\boldsymbol{T}$ is updated by concatenating it with $\boldsymbol{V}_{\text{step}}$. This procedure is summarized by lines 4 to 8 in Alg. 1.

2) Procedure for "TRTZ walks extraction": The list of row indices $\mathcal{L}_{\text{cycle}}$ in $\boldsymbol{T}$, where vertex indices form cycles, is obtained through line 9 in Alg. 1. TRTZ walks are then identified by finding the cycles that generate valid TRTZ sequences in line 11. They are subsequently appended to the list $\mathcal{T}_{\text{walk}}$ of previously identified TRTZ walks in line 12.

Algorithm 1 Minimum Distance Estimation Method

Initialize $\boldsymbol{E}_{\text{step}}$ and $\mathcal{T}_{\text{walk}} = \{\emptyset\}$
1: **for** $r \in [\![0, K-1]\!]$ **do**
2: Set $\boldsymbol{T} = r$.
3: **for** $\ell \in [\![1, w_{\max}]\!]$ **do**
4: Extract the last column of T to form a vector $\boldsymbol{V}$ of size $L_T \times 1$.
5: Compute the vertex indices after traversing a jump edge:

$$\boldsymbol{V}_{\text{jump}} \leftarrow \begin{cases} \pi(\boldsymbol{V}), & \text{if } \text{mod}_2(\ell) = 1, \\ \pi^{-1}(\boldsymbol{V}), & \text{otherwise.} \end{cases}$$

6: Expand V_{jump} by repeating each row $s_{\max}$ times:

$$\boldsymbol{V}_{\text{exp}} \leftarrow \boldsymbol{V}_{\text{jump}} \otimes \mathbf{1}_{s_{\max}},$$

7: Compute the vertex indices after traversing step edges:

$$\boldsymbol{V}_{\text{step}} \leftarrow \text{mod}_K(\boldsymbol{V}_{\text{exp}} + \boldsymbol{E}_{\text{step}} \otimes \mathbf{1}_{L_V \times 1}).$$

8: Update T by concatenating it with $\boldsymbol{V}_{\text{step}}$:

$$\boldsymbol{T} \leftarrow [\boldsymbol{T} \otimes \mathbf{1}_{s_{\max}}, \boldsymbol{V}_{\text{step}}]$$

9: Identify TRTZ walks that form a cycle

$$\mathcal{L}_{\text{cycle}} = \{i : \exists j \in [\![0, \ell-1]\!] \mid \boldsymbol{T}^{(\ell)}[i, j] = \boldsymbol{T}^{(\ell)}[i, \ell]\}$$

10: **for** $i \in \mathcal{L}_{\text{cycle}}$ **do**
11: **if** $\mathcal{W} = \{\boldsymbol{T}[i, j]\}_{j \in [\![0, \ell-1]\!]}$ satisfies (11) **then**
12: $\mathcal{T}_{\text{walk}} = \mathcal{T}_{\text{walk}} \cup \mathcal{W}$.
13: **end if**
14: **end for**
15: **end for**
16: **end for**
17: Calculate d_{min} using (13) and $|\mathcal{T}_{\text{walk}}^{\text{mHD}}|$ using (14).

The procedures are repeated for each IW iteration ℓ until $w_{\max}$ is reached, then for each root vertex until the last vertex (end of the frame) is reached. Then, the minimum distance d_{min} and its corresponding multiplicity $|\mathcal{T}_{\text{walk}}^{\text{mHD}}|$ are computed using line 17 in Alg. 1.

Fig. 4 illustrates an example of the tree-search procedure for finding the TRTZ sequence $\{22, 29, 62, 69\}$, as introduced in the previous section (see Fig. 3). For this example, the set of step edges being explored is limited to $\boldsymbol{E}_{\text{step}} = [-7, 7]$. Starting at vertex 22 in C_1, a jump leads to vertex 65 in C_2, followed by a move to vertices 58 or 72 after traversing 7 steps either backward or forward, thereby opening 2 paths in the search tree. This process continues, resulting in 4 paths in the next layer, corresponding to 2 iterations in Alg. 1. At this point, the matrix $\boldsymbol{T}$ is:

$$\boldsymbol{T} = \begin{pmatrix} 22 & 72 & 69 \\ 22 & 72 & 45 \\ 22 & 58 & 65 \\ 22 & 58 & 51 \end{pmatrix}. \tag{15}$$

After $w_{\max}$ iterations, a cycle $\{22, \pi(62), 69, \pi(29), 22\}$ is identified as the IW-4 TRTZ sequence used in the example.

E. Additional constraints for zero-termination codes

Alg. 1 is formulated for tailbiting termination, leveraging the symmetric structure of the graph $\mathcal{G}$ at both the start and end of the frame. However, in the case of zero termination, the computation of $\boldsymbol{V}_{\text{step}}$ must be adjusted to properly handle the boundary conditions at the head and tail vertices. Specifically, any step that would move below the head vertex (index 0) must be constrained to remain at vertex 0, while any step exceeding the tail vertex (index $K-1$) must be constrained to $K-1$. Incorporating these constraints, the computation of line 7 in Alg. 1 for zero termination is modified as follows:

$$\boldsymbol{V}_{\text{step}} \leftarrow \min\big(\max(\boldsymbol{V}_{\text{exp}} + \boldsymbol{E}_{\text{step}} \otimes \mathbf{1}_{L_V \times 1}, 0), K-1\big).$$

RTZ sequences are forced at the end of the frame for each CC by component-wise termination bits, leading to termination vertices. Not shared by both component encoders, these vertices do not allow for jump edges and IW $w = 1$ TRTZ sequences now arise for root vertices starting towards the end of the frame. To assess this effect, we propose to explicitly evaluate these $w = 1$ TRTZ sequences. With a total of K possible IW $w = 1$ codewords, this evaluation can be performed with a negligible computation time overhead.

F. Complexity analysis

The computational complexity of the proposed approach corresponds to exploring a tree of depth $w_{\max}$, where each node has $s_{\max}$ branches (as illustrated in Fig. 4). This tree search is conducted for each bit position $r \in [\![0, K-1]\!]$, which serves as the root of the tree. Consequently, the complexity scales as $O(K \cdot s_{\max}^{w_{\max}})$. Note that this represents an upper bound on the complexity. Indeed, it can be significantly reduced due to the frequent occurrence of cycles in the tree, which can be pruned. Applied pruning methods include:

1) If a cycle is detected, all subsequent child nodes along that path can be discarded.
2) With zero termination, once the head or tail vertices are encountered in the tree search, all subsequent child nodes along that path can be discarded.
3) Any paths that involve a vertex index $p < r$ and have already been evaluated in previous searches with root p can be eliminated. This pruning strategy is illustrated in Fig. 4, where all vertex indices lower than 22 in C_1 are removed from the search space.
4) The set $\boldsymbol{E}_{\text{step}}$ can be chosen to leverage bit patterns specific to certain types of TRTZ sequences. For instance, to detect TRTZ sequences composed of multiple IW-2 RTZ sequences, only multiples of p steps are explored when searching for TRTZ walks:

$$\boldsymbol{E}_{\text{step}} = [-p, p, -2p, 2p, \cdots] = \big[(-1)^k p\lceil k/2 \rceil\big]_{k \in [\![1, s_{\max}]\!]}, \tag{16}$$

where $\lceil . \rceil$ denotes the ceiling operator.

By appropriately selecting values for $w_{\max}$ and $s_{\max}$, an effective trade-off can be made between computational complexity and accuracy. Indeed, using pruning method 4) motivates applying twice Alg. 1, first with the set of $(w_{\max}, s_{\max})$ values optimized for detecting multiple IW-2 RTZ sequences, an second with different set of values for the detection of remaining TRTZs. Typical values for $w_{\max}$ range from 2 to 6 and for $s_{\max}$ from 1 to 64. Finally, the obtained list of TRTZ walks $\mathcal{T}_{\text{walk}}$ of the two algorithm runs are merged before evaluating $d_{\min}$ and its corresponding multiplicity.

Furthermore, whereas the complexity of impulse-based methods scales as $O(K^2)$, the proposed algorithm achieves a significantly lower complexity, scaling linearly with K. This reduction becomes particularly advantageous for long frame sizes as shown in Section IV.

IV. Case Study: Application to the LTE Code

To assess the efficiency of the proposed algorithm, the following parameters were considered:

- *First algorithm run for detection of TRTZ sequences composed of multiple IW-2 RTZ sequences:* The maximum weight was set to $w_{\max} = 6$, and the step edges explored were defined as $\boldsymbol{E}_{\text{step}} = [-p, p, -2p, 2p, \cdots]$, with $s_{\max} = 14$.
- *Second algorithm run for detection of remaining TRTZ sequences:* The parameters were set to $w_{\max} = 4$, $\boldsymbol{E}_{\text{step}} = [-1, 1, -2, 2, \cdots]$, with $s_{\max} = 40$.

The proposed algorithm was applied to the LTE turbo code for several frame sizes, using the corresponding QPP interleaver. The LTE standardized zero termination and the tailbiting termination were both considered. Table I provides the estimated mHDs, $d_{\min}$, along with the corresponding estimated multiplicities. For zero termination, the distance results marked with $\star$ align with those in [10]. It was observed that termination-induced TRTZ sequences uniquely define $d_{\min}$, with only one associated multiplicity. Notably, we found that the upper bound value of $d_{\min} \leq 50$ reported in [11] for $K = 6144$, later refined to $d_{\min} \leq 47$ in [12], is fairly loose. Indeed, our algorithm found a mHD of $d_{\min} = 26$ via the input sequence $\{6124, 6141\}$. On the other hand, adopting tailbiting termination eliminates TRTZ sequences introduced by the termination bits, resulting in a significantly larger mHD.

Table I also provides the average runtime (in minutes) required to compute $d_{\min}$[1] and corresponding multiplicity for both the proposed and the TIM algorithms. To ensure a fair comparison, all algorithms were implemented and simulated on the same CPU (Intel® Xeon® Silver 4114 @ 2.20GHz) for all frame sizes.

Both algorithms converge to the same $d_{\min}$ and multiplicity values upon completion. However, there is a significant difference in execution time. For very small frame sizes ($K = 40$ or $K = 64$) and tailbiting termination, the proposed algorithm requires 2.6 min and 4.8 min, respectively, which is slightly

[1]The two algorithm runs are taken into consideration in the reported runtime.

TABLE I: Estimated mHD and multiplicity values for several frame sizes of the LTE code family, coding rate $R = 1/3$ and two termination techniques. Comparison of algorithm execution time for the proposed and TIM [6] methods (the algorithm execution was bounded to 10,000 min).

	Zero Termination			Tailbiting Termination					
	Proposed			Proposed			TIM [6]		
K	$d_{\min}$	Mult.	Time (min)	$d_{\min}$	Mult.	Time	$d_{\min}$	Mult.	Time (min)
40	11*	1	0.06	17	20	2.6	17	20	1.08
64	12*	1	0.11	20	32	4.8	20	32	4.36
128	16*	1	0.28	21	64	8.6	21	64	42.7
512	27*	1	3.3	33	128	30	33	128	6480
2048	27	1	20.4	44	512	80.5	44	4	10,000
4096	31	1	41.8	44	512	76.5	50	10	10,000
6144	26	1	77.2	50	12288	101.1	50	2	10,000

*Distance estimation in accordance with [10].

more than TIM's 1.08 min and 4.36 min. However, as K increases, our algorithm becomes significantly more time-efficient than TIM. For example, for $K = 512$, the proposed algorithm requires 30 min compared to TIM's 6480 min. Notably, for large frame sizes, our algorithm remains practical and accurate, whereas TIM fails to find the correct d_{min} within 10,000 min for $K = 4096$.

Interestingly, the execution time of the proposed algorithm for $K = 2048$ with tailbiting termination is greater than for $K = 4096$. This difference can be attributed to the simplifications introduced in Section III-F, whose impact depends on the specific characteristics of the interleaver.

The reduced runtime of the proposed method makes it particularly suitable for integration with interleaver design approaches, such as those in [13], [14]. This combination represents a promising direction for further investigation, as the proposed distance evaluation method demonstrates improved scalability with frame size, a critical factor for this class of codes.

ACKNOWLEDGEMENT

This work was supported by the PEPR 5G and Future Networks project funded by the Agence Nationale de la Recherche (ANR) grant NF-PERSEUS : "ANR-22-PEFT-0004".

V. CONCLUSION

This work presents a novel approach for estimating the minimum distance of concatenated convolutional codes by introducing a new method for detecting RTZ sequences generated by these codes. The primary innovation of the proposed method lies in leveraging encoder characteristics exclusively, thereby eliminating the need for iterative decoding used in prior leading methods, such as impulse-based approaches. The proposed method achieves reliability comparable to one of the best existing methods (TIM) while significantly reducing computational complexity, resulting in substantial reductions in execution time. This approach serves as an effective tool for analyzing and designing concatenated convolutional codes in practical applications. Future work will explore the proposed RTZ detection method to design serial, parallel and hybrid concatenated codes, even for large frame sizes.

REFERENCES

[1] 3GPP, "LTE; Evolved universal terrestrial radio access (E-UTRA); Multiplexing and channel coding," TS 36.212 Release 14, 2017.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proc. of ICC '93 - IEEE Int. Conf. Commun.*, vol. 2, May 1993, pp. 1064–1070 vol.2.

[3] R. Garello, P. Pierleoni, and S. Benedetto, "Computing the free distance of turbo codes and serially concatenated codes with interleavers: algorithms and applications," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 5, pp. 800–812, Aug. 2001.

[4] C. Berrou, S. Vaton, M. Jezequel, and C. Douillard, "Computing the minimum distance of linear codes by the error impulse method," in *Proc. 2002 Global Telecommun. Conf.*, vol. 2, Mar. 2002, pp. 1017–1020 vol.2.

[5] R. Garello and A. Vila-Casado, "The all-zero iterative decoding algorithm for turbo code minimum distance computation," in *2004 IEEE Int. Conf. Commun.*, vol. 1, Jul. 2004, pp. 361–364.

[6] S. Crozier, P. Guinand, and A. Hunt, "Estimating the minimum distance of turbo-codes using double and triple impulse methods," *IEEE Commun. Lett.*, vol. 9, no. 7, pp. 631–633, Jul. 2005.

[7] ——, "Estimating the minimum distance of large-block turbo codes using iterative multiple-impulse methods," in *Proc. Int. Symp. Turbo Codes Rel. Topics (ISTC)*, Jun. 2006, pp. 1–6.

[8] C. Douillard and C. Berrou, "Turbo codes with rate-m/(m+1) constituent convolutional codes," *IEEE Trans. Commun.*, vol. 53, no. 10, pp. 1630–1638, Oct. 2005.

[9] C. Weiss, C. Bettstetter, and S. Riedel, "Code construction and decoding of parallel concatenated tail-biting codes," *IEEE Trans. on Inf. Theory*, vol. 47, no. 1, pp. 366–386, 2001.

[10] L. Trifina and D. G. Tarniceriu, "Improved method for searching of interleavers using Garello's method," *CoRR*, vol. abs/1203.1410, 2012.

[11] A. Nimbalker, Y. Blankenship, B. Classon, and T. K. Blankenship, "ARP and QPP interleavers for LTE turbo coding," in *IEEE Wireless Commun. Netw. Conf. (WCNC)*, Las Vegas, NV, USA, 2008, pp. 1032–1037.

[12] M. Punekar, F. Kienle, N. Wehn, A. Tanatmis, S. Ruzika, and H. W. Hamacher, "Calculating the minimum distance of linear block codes via integer programming," in *Proc. Int. Symp. on Turbo Codes & Iterative Inf. Process. (ISTC*, Brest, France, Sep. 2010, pp. 329–333.

[13] R. Garzón Bohórquez, C. Abdel Nour, and C. Douillard, "On the equivalence of interleavers for turbo codes," *IEEE Wireless Commun. Lett.*, vol. 4, no. 1, pp. 58–61, Nov 2015.

[14] R. Garzón-Bohórquez, C. Abdel Nour, and C. Douillard, "Protograph-based interleavers for punctured turbo codes," *IEEE Trans. Commun.*, vol. 66, no. 5, pp. 1833–1844, Dec. 2018.