An Efficient ML Decoder for Tail-Biting Codes Based on Circular Trap Detection

Xiaotao Wang, Hua Qian, *Member, IEEE*, Weidong Xiang, *Member, IEEE*, Jing Xu, and Hao Huang, *Member, IEEE*

***Abstract*—Tail-biting codes are efficient coding techniques to eliminate the rate loss in conventional known-tail convolutional codes at a cost of increased complexity in decoders. In addition, tail-biting trellis representation of block codes makes the trellis-based maximum likelihood (ML) decoder desirable for implementation. Circular Viterbi algorithm (CVA) is introduced to decode the tail-biting codes for its decoding efficiency. However, its decoding process suffers from circular traps, which degrade the decoding efficiency. In this paper, we propose an efficient checking rule for the detection of circular traps. Based on this rule, a novel maximum likelihood (ML) decoding algorithm for tail-biting codes is presented. On tail-biting trellis, computational complexity and memory consumption of this decoder are significantly reduced comparing to other available ML decoders, such as the two-phase ML decoder. To further reduce the decoding complexity, we propose a new near-optimal decoding algorithm based on a simplified trap detection strategy. The performance of the above algorithms is validated with simulation.**

***Index Terms*—Tail-biting trellis, convolutional code, circular Viterbi algorithm, optimal decoder, circular trap.**

I. Introduction

TAIL-BITING technique is widely used in convolutional codes to eliminate the rate loss caused by the known-tail bits [1]–[3]. For example, the Worldwide Interoperability for Microwave Access (WiMAX) [4] and the Long Term Evolution (LTE) [5] standards all adopted the tail-biting convolutional codes in the control channel or broadcasting channel. For tail-biting convolutional codes, its encoder is initialized with the last several information bits, and thus its trellis representation has multiple initial states and the same multiple final states. The trellis of this kind is called tail-biting trellis. An example of the tail-biting trellis is given in Fig. 1, which has eight sections and four states at each section.

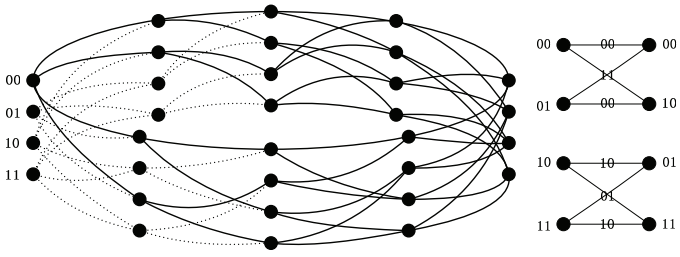


Fig. 1. The tail-biting trellis of a (8, 4) convolutional codes with octal generator polynomials of {7, 5}.The state transition butterflies are shown on the right, where the digital bits on the lines are outputs of the corresponding branches.

In tail-biting trellis, the branches that connect a starting state and a final state compose a path. Only the path that starts and terminates at the same state corresponds to a codeword in tail-biting codes; and this path is called a tail-biting path. All tail-biting paths which start from the same state form a sub-tail-biting trellis of this state. For example, in Fig. 1, all tail-biting paths starting from state 00 represented by solid lines form a sub-tail-biting trellis of state 00.

Many other block codes can also be represented by tail-biting trellis [6]–[10]. For the same linear block codes, the number of states in its tail-biting trellis representation can be as low as the square root of the number of states in its minimal conventional trellis representation [11]. For example, for the (24, 12) extended Golay code, the maximum number of states in its conventional trellis is 512, while the maximum number of states on its tail-biting trellis is only 16 [12]. In many cases, the tail-biting trellis can be advantageous over the conventional trellis. An efficient optimal decoder for tail-biting trellis is the focus of this work.

For tail-biting codes, the ML decoder is to find the optimal tail-biting path on the trellis given the received soft symbols. In the case of known-tail convolutional codes, the Viterbi algorithm obtains the ML path by finding the path with the maximum accumulated metric on the sub-tail-biting trellis of a known state for the received soft symbols. In a similar way, a straightforward ML decoder of tail biting codes can be obtained by performing Viterbi algorithm on all sub-tail-biting trellises. On the other hand, this method is impractical due to high computation complexity and large time delay.

The bidirectional efficient algorithm for searching code trees

Manuscript received on April 20, 2012; revised July 26, September 28, and November 30, 2012. The associate editor coordinating the review of this paper and approving it for publication was E. Ayanoglu.

This work was supported in part by the 100 Talents Program of the Chinese Academy of Sciences, the Shanghai Pujiang Talent Program No. 11PJ1408700, the National Key Science and Technology Project No. 2011ZX03003-003-04, and the International Science and Technology Cooperation project of China No. 2012DFG12060.

This paper was presented in part at IEEE Globecom, Houston, USA, December 2011.

X. Wang and H. Huang are with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, and the Graduate University of the Chinese Academy of Sciences (e-mail: {xiaotao.wang, hao.huang}@shrcwc.org).

H. Qian and J. Xu are with the Shanghai Research Center for Wireless Communications, Key Laboratory of Wireless Sensor Network and Communication, and the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences (e-mail: {hua.qian, jing.xu}@shrcwc.org).

W. Xiang is with the Electrical and Computer Engineering Department, University of Michigan-Dearborn (e-mail: xwd@umd.umich.edu).

Digital Object Identifier 10.1109/TCOMM.2013.020813.120275

(BEAST) algorithm, which is generally more efficient than Viterbi algorithm on conventional trellis, is an alternative ML decoder for tail-biting codes [12]. However, the decoding efficiency of the BEAST decoder degrades on tail-biting trellis due to the fact that the BEAST decoder has to be applied on every sub-tail-biting trellis consecutively.

The two-phase ML decoding scheme has been proposed for tail-biting trellis in [13], [14]. In the first phase, the decoder performs modified Viterbi algorithm [15] on tail-biting trellis and obtains accumulated path metrics of all survivor paths at every stage. In the second phase, the decoder performs the heuristic search algorithm to find the optimal path with metrics obtained in the first phase. In this scheme, however, two distinct algorithms have been adopted in two phases respectively, which make the decoder design complicated for practical application. In addition, the second phase of this algorithm requires unpredictable memory space for storing open paths, terminating states, heuristic function values of these states and the close table [14].

To simplify the decoder design, circular Viterbi algorithm (CVA) has been proposed by recording and repeating the received (soft) symbols beyond the block boundary and continuing Viterbi decoding. Many suboptimal CVA-based decoding algorithms, such as wrap-around Viterbi algorithm (WAVA) and Wang-Bhargava algorithm, have been developed [1], [15]. In some cases, these algorithms achieve near-optimal performance. However, during the decoding process, it can happen that the survivor paths obtained in current iteration are the same as those found in previous iterations. We call this phenomenon a circular trap. The presence of circular traps gives rise to a large amount of redundant iterations in CVA, the existing CVA-based decoding schemes are inherently non-convergent duo to circular traps. In this paper, we study the properties of CVA and circular traps. An efficient checking rule for circular traps is obtained. With the circular trap detection rule, we design a novel ML decoding algorithm for tail-biting codes, which not only provides optimum decoding performance but also converges with high speed. These features make it very attractive in practice.

To further simplify the decoder design, we can simplify the circular trap checking rule to get a sub-optimal decoding algorithm. The modified scheme exhibits near-optimal performance with low complexity. This algorithm has higher decoding efficiency than other existing sub-optimal decoding schemes.

The following parts of this article are organized as follows. In Section II, we review the decoding process of CVA and illustrate the circular trap phenomenon. In Section III, we propose an optimal decoding algorithm based on trap detection. In Section IV, modifications are proposed for the optimal decoder to further reduce the decoding complexity. Simulations are performed in Section V to demonstrate the performance of the proposed algorithms. Section VI concludes this paper.

II. Circular Trap in CVA

A. Brief review

In Fig.1, we give an example of the tail-biting trellis, which represents a (16, 8) tail-biting convolutional codes with octal generator polynomials of $\{7, 5\}$. The length of this trellis is $L = 8$. It contains 2^v states at location k, where $0 \leq k \leq L$, and v is the register length of its encoder. Denote by $\mathbf{S}_k$ the state space at location k. For tail-biting trellis, we have $\mathbf{S}_0 = \mathbf{S}_L$. In Fig.1, there are four states: $\{00, 01, 10, 11\}$ at each location. For convenience, we denote the state in its octal form, e.g., at location k, state 11 is denoted by $s_k(3)$. We have $\mathbf{S}_k = \{s_k(0), s_k(1), s_k(2), s_k(3)\}$, where $0 \leq k \leq 8$.

CVA decoder performs Viterbi trials along with the tail-biting trellis repeatedly until terminating conditions are fulfilled. Denote by $P^i(\beta^i(s), s)$ a survivor path obtained in the ith iteration, and $M^i_{\mathtt{path,net}}(\beta^i(s), s)$ the corresponding net path metric, where state $\beta^i(s)$ and state s are the starting and ending state of path $P^i(\beta^i(s), s)$. The net path metric $M^i_{\mathtt{path,net}}(\beta^i(s), s)$ is defined as:

$$M^i_{\mathtt{path,net}}(\beta^i(s), s) = M^i_{\mathtt{path},L}(s) - M^i_{\mathtt{path},0}(\beta^i(s)), \quad (1)$$

which is the summation of branch metrics accumulated in the current iteration. In equation (1), $M^i_{\mathtt{path},k}(s)$ is the accumulated path metric of the survivor path entering state s at location k in the ith iteration.

Among all survivor paths, there is an ML path (MLP) $P^i_{\mathtt{MLP}}(\beta^i(s^\dagger), s^\dagger)$ with net path metric $M^i_{\mathtt{MLP,net}}(\beta^i(s^\dagger), s^\dagger)$; and there could be an ML tail-biting path (MLTBP) $P^i_{\mathtt{MLTBP}}(s', s')$ with net path metric $M^i_{\mathtt{MLTBP,net}}(s', s')$, where $s^\dagger, s' \in \mathbf{S}_L$. For the first i iterations, the optimal ML path and its net path metric are denoted $P^{\mathtt{O}}_{\mathtt{MLP}}$ and $M^{\mathtt{O}}_{\mathtt{MLP}}$; and the optimal ML tail-biting path and its net path metric are denoted as $P^{\mathtt{O}}_{\mathtt{MLTBP}}$ and $M^{\mathtt{O}}_{\mathtt{MLTBP}}$. Initial values of $M^{\mathtt{O}}_{\mathtt{MLP}}$ and $M^{\mathtt{O}}_{\mathtt{MLTBP}}$ are set to zeros in the decoding process.

Similarly, we define the net state metric of state s as:

$$M_{\mathtt{state,net}}(s) = \min_i \{M^i_{\mathtt{path},L}(s) - M^i_{\mathtt{path},0}(s)\}. \quad (2)$$

In addition, denote by $sum(i)$ the number of states that have net state metrics greater than $M^{\mathtt{O}}_{\mathtt{MLTBP}}$ after the ith iteration. Key terms and notations defined in this section are summarized in Table I.

B. Circular Trap

Circular trap is a unique phenomenon in the CVA decoding process, which leads to non-convergence of the decoder. The following example demonstrates this phenomenon.

Example 1: A tail-biting code that can be represented by the tail-biting trellis in Fig. 1 is selected in this example. The information bit sequence is {00010000}, then the transmitted codeword is {(00), (00), (00), (11), (10), (11), (00), (00)}. After passing through an additive white Gaussian noise (AWGN) channel where the signal to noise ratio (SNR) is $E_b/N_0 = 0$ dB, the received sequence (for one realization) is {(1.207 0.506), (-0.664 1.356), (0.573 -0.344), (-1.960 -0.385), (-1.065 0.079), (-0.812 -1.917), (-0.421 0.669), (0.480 0.123)}. CVA is applied to decode the received sequence and the decoding procedure is presented as below:

1) For the 1st iteration, four survival paths are $P^1_0(s_0(1), s_8(0))$, $P^1_1(s_0(2), s_8(1))$, $P^1_2(s_0(1), s_8(2))$ and $P^1_3(s_0(1), s_8(3))$, where the net path metrics of each path are {10.573, 9.521, 10.945, 9.831} and the net state metrics of each final state are {10.573,

TABLE I
LIST OF KEY NOTATIONS AND TERMS

Notations/Terms	Description
v	The length of the encoder registers
$\mathbf{S}_k$	The set of states at location k of a tail-biting trellis, where $0 \le k \le L$, and the size of $\mathbf{S}_k$ is 2^v
$P^i(\beta^i(s), s)$	The survivor path that starts from state $\beta^i(s)$ and terminates at state s in the ith iteration
$M^i_{\mathtt{path,net}}(\beta^i(s), s)$	The net path metric of the path $P^i(\beta^i(s), s)$
$P^{\mathtt{O}}_{\mathtt{MLP}}, M^{\mathtt{O}}_{\mathtt{MLP}}$	The optimal ML path and its net path metric obtained in the first i iterations
$P^{\mathtt{O}}_{\mathtt{MLTBP}}, M^{\mathtt{O}}_{\mathtt{MLTBP}}$	The optimal ML tail-biting path and its net path metric obtained in the first i iterations
$M_{\mathtt{state,net}}(s)$	The net state metric of state s
$sum(i)$	The number of states that have net state metrics greater than $M^{\mathtt{O}}_{\mathtt{MLTBP}}$ after the ith iteration

9.521, 10.945, 9.831}, respectively. The ML path is $P^1_{\mathtt{MLP}} = P^1_2(s_0(1), s_8(2))$ with net path metric $M^1_{\mathtt{MLP,net}} = 10.945$. In this iteration, we cannot find a tail-biting path, thus $P^1_{\mathtt{MLTBP}}$ do not exist, and $M^1_{\mathtt{MLTBP,net}} = 0$. We have $sum(1) = 4$.

2) For the 2nd iteration, four survivor paths are $P^2_0(s_0(0), s_8(0))$, $P^2_1(s_0(2), s_8(1))$, $P^2_2(s_0(1), s_8(2))$ and $P^2_3(s_0(2), s_8(3))$, where the net path metrics of each path are {9.703, 9.521, 10.945, 8.807} and the net state metrics of each final state are {9.703, 9.521, 9.521, 9.831}. The ML path is $P^2_{\mathtt{MLP}} = P^2_2(s_0(1), s_8(2))$ with net path metric $M^2_{\mathtt{MLP,net}} = 10.945$. The ML tail biting path is $P^2_{\mathtt{MLTBP}} = P^2_0(s_0(0), s_8(0))$ with net path metric $M^2_{\mathtt{MLTBP,net}} = 9.703$. We have $sum(2) = 1$.
3) For the 3rd iteration, survivor paths are the same as that in the 1st iteration; the net state metrics of each final state are {9.703, 9.521, 9.521, 9.831}, $P^{\mathtt{O}}_{\mathtt{MLTBP}}$ is not updated, and $sum(3) = 1$.
4) For the 4th iteration, survivor paths are the same as that in the 2nd iteration; the net state metrics of each final state are {9.703, 9.521, 9.521, 9.831}, $P^{\mathtt{O}}_{\mathtt{MLTBP}}$ is not updated and $sum(4) = 1$.
5) $\cdots$.

From the decoding process above, we find that starting from the 3rd iteration and the 4th iteration, the survival paths and their corresponding net path metrics circulate. Circular traps occur, which make any additional iterations of CVA redundant. Unfortunately, there has been no simple way to detect circular traps and terminate the decoder in time [1], [15]. From the decoding process of the above example, we observe that the value of function $sum(i)$ remains unchanged for $i \ge 2$. This phenomenon suggests that we can detect circular traps by examining the value of $sum(i)$. In fact, the decoding process can be terminated after the 2nd iteration in the above example, and $P^2_{\mathtt{MLTBP}}$ is the correct tail-biting path corresponding to the transmitted codeword. With the circular trap detection, we can design a new ML decoder for tail-biting codes.

III. CIRCULAR TRAP DETECTION BASED ML DECODER

From the above example, we observe that if we can control the CVA decoding process by checking the value of $sum(i)$, it is possible to obtain a global optimal decoder with little computational complexity. We can exclude sub-tail-biting trellis of the state whose net state metric is less than $M^{\mathtt{O}}_{\mathtt{MLTBP}}$ from the set of searching candidates in the following iterations (the solid lines in Fig. 1 compose the sub-tail-biting trellis of state 00). We summarize these in Lemma 1.

$\mathbf{S}^i_{\mathtt{TB}}$ denotes the set of final states of all tail-biting paths that the decoder has discovered till iteration i, then we have $\mathbf{S}^i_{\mathtt{TB}} \subseteq \mathbf{S}_L$. $A \setminus B$ denotes the relative complement of B in A.

Lemma 1: After the first i iterations, to reduce searching candidates in the following iterations, any state s, who meets either of the two conditions below can be excluded from the set of starting states $\mathbf{S}_L$ (=$\mathbf{S}_0$):

1) $s \in \mathbf{S}^i_{\mathtt{TB}}$;
2) $s \in \mathbf{S}_L \setminus \mathbf{S}^i_{\mathtt{TB}}$ and $M_{\mathtt{state,net}}(s) < M^{\mathtt{O}}_{\mathtt{MLTBP}}$.

The proof of Lemma 1 has been presented in Appendix A. Lemma 1 illuminates a way to find the global optimal tail-biting path without performing exhaustive searches. Based on these, the proposed trap detection based ML decoding algorithm can be described as follows.

We use variable c to count the number of states whose net state metric is greater than $M^{\mathtt{O}}_{\mathtt{MLTBP}}$ in the ith iteration. Notation "$\leftarrow$" is used to assign the value from the right hand side to the left hand side in algorithm description.

Trap Detection based ML decoder

step 1: Initialization: $M^1_{\mathtt{path},0}(s) \leftarrow 0$, $M_{\mathtt{state,net}}(s) \leftarrow +\infty$, $\forall s \in \mathbf{S_0}$; $sum(0) \leftarrow +\infty$ and $M^{\mathtt{O}}_{\mathtt{MLTBP}} \leftarrow 0$.

step 2: For iteration i, $i \ge 1$:

2.1 Initialize $c \leftarrow 0$, perform modified Viterbi algorithm, find $P^i_{\mathtt{MLP}}(\beta(s), s)$ and $P^i_{\mathtt{MLTBP}}(s', s')$;

2.2 If $M^i_{\mathtt{MLTBP,net}}(s', s') > M^{\mathtt{O}}_{\mathtt{MLTBP}}$, update ($P^{\mathtt{O}}_{\mathtt{MLTBP}}$, $M^{\mathtt{O}}_{\mathtt{MLTBP}}$); if $i = 1$, $M_{\mathtt{state,net}}(s) \leftarrow M^1_{\mathtt{path},L}(s)$, $\forall s \in \mathbf{S}_L$;

2.3 Update the net state metric of each state s as follows:

for $\forall s \in \mathbf{S}_L$ **do**
if $M_{\mathtt{state,net}}(s) \le M^{\mathtt{O}}_{\mathtt{MLTBP}}$ **then**

$$M_{\mathtt{state,net}}(s) \leftarrow 0, \tag{3}$$

$$M^{i+1}_{\mathtt{path},0}(s) \leftarrow 0. \tag{4}$$

else
$M^{i+1}_{\mathtt{path},0}(s) \leftarrow M^i_{\mathtt{path},L}(s)$, $c{+}{+}$.
if $M_{\mathtt{state,net}}(s) > M^i_{\mathtt{path},L}(s) - M^i_{\mathtt{path},0}(s)$ **then**
$M_{\mathtt{state,net}}(s) \leftarrow M^i_{\mathtt{path},L}(s) - M^i_{\mathtt{path},0}(s)$.
end if

end if
end for
$sum(i) \leftarrow c$.

2.4 Special controls: if $sum(i) = 0$, go to step 3; else if $sum(i) = sum(i-1)$, perform Viterbi algorithm on the sub-tail-biting trellis of state $\beta(s)$ and obtain tail-biting path $P_{\mathtt{TBP}}(\beta(s), \beta(s))$ and its net path metric $M_{\mathtt{TBP}}(\beta(s), \beta(s))$; do

$$M^{i+1}_{\mathtt{path},0}(\beta(s)) \leftarrow 0, M_{\mathtt{state,net}}(\beta(s)) \leftarrow 0, \quad (5)$$

and update ($P^{\circ}_{\mathtt{MLTBP}}$, $M^{\circ}_{\mathtt{MLTBP}}$) if $M_{\mathtt{TBP}}(\beta(s), \beta(s)) > M^{\circ}_{\mathtt{MLTBP}}$;

2.5 $i{+}{+}$, and go back to step 2.

step 3: Stop decoding and output the codeword associated with path $P^{\circ}_{\mathtt{MLTBP}}$.

This proposed scheme is based on Viterbi algorithm. The decoding process is straightforward and the memory requirement is comparable in size of memory space required by Viterbi algorithm. Comparing to other ML decoders, the proposed algorithm can greatly save the computational complexity. In step 2.3 of the above algorithm, we can reduce searching candidates by excluding state s with net state metric not exceeding $M^{\circ}_{\mathtt{MLTBP}}$ from the set of starting states according to Lemma 1. In step 2.4, we take special control on the decoding process of CVA when condition $sum(i) = sum(i-1)$ is detected. In this situation, a normal Viterbi trail is performed on the sub-tail-biting trellis of state $\beta(s)$. When the ML tail-biting path $P_{\mathtt{TBP}}(\beta(s), \beta(s))$ is obtained, the state $\beta(s)$ can also be excluded from the searching set $\mathbf{S}_L$. With this special control, the decoder can converge to global optimal solution eventually. Based on these, we can now prove that the decoding algorithm presented above is optimal.

Theorem 1: The decoder presented above from step 1 to step 3 is an ML decoder for tail-biting codes.

Proof: From the description of the algorithm above, we find that the decoding process can only be terminated by the condition $sum(i) = 0$. Consequently, the optimality of this decoder can be concluded by demonstrating the following two points:

1) function $sum(i)$ can converge to zero within finite iterations;
2) when the decoding process is terminated by condition $sum(i) = 0$, $P^{\circ}_{\mathtt{MLTBP}}$ records the optimal tail-biting path among all tail-biting paths in all sub-tail-biting trellises.

Firstly, we divide the set $\mathbf{S}_L \setminus \mathbf{S}^i_{\mathtt{TB}}$ to $\underline{\mathbf{S}}^i$ and $\overline{\mathbf{S}}^i$, where $\underline{\mathbf{S}}^i$ denotes the set of states with net state metric not exceeding $M^{\circ}_{\mathtt{MLTBP}}$ in $\mathbf{S}_L \setminus \mathbf{S}^i_{\mathtt{TB}}$, and $\overline{\mathbf{S}}^i$ denotes the set of states with net state metric larger than $M^{\circ}_{\mathtt{MLTBP}}$ in $\mathbf{S}_L \setminus \mathbf{S}^i_{\mathtt{TB}}$. Their relations are: $\underline{\mathbf{S}}^i \cup \overline{\mathbf{S}}^i = \mathbf{S}_L \setminus \mathbf{S}^i_{\mathtt{TB}}$, and $\underline{\mathbf{S}}^i \cap \overline{\mathbf{S}}^i = \varnothing$.

Denote by $|\overline{\mathbf{S}}^i|$ the size of $\overline{\mathbf{S}}^i$. For any state $s' \in \mathbf{S}^i_{\mathtt{TB}}$, its net state metric will be assigned to zero and remain unchanged since then. Consequently, we have,

$$sum(i-1) = |\overline{\mathbf{S}}^{i-1}|, \quad (6)$$

$$sum(i) = |\overline{\mathbf{S}}^{i}|. \quad (7)$$

For any state $s^{\dagger} \in \underline{\mathbf{S}}^{i-1}$, the decoder deletes it from the set of starting states by equation (4) in the ith iteration. So there is no survivor path that starts from any state of the set $\underline{\mathbf{S}}^{i-1}$ in the ith iteration. The net state metric of any state in $\underline{\mathbf{S}}^{i-1}$ will be unchanged since then. Hence, the states in $\underline{\mathbf{S}}^{i-1}$ don't influence the value of $sum(i)$, i.e., $\underline{\mathbf{S}}^{i-1} \cap \overline{\mathbf{S}}^{i} = \varnothing$.

As a result, the value of $sum(i)$ is only affected by the state $s \in \overline{\mathbf{S}}^{i-1}$. From the definition of $M_{\mathtt{state,net}}(s)$ in equation (2), we can conclude that $M_{\mathtt{state,net}}(s)$ is a non-increasing function of i. While from the updating procedure of $M^{\circ}_{\mathtt{MLTBP}}$, we know that $M^{\circ}_{\mathtt{MLTBP}}$ is non-decreasing function of i. Consequently, we have:

$$\overline{\mathbf{S}}^{i} \subseteq \overline{\mathbf{S}}^{i-1}. \quad (8)$$

Combining (8) with equation (6) and (7), we have: $sum(i) \leq sum(i-1)$. i.e., $sum(i)$ is a non-increasing function of i.

Based on the definition of $sum(i)$ and equation (7), the maximum possible value of function $sum(i)$ is 2^v, where v is the length of the encoder registers. If $sum(i) < sum(i-1)$ holds for all $i > 1$, then $sum(i)$ converges to zero within finite iterations. Therefore, we only need to check that how the optimal decoder guarantees $sum(i)$ to converge to zero in the case of $sum(i) = sum(i-1)$. In this condition, $\overline{\mathbf{S}}^{i-1} = \overline{\mathbf{S}}^{i}$. The ML path of iteration i is $P^i_{\mathtt{MLP}}(\beta(s), s)$, and $\beta(s)$ must be a state in $\overline{\mathbf{S}}^{i-1}$. In step 2.4) we get a tail-biting path $P_{\mathtt{TBP}}(\beta(s), \beta(s))$. If $M_{\mathtt{TBP}}(\beta(s), \beta(s)) \leq M^{\circ}_{\mathtt{MLTBP}}$, the state $\beta(s)$ will be incorporated into $\underline{\mathbf{S}}^{i}$ by equation (5), i.e., $M^{i+1}_{\mathtt{path},0}(\beta(s)) \leftarrow 0, M_{\mathtt{state,net}}(\beta(s)) \leftarrow 0$. Then, by special control in step 2.4), at least one state can be deleted from the set of starting state candidates.

From above analysis, we have:

$$\overline{\mathbf{S}}^{i+1} \subsetneqq \overline{\mathbf{S}}^{i}. \quad (9)$$

Combining equation (6), (7) and (9), we have,

$$sum(i) - sum(i+1) \geqslant 1. \quad (10)$$

If $M_{\mathtt{TBP}}(\beta(s), \beta(s)) > M^{\circ}_{\mathtt{MLTBP}}$, the path $P^{\circ}_{\mathtt{MLTBP}}$ is updated with the new tail-biting path $P_{\mathtt{TBP}}(\beta(s), \beta(s))$, and $M^{\circ}_{\mathtt{MLTBP}}$ is updated with $M_{\mathtt{TBP}}(\beta(s), \beta(s))$. The state $\beta(s) \in \overline{\mathbf{S}}^{i}$ can be placed into the set $\underline{\mathbf{S}}^{i}$ since the best tail-biting path on its sub-tail-biting trellis has been found and recorded in $P^{\circ}_{\mathtt{MLTBP}}$. In the $(i+1)$th iteration, all states whose net state metrics not exceeding $M^{\circ}_{\mathtt{MLTBP}}$ can be excluded from the set $\overline{\mathbf{S}}^{i}$. Similarly, we have $\overline{\mathbf{S}}^{i+1} \subsetneqq \overline{\mathbf{S}}^{i}$ and $sum(i) - sum(i+1) \geqslant 1$. From the above analysis, we know that in the case of $sum(i) = sum(i-1)$, $sum(i+1) < sum(i)$ always holds. Therefore, $sum(i)$ will converge to zero after finite iteration.

The convergent process indicates that every state s in the set $\overline{\mathbf{S}}^{i}$ shall be placed into the set $\underline{\mathbf{S}}^{j}$ or $\mathbf{S}^{j}_{\mathtt{TB}}$ as iterations continue, where $j > i$. With Lemma 1, when state s is excluded from the set $\overline{\mathbf{S}}^{i}$, there is no tail-biting path better than $P^{\circ}_{\mathtt{MLTBP}}$ on the sub-tail-biting trellis of state s. Similarly, we can conclude that when $sum(i) = 0$, there is no tail-biting path in any sub-tail-biting trellis of any possible starting state having larger net path metric than $M^{\circ}_{\mathtt{MLTBP}}$. In other words, $P^{\circ}_{\mathtt{MLTBP}}$ must be the global optimum tail-biting path. In summary, the trap detection based decoder presented above is an ML decoder for tail-biting codes that can converge to the global optimum result within finite iterations. ■

To illustrate the optimal decoding process, we can apply this algorithm to the case shown in Example 1. After the second iteration, we obtain the ML tail-biting path $P^{\circ}_{\mathtt{MLTBP}} = P^{2}_{\mathtt{MLTBP}}$ with net path metric $M^{\circ}_{\mathtt{MLTBP}} = 9.703$ and net state metrics $\{9.703, 9.521, 9.521, 9.831\}$. According to equation (3) and (4), we can initialize the accumulated path metric of paths which start from states $s_0(1)$ and $s_0(2)$ to zero for the 3rd iteration. In the 3rd iteration, the four survivor paths are $P^3_0(s_0(0), s_8(0))$, $P^3_1(s_0(0), s_8(1))$, $P^3_2(s_0(0), s_8(2))$ and $P^3_3(s_0(0), s_8(3))$, and the net state metrics are $\{9.703, 0, 0, 9.485\}$. Consequently, we have $sum(3) = 0$. The decoder is terminated after iteration 3 and outputs the decoding result recorded in the path $P^{\circ}_{\mathtt{MLTBP}}$. While for the conventional ML decoder, we need to perform four independent Viterbi trials on each sub-tail-biting trellis. The net path metrics of the tail-biting paths are $\{9.703, 9.117, 9.419, 7.405\}$. In this example, our decoder requires three iterations to find the ML tail-biting path. As the length of encoder registers becomes longer, the proposed optimal decoder has more advantages in terms of the memory size and decoding efficiency than other ML decoding schemes.

IV. Simplified Trap Detection Based Decoder

Comparing to the conventional ML decoder, the proposed decoder retains the optimality while greatly reduces the complexity. On the other hand, the decoding process is still a little bit complex when calculating and updating the net state metric of each final state for every iteration as well as applying these values to detect circular traps. For real-time implementation, we can further simplify the circular traps checking rule by comparing the ML tail-biting path obtained in current iteration: $P^{i}_{\mathtt{MLTBP}}$ with the optimal tail-biting path obtained in previous iterations: $P^{\circ}_{\mathtt{MLTBP}}$. A circular trap occurs when the following equation holds:

$$P^{i}_{\mathtt{MLTBP}} = P^{\circ}_{\mathtt{MLTBP}}. \tag{11}$$

Equation (11) is obtained directly from the definition of circular traps.

If the net path metric rather than the path itself is used to detect the circular trap, the decoder can avoid comparing each branch of the two paths: $P^{i}_{\mathtt{MLTBP}}$ and $P^{\circ}_{\mathtt{MLTBP}}$, and can further reduce the computational complexity. The circular trap checking rule can be summarized as:

$$M^{i}_{\mathtt{MLTBP,net}} = M^{\circ}_{\mathtt{MLTBP,net}}. \tag{12}$$

Equation (12) can be used to detect the circular trap and terminate the decoding process timely. On the other hand, Equation (11) and (12) cannot be used to control the convergence of the decoding process, in other words, the decoder based on circular traps checking rule in equation (11) and (12) are suboptimal. There may be no tail-biting path discovered when the decoding process is terminated. In this case the decoder outputs the path $P^{\circ}_{\mathtt{MLP}}$ as the decoding result.

In existing conventional sub-optimal algorithms, decoders try to update ML path and ML tail-biting path after every iteration. While in fact, the path $P^{1}_{\mathtt{MLP}}$ has the largest net path metric among all survivor paths obtained in all iterations, i.e., $P^{\circ}_{\mathtt{MLP}} = P^{1}_{\mathtt{MLP}}$. Consequently, we omit the updating step of $P^{\circ}_{\mathtt{MLP}}$ in the following proposed algorithm. In addition, a new stopping rule is adopted: $M^{i}_{\mathtt{MLP,net}} = M^{i}_{\mathtt{MLTBP,net}}$, which is more efficient than the rule $P^{\circ}_{\mathtt{MLP}} = P^{\circ}_{\mathtt{MLTBP}}$ that has been used in other schemes, such as in WAVA [1]. We summarized these two properties in Theorem 2 and Theorem 3:

Theorem 2: In the CVA decoding process, among all survivor paths obtained in all iterations, the MLP obtained in the first iteration has the largest net path metric.

Theorem 3: For the CVA, the equation $P^{\circ}_{\mathtt{MLP}} = P^{\circ}_{\mathtt{MLTBP}}$ can be fulfilled if and only if $P^{1}_{\mathtt{MLP}} = P^{1}_{\mathtt{MLTBP}}$.

The proofs of these two theorems are included in Appendix A.

Based on these observations, we present a simplified trap detection based decoding scheme as below:

Simplified Trap Detection based decoder

step 1: Initialization: $M^{1}_{\mathtt{path},0}(s) \leftarrow 0$, $\forall s \in \mathbf{S}_0$; perform modified Viterbi algorithm; find ($P^{1}_{\mathtt{MLP}}$, $M^{1}_{\mathtt{MLP,net}}$) and ($P^{1}_{\mathtt{MLTBP}}$, $M^{1}_{\mathtt{MLTBP,net}}$). If $M^{1}_{\mathtt{MLP,net}} = M^{1}_{\mathtt{MLTBP,net}}$, go to step 3; otherwise, record $P^{1}_{\mathtt{MLP}}$, and if $M^{1}_{\mathtt{MLTBP,net}} > 0$, ($P^{\circ}_{\mathtt{MLTBP}}$, $M^{\circ}_{\mathtt{MLTBP}}$) $\leftarrow$ ($P^{1}_{\mathtt{MLTBP}}$, $M^{1}_{\mathtt{MLTBP,net}}$).

step 2: For iteration i, $i > 1$:

2.1 Apply the value $M^{i-1}_{\mathtt{path},L}(s)$ to initialize the accumulated path metric of the path started from state s, i.e., $M^{i}_{\mathtt{path},0}(s) \leftarrow M^{i-1}_{\mathtt{path},L}(s)$;

2.2 Perform modified Viterbi algorithm, find $M^{i}_{\mathtt{MLP,net}}$ and ($P^{i}_{\mathtt{MLTBP}}, M^{i}_{\mathtt{MLTBP,net}}$);

2.3 If $M^{i}_{\mathtt{MLTBP,net}} > M^{\circ}_{\mathtt{MLTBP}}$, set $P^{\circ}_{\mathtt{MLTBP}} \leftarrow P^{i}_{\mathtt{MLTBP}}$;

2.4 If $M^{i}_{\mathtt{MLTBP,net}} = M^{\circ}_{\mathtt{MLTBP}}$ or $M^{i}_{\mathtt{MLP,net}} = M^{i}_{\mathtt{MLTBP}}$, stop decoding process and go to step 3; otherwise, go to step 2.5;

2.5 If $M^{i}_{\mathtt{MLTBP,net}} > M^{\circ}_{\mathtt{MLTBP}}$, set $M^{\circ}_{\mathtt{MLTBP}} \leftarrow M^{i}_{\mathtt{MLTBP,net}}$. Go back to step 2 until the maximum allowed number of iterations N is reached.

step 3: If path $P^{\circ}_{\mathtt{MLTBP}}$ exists, output the codeword associated with path $P^{\circ}_{\mathtt{MLTBP}}$; otherwise, output the codeword associated with path $P^{1}_{\mathtt{MLP}}$.

Comparing with other decoding schemes, this algorithm has three main advantages with our algorithm. First of all, the circular trap checking rule removes redundant iterations in CVA. Secondly, the decoder omits the updating step of MLP after the first iteration according to Theorem 2. Finally, the new stopping rule adopted in step 2.4 is more efficient than the rule that has been used in other schemes, such as in WAVA [1]. The following example demonstrates the effectiveness of the new stopping rule.

Example 2: We use the same tail-biting codes as that used in Example 1. For the information sequence $\{10010100\}$, the transmitted codeword is $\{(11), (10), (11), (11), (10), (00), (10), (11)\}$. After passing through the AWGN channel with $E_b/N_0 = 0$ dB, the received sequence (for one realization) is {(-1.409 -0.283), (0.276 -0.159), (-1.914 -0.561), (0.109 0.179), (-1.208 -0.708), (-1.273 -2.074), (-0.842 0.716), (-0.781 -0.176)}. In the 1st iteration, we obtain four survivor paths P^1_0, P^1_1, P^1_2 and P^1_3 with net path metrics $\{12.132, 8.326, 10.218, 9.204\}$. The MLP is $P^{1}_{\mathtt{MLP}} = P^1_0$ with net path metric $M^{1}_{\mathtt{MLP,net}} = 12.132$ and the MLTBP is $P^{\circ}_{\mathtt{MLTBP}} = P^{1}_{\mathtt{MLTBP}} = P^1_3$ with net path metric $M^{\circ}_{\mathtt{MLTBP}} = M^{1}_{\mathtt{MLTBP,net}} = 9.204$. In the 2nd iteration, we obtain four survivor paths

P_0^2, P_1^2, P_2^2 and P_3^2 with net path metrics {11.188, 8.022, 9.274, 8.900}. The path P_0^2 is the MLP, which is a tail-biting path as well, where P_0^2={$s_0(0)$, $s_1(2)$, $s_2(3)$, $s_3(3)$, $s_4(1)$, $s_5(0)$, $s_6(2)$, $s_7(1)$, $s_8(0)$}. Since $M^2_{\mathtt{MLP,net}} = M^2_{\mathtt{MLTBP,net}} = 11.188 > M^{\mathtt{O}}_{\mathtt{MLTBP}} = 9.204$, we update $P^{\mathtt{O}}_{\mathtt{MLTBP}}$ with P_0^2 and $M^{\mathtt{O}}_{\mathtt{MLTBP}}$ with 11.188. The stopping rule $M^i_{\mathtt{MLP,net}} = M^i_{\mathtt{MLTBP,net}}$ can terminate the decoding process, while with the rule in [1], i.e. $P^{\mathtt{O}}_{\mathtt{MLP}} = P^{\mathtt{O}}_{\mathtt{MLTBP}}$, we were not able to stop decoding process, since $P^{\mathtt{O}}_{\mathtt{MLP}} \neq P^{\mathtt{O}}_{\mathtt{MLTBP}}$. In the meanwhile, we can find that the MLP obtained from the 1st iteration has the largest net path metric.

V. Simulation Results

In order to show the decoding accuracy and decoding efficiency, we compare the proposed trap detection based ML decoder and suboptimal decoder with other widely used tail-biting decoders in the following three examples. In these examples, the coded information bits are modulated to QPSK symbols, and then passed through an AWGN channel. The results shown in figures are obtained by observing at least 100 block error events. For convenience, we call the decoding scheme in [14] the two-phase decoder.

Golay code can be represented by either tail-biting trellis or conventional trellis. In the first example, we examine the performance of different ML decoders for (24, 12) binary Golay code [7], [12], which is often used as a benchmark in studies of code structure and decoding algorithms. Three different trellis structures of (24, 12) Golay codes are used for different decoders. The first one is a 64-state tail-biting trellis obtained from octal generator polynomials of {103, 166} [7], the second one is a 16-state tail-biting trellis [11], and the last one is a conventional trellis in the minimum span form deduced directly from its generator matrix [12].

Table II shows the block error rate (BLER) performance of different decoders. The ML decoder in Table II refers to any kind of ML decoders mentioned in this paper: trap detection based ML decoder, two-phase ML decoder or BEAST decoder, since all of them exhibit exactly the same block error rate performance. We find that the suboptimal decoders, such as WAVA or the simplified trap detection based decoder have similar performance and are all close to optimal.

For a complete comparison, we check computational complexity and average memory space consumption of different ML decoding algorithms. The average number of visited states per decoded bit has been selected as metric to measure the decoding complexity. Fig. 2 illustrates computational complexity of the above three algorithms on different trellis structures. We find that, on tail-biting trellises, the trap detection based decoder exhibits excellent decoding efficiency. The number of visited states of the two-phase decoder drops as the SNR increases [1], and at high SNR region, two-phase decoder exhibits almost the same performance as the trap detection

[1]The number of visited states during the decoding process of the two-phase decoder consists of three parts: the number of states visited by Viterbi searches on tail-biting trellis during the first phase, the number of states visited by heuristic searches on tail-biting trellis during the second phase, and the number of states visited in sorting the open stack and in retrieving the close table in the second phase [14]. The last part plays a dominant role in complexity calculation at low SNR region.

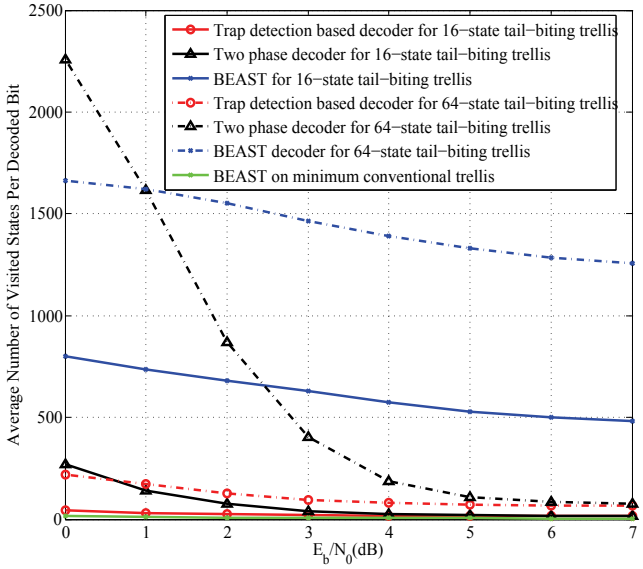


Fig. 2. A comparison of decoding complexity of different ML decoders: trap detection based ML decoder, two-phase ML decoder and BEAST ML decoder on the 16-state tail-biting trellis, 64-state tail-biting trellis, and minimum conventional trellis of (24, 12) Golay code.

based decoder. The results of the trap detection based decoder and the two phase decoder on 16-state and 64-state tail-biting trellises show that a well-constructed tail-biting trellis that has lower state complexity can significantly improve the decoding efficiency. Meanwhile, we find that BEAST works with high complexity on tail-biting trellis. On the other hand, the Golay codes can also be represented by a minimal span conventional trellis. The performance of BEAST on such trellis has very low complexity as shown in Fig. 2 [12]. For each decoded bit, only 7.9 states and 4.6 states are visited at E_b/N_0=2dB and 7dB respectively. In this case, BEAST works very well on minimal conventional trellis and outperforms the proposed trap detection based algorithm.

Fig. 3 compares memory space required of different ML decoders. The average memory space required for trap detection based decoder is a constant. Comparing to the two-phase decoder, trap detection based decoder consumes less than $1/2$ of the memory space. This is due to the fact that the two-phase decoder has to store path metrics of all survivor paths at each section obtained from the first phase for the second phase decoding. Meanwhile, BEAST requires almost 3 times more memory space while decoding on tail-biting trellis.

Combining these observations from the first example, we come to the conclusion that trap detection based ML decoder is excellent in saving memory space and reducing computational complexity on tail-biting trellis.

In the following example, we examine the performance of different decoders for long and medium tail-biting codes on their tail-biting trellises [1]. The (120, 40) tail-biting convolutional codes with octal generator polynomials of {133, 171, 165} [5] is a long tail-biting codes, which has payload size of 40 bits corresponding to the minimal encoding block size in physical broadcasting channel of LTE. Since the length of encoder registers is 6, the tail-biting encoding provides 13% rate saving comparing to the known-tail encoding. The

TABLE II
THE BLOCK ERROR RATE PERFORMANCE OF WAVA, SIMPLIFIED TRAP DETECTION BASED DECODER AND ML DECODERS (TRAP DETECTION ML DECODER, TWO-PHASE ML DECODER OR BEAST ML DECODER) FOR THE (24, 12) GOLAY CODE OVER THE AWGN CHANNEL.

BLER performance	**SNR**				
	1.0 dB	2.0 dB	3.0 dB	4.0 dB	5.0 dB
WAVA	1.392×10^{-1}	5.187×10^{-2}	1.328×10^{-2}	2.036×10^{-3}	1.632×10^{-4}
Simplified Trap Detection based decoder	1.404×10^{-1}	5.279×10^{-2}	1.375×10^{-2}	2.166×10^{-3}	1.813×10^{-4}
ML decoders	1.346×10^{-1}	4.992×10^{-2}	1.275×10^{-2}	1.963×10^{-3}	1.596×10^{-4}

TABLE III
THE BLOCK ERROR RATE PERFORMANCE OF WAVA, SIMPLIFIED TRAP DETECTION BASED DECODER AND ML DECODERS (TRAP DETECTION ML DECODER, TWO-PHASE ML DECODER OR BEAST DECODER) FOR THE TAIL-BITING CONVOLUTIONAL CODE (120, 40) WITH GENERATOR POLYNOMIALS OF $\{133, 171, 165\}$ OVER THE AWGN CHANNEL.

BLER performance	**SNR**				
	1.0 dB	2.0 dB	3.0 dB	4.0 dB	5.0 dB
WAVA	8.535×10^{-2}	1.372×10^{-2}	1.353×10^{-3}	7.280×10^{-5}	1.960×10^{-6}
Simplified Trap Detection based decoder	8.598×10^{-2}	1.377×10^{-2}	1.352×10^{-3}	7.280×10^{-5}	1.960×10^{-6}
ML decoders	8.466×10^{-2}	1.363×10^{-2}	1.352×10^{-3}	7.280×10^{-5}	1.960×10^{-6}

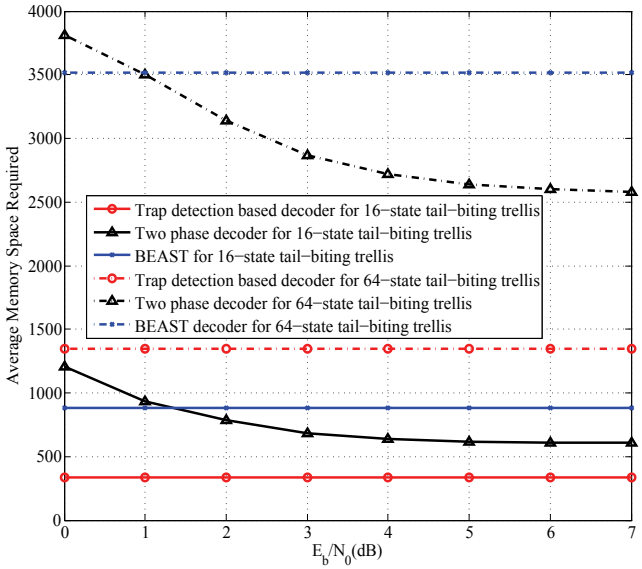


Fig. 3. Average memory space required for trap detection based ML decoder, two-phase ML decoder and BEAST ML decoder on the 16-state and 64-state tail-biting trellises of (24, 12) Golay codes.

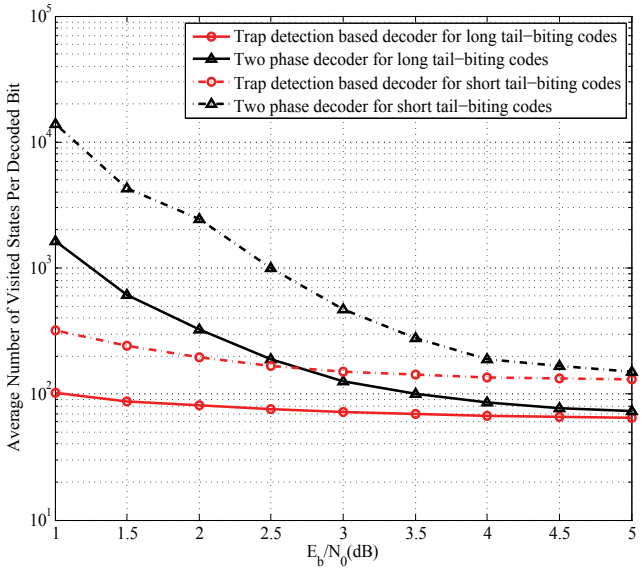


Fig. 4. A comparison of decoding complexity of trap detection based ML decoder and two-phase ML decoder for decoding (120, 40) long tail-biting codes and for decoding (64, 32) medium tail-biting codes.

example of short tail-biting codes is (64, 32) tail-biting convolutional codes with octal generator polynomials of $\{345, 237\}$ [1].

Fig. 4 and Fig. 5 show the decoding complexity and average memory space required of different decoders for decoding medium and long tail-biting codes. Fig. 4 shows that, comparing with the two phase decoder, the trap detection based decoder exhibits lower decoding complexity for both medium and long tail-biting codes. From Fig. 5, we observe that the trap detection based decoder consumes less memory space than that of two-phase decoder.

In the last example, we check the performance of the simplified trap detection based decoder. Compared with the trap detection based ML decoder, its simplified scheme exhibits almost the same decoding complexity in terms of the average number of visited states per decoded bit. However, the simplified scheme works with no ML path identification at iteration i, where $i > 1$, and with no net state metric calculations, updates and comparisons after each iteration. Consequently, the decoding process is simplified. This advantage can be significant as the length of encoder registers becomes longer. The average number of iterations is calculated to measure the decoding efficiency of sub-optimal decoders: WAVA and the simplified trap detection based decoder, where the maximum allowed number of iteration is 20 for these two sub-optimal decoders. The result is illustrated in Fig. 6, which demonstrates that the simplified trap detection based decoder is more efficient than WAVA due to circular traps detection rule and the new stopping rule.

Table III shows the block error rate performance of WAVA,

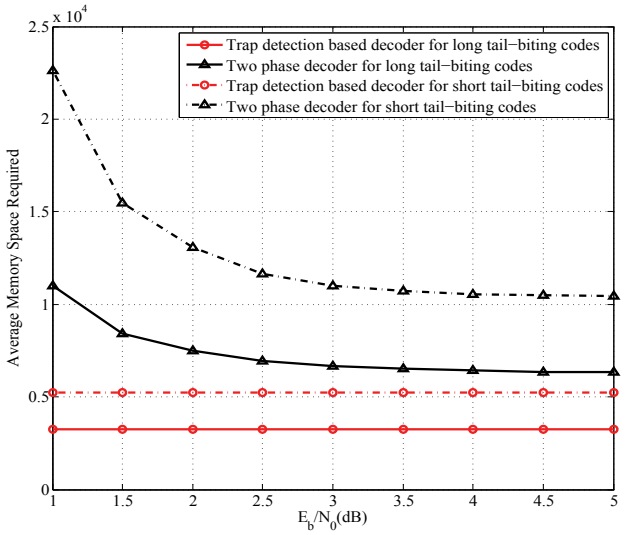


Fig. 5. Average memory space required for trap detection based ML decoder and two-phase ML decoder for decoding (120, 40) long tail-biting codes and for decoding (64, 32) medium tail-biting codes.

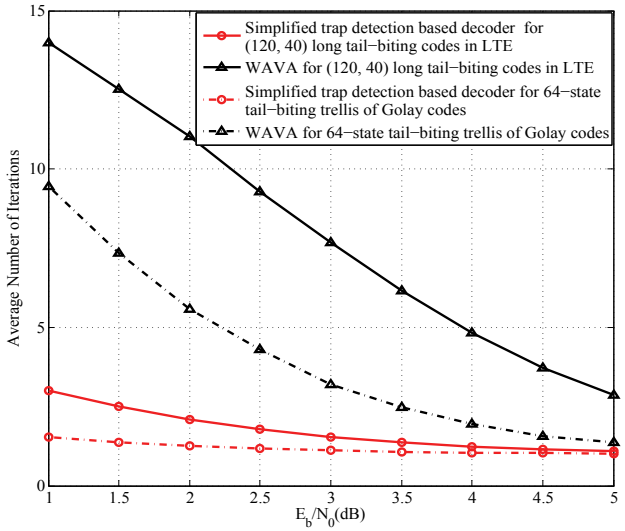


Fig. 6. Average number of iterations needed for WAVA and simplified trap detection based decoder for (24, 12) Golay code on its 64-state tail-biting trellis and (120, 40) tail-biting convolutional codes with generator polynomials of {133, 171, 165}.

simplified trap detection based decoder and ML decoders (all three ML decoders). We find that both of the two suboptimal decoders exhibit near-optimal performance.

VI. Conclusions

Viterbi algorithm is a natural choice for tail-biting codes. However, there is no optimal decoder based on the Viterbi algorithm. Circular traps existing in the decoding process of CVA significantly reduce the decoding efficiency. An efficient checking rule has been proposed to detect the circular trap, which can be used to take special controls on the decoding process of the CVA when circular trap occurs. We propose a trap detection based ML decoder to take advantage of the circular trap detection. Comparing to other optimal decoders, the proposed optimal decoder exhibits fast convergence speed and low decoding complexity. In addition, the memory space requirement of our decoder is also the lowest among all optimal decoders. The proposed ML decoder is attractive in field application.

To further reduce the decoding complexity, we simplify the circular trap checking rule and stopping rule. With these modifications, a simplified trap detection based decoder is proposed. This algorithm can provide near optimal error performance with much lower computational complexity and higher decoding efficiency than other sub-optimal decoders such as WAVA.

Appendix A

Lemma 1: After the first i iterations, to reduce searching candidates in the following iterations, any state s, which meets either of the two conditions below can be excluded from the set of starting states $\mathbf{S}_L$ (=$\mathbf{S}_0$):

1) $s \in \mathbf{S}^i_{\mathtt{TB}}$;
2) $s \in \mathbf{S}_L \setminus \mathbf{S}^i_{\mathtt{TB}}$ and

$$M_{\mathtt{state,net}}(s) < M^{\mathtt{O}}_{\mathtt{MLTBP}}. \tag{13}$$

Proof: For any state $s \in \mathbf{S}^i_{\mathtt{TB}}$, there exists a tail-biting survival path in the the sub-tail-biting trellis of state s in the first i iterations. Without loss of generality, we assume that the associated tail-biting path is $P^j(s,s)$ and its net path metric is $M^j_{\mathtt{path,net}}(s,s)$. It is obtained in the jth iteration, where $j \le i$. The Lemma 1 of [1] states that $P^j(s,s)$ has the largest net path metric on the sub-tail-biting trellis of state s, since all paths in this sub-tail-biting trellis have the same initial path metric at certain iteration while only $P^j(s,s)$ survives. Consequently, if $M^j_{\mathtt{path,net}}(s,s) \le M^{\mathtt{O}}_{\mathtt{MLTBP}}$, state s can be deleted safely from the set of starting state candidates. If $M^j_{\mathtt{net}}(s,s) > M^{\mathtt{O}}_{\mathtt{MLTBP}}$, the path $P^j(s,s)$ is recorded in $P^{\mathtt{O}}_{\mathtt{MLTBP}}$. Since there is no tail-biting path in the sub-tail-biting trellis of the state s with larger net path metric than the path $P^j(s,s)$, which has already been recorded in $P^{\mathtt{O}}_{\mathtt{MLTBP}}$, state s can be deleted from $\mathbf{S}_L$. In summary, it is safe to delete any state $s \in \mathbf{S}^i_{\mathtt{TB}}$ from the set of starting states.

From the Lemma 1 in [1], we know that for any state $s \in \mathbf{S}_L \setminus \mathbf{S}^i_{\mathtt{TB}}$, the paths in its sub-tail-biting trellis have net path metrics not exceeding $M^1_{\mathtt{path},L}(s)$ in the first iteration where $M^1_{\mathtt{path},0}(s) = 0$, $M^2_{\mathtt{path},L}(s) - M^2_{\mathtt{path},0}(s)$ in the 2nd iteration, $\ldots$, and $M^i_{\mathtt{path},L}(s) - M^i_{\mathtt{path},0}(s)$ in ith iteration, i.e.,

$$\begin{aligned} M_{\mathtt{path,net}}(s,s) \le \min\{&M^1_{\mathtt{path},L}(s), \\ &M^2_{\mathtt{path},L}(s) - M^2_{\mathtt{path},0}(s), \ldots, \\ &M^i_{\mathtt{path},L}(s) - M^i_{\mathtt{path},0}(s)\}. \end{aligned} \tag{14}$$

Combining the definition of $M_{\mathtt{state,net}}(s)$ in equation (2), equation (14) can be rewritten as

$$M_{\mathtt{path,net}}(s,s) \le M_{\mathtt{state,net}}(s). \tag{15}$$

Equation (15) indicates that any tail-biting path in the sub-tail-biting trellis of state s has net path metric not exceeding the obtained net state metric of state s up to current iteration. Combining (13) and (15), we have, $M_{\mathtt{path,net}}(s,s) < M^{\mathtt{O}}_{\mathtt{MLTBP}}$.

Consequently, the ML tail-biting path does not locate in the sub-tail-biting trellis of state s. State s can also be excluded from the set of starting states. ■

Theorem 2: In the CVA decoding process, among all survivor paths obtained in all iterations, the MLP obtained in the first iteration has the largest net path metric.

Proof: Let us denote by $P^1_{\mathtt{MLP}}(\beta^1(s), s)$ the MLP obtained in the first iteration. The accumulated path metric is $M^1_{\mathtt{path},L}(s)$. Since the state metric is initialized with zero, the corresponding net path metric is $M^1_{\mathtt{MLP,net}}(\beta^1(s), s) = M^1_{\mathtt{path},L}(s)$. For the ith iteration, the net path metric of the survivor path $P^i(\beta^i(s'), s')$ is

$$M^i_{\mathtt{path,net}}(\beta^i(s'), s') = M^i_{\mathtt{path},L}(s') - M^i_{\mathtt{path},0}(\beta^i(s')). \tag{16}$$

The MLP in this iteration is given by $P^i_{\mathtt{MLP}}(\beta^i(s^\dagger), s^\dagger)$ with a net path metric $M^i_{\mathtt{MLP,net}}(\beta^i(s^\dagger), s^\dagger))$, where $s, s', s^\dagger \in \mathbf{S}_L$. According to the definition of the MLP,

$$M^i_{\mathtt{path,net}}(\beta^i(s'), s') \le M^i_{\mathtt{MLP,net}}(\beta^i(s^\dagger), s^\dagger). \tag{17}$$

Let us apply proof by contradiction. Assume that

$$M^i_{\mathtt{MLP,net}}(\beta^i(s^\dagger), s^\dagger) > M^1_{\mathtt{MLP,net}}(\beta^1(s), s), \tag{18}$$

where $i > 1$. From equation (17), for the 1st iteration, we have

$$M^1_{\mathtt{path,net}}(\beta^1(s^\dagger), s^\dagger) \le M^1_{\mathtt{MLP,net}}(\beta^1(s), s). \tag{19}$$

Combining (18) and (19), we get,

$$M^i_{\mathtt{MLP,net}}(\beta^i(s^\dagger), s^\dagger) > M^1_{\mathtt{path,net}}(\beta^1(s^\dagger), s^\dagger). \tag{20}$$

The inequality (20) implies that the path $P^i(\beta^i(s^\dagger), s^\dagger)$ has larger net path metric than $P^1(\beta^1(s^\dagger), s^\dagger)$. On the other hand, with the conclusion in [15], we know that the final survivor path for each state in $\mathbf{S}_L$ is the path of the largest metric from any state in $\mathbf{S}_0$ after the first iteration, or

$$M^i_{\mathtt{MLP,net}}(\beta^i(s^\dagger), s^\dagger) \le M^1_{\mathtt{path,net}}(\beta^1(s^\dagger), s^\dagger). \tag{21}$$

Inequality (21) holds for $\forall i \in \mathbb{Z}^+$, and it contradicts with (18). Therefore, the assumption (18) does not hold. We conclude that among all survivor paths obtained in all iterations, the MLP obtained in the first iteration has the largest net path metric. ■

Theorem 3: For the CVA, the equation

$$P^{\mathtt{O}}_{\mathtt{MLP}} = P^{\mathtt{O}}_{\mathtt{MLTBP}} \tag{22}$$

can be fulfilled if and only if

$$P^1_{\mathtt{MLP}} = P^1_{\mathtt{MLTBP}}. \tag{23}$$

Proof: If $P^1_{\mathtt{MLP}} = P^1_{\mathtt{MLTBP}}$, the MLP and the MLTBP that we find in the first iteration are the same. The CVA stops here and the decoding process is successful. Equation $P^{\mathtt{O}}_{\mathtt{MLP}} = P^{\mathtt{O}}_{\mathtt{MLTBP}}$ holds.

If $P^{\mathtt{O}}_{\mathtt{MLP}} = P^{\mathtt{O}}_{\mathtt{MLTBP}}$, we show that $P^1_{\mathtt{MLP}} = P^1_{\mathtt{MLTBP}}$ too. $P^{\mathtt{O}}_{\mathtt{MLP}}$ stores the MLP corresponding to the largest net path metric among all survivor paths from the 1st iteration to current iteration. From Theorem 2, we know that:

$$P^{\mathtt{O}}_{\mathtt{MLP}} = P^1_{\mathtt{MLP}}, \tag{24}$$

The equation above is independent of i. Therefore, equation (22) can be rewritten as

$$P^1_{\mathtt{MLP}} = P^{\mathtt{O}}_{\mathtt{MLTBP}}. \tag{25}$$

Equation (25) tells us that $P^1_{\mathtt{MLP}}$ obtained in the first iteration is identical to the best tail-biting path obtained in the first i iterations. It implies that the path $P^1_{\mathtt{MLP}}$ is a tail-biting path. Since the MLP found in the first iteration has the largest net path metric and it is a tail-biting path, the MLTBP in the first iteration must be $P^1_{\mathtt{MLP}}$ as well. We have

$$P^1_{\mathtt{MLP}} = P^1_{\mathtt{MLTBP}}. \tag{26}$$

Therefore, $P^{\mathtt{O}}_{\mathtt{MLP}} = P^{\mathtt{O}}_{\mathtt{MLTBP}}$ can be fulfilled if and only if $P^1_{\mathtt{MLP}} = P^1_{\mathtt{MLTBP}}$. ■

REFERENCES

[1] R. Shao, S. Lin, and M. Fossorier, "Two decoding algorithms for tailbiting codes," *IEEE Trans. Commun.*, vol. 51, no. 10, pp. 1658–1665, Oct. 2003.

[2] J. Sun and O. Takeshita, "Extended tail-biting schemes for turbo codes," *IEEE Commun. Lett.*, vol. 9, no. 3, pp. 252–254, Mar. 2005.

[3] G. Forney, Jr., M. Grassl, and S. Guha, "Convolutional and tail-biting quantum error-correction codes," *IEEE Trans. Inf. Theory*, vol. 53, no. 3, pp. 865–880, Mar. 2007.

[4] IEEE Std 802.16-2009, "IEEE standard for local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems," May 2009.

[5] 3GPP TS 36.212, "3rd Generation Partnership Project; Technical pecification group radio access network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding (Release 8)," Sept. 2007.

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

[7] P. Stahl, J. Anderson, and R. Johannesson, "Optimal and near-optimal encoders for short and moderate-length tailbiting trellises," *IEEE Trans. Inf. Theory*, vol. 45, no. 7, pp. 2562–2571, Nov. 1999.

[8] I. Bocharova, R. Johannesson, and B. Kudryashov, "Low state complexity block codes via convolutional codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 9, pp. 2022–2030, Sept. 2004.

[9] I. Bocharova, M. Handlery, R. Johannesson, and B. Kudryashov, "Tailbiting codes obtained via convolutional codes with large active distance-slopes," *IEEE Trans. Inf. Theory*, vol. 48, no. 9, pp. 2577–2587, Sept. 2002.

[10] H. Gluesing-Luerssen and E. Weaver, "Linear tail-biting trellises: characteristic generators and the BCJR-Construction," *IEEE Trans. Inf. Theory*, vol. 49, no. 2 pp. 738–751, Feb. 2011.

[11] A. Calderbank, G. Forney, and A. Vardy, "Minimal tailbiting trellises: Golay code and more," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1435–1455, July 1999.

[12] I. E. Bocharova, R. Johannesson, B. D. Kudryashov, and M. Loncar, "BEAST decoding for block codes," *Europ. Trans. Telecomm.*, vol. 15, pp. 297–305, 2004.

[13] P. Shankar, P. Kumar, K. Sasidharan, B. Rajan, and A. Madhu, "Efficient convergent maximum likelihood decoding on tail-biting." Available: http://arxiv.org/abs/cs.IT/0601023.

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

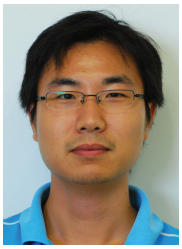
[15] Q. Wang. and V. Bhargava, "An efficient maximum-likelihood decoding algorithm for generalized tailbiting convolutional codes including quasi-cyclic codes," *IEEE Trans. Commun.*, vol. 37, no. 8, pp. 875–879, Aug. 1989.

Xiaotao Wang received the B.S. degree from Anhui Normal University and is currently persuing the Ph.D. degree in Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences. His research interests include coding theory, detection, and synchronization, and their application to communications systems.

Hua Qian received his bachelor's degree and master's degree from Department of Electrical Engineering, Tsinghua University in 1998 and 2000, respectively. He obtained his Ph.D. degree from School of Electrical and Computer Engineering, Georgia Institute of Technology in 2005. From 2005 to 2006, he worked for Chrontel, Inc., San Jose, CA, USA as a digital design engineer. From 2006 to 2010, he worked for Marvell Semiconductor, Inc., Santa Clara, CA, USA as a staff system design engineer. Since 2010, he joined Shanghai Research Center for Wireless Communications, Shanghai Institute of Microsystem and Information Technology Research Institute, Chinese Academy of Sciences as a professor. He was sponsored by the 100 Talents Program of Chinese Academy of Sciences and the Shanghai Pujiang Talent Program. His current research interests include physical layer algorithms and system design of wireless communications, nonlinear signal processing, wireless sensor networks, etc.

Weidong Xiang received his M.S.and Ph.D. degrees from Tsinghua University, Beijing, China, in 1996 and 1999, respectively. From 1999 to 2004, he worked as a Postdoctoral Fellow and then a Research Scientist in the Software Radio Laboratory (SRL) at Georgia Institute of Technology, Atlanta, USA. In 2004, he joined the ECE Department, University of Michigan, Dearborn (UMD) where he currently is an Associate Professor. His research interest includes LTE, vehicular communications and networks, smart grid, software radio/cognitive radio, ultra-wideband (UWB), and wireless networked control systems.

Jing Xu received his M.S. degrees in electronic engineering from Jilin University, Changchun, China, in 2001, and received his Ph.D. degree from the radio engineering department of Southeast University, Nanjing, China, in 2005. He is an associate professor with Shanghai Institute of Microsystem and Information Technology (SIMIT) of Chinese Academy of Sciences (CAS) and has obtained Shanghai Young Rising Star (2010). He is the head of the second research department of shanghai research for wireless communications. His main research interest includes channel equalization, relay technology, inter-cell interference mitigation and cooperative communications.



Hao Huang received the B.S. degree from Beijing Jiaotong University and is currently persuing the Ph.D. degree in Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences. His research interests include adaptive digital predistortion for high efficient power amplifiers, nonlinear system modeling, and design of wireless transceivers.