EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

A low-complexity maximum likelihood decoder for tail-biting trellis

Xiaotao Wang[1,2], Hua Qian[1,3,4*], Kai Kang[1,3,4] and Weidong Xiang[5]

Abstract

Tail-biting trellises are defined on a circular time axis and can have a smaller number of states than equivalent conventional trellises. Existing circular Viterbi algorithms (CVAs) on the tail-biting trellis are non-convergent and sub-optimal. In this study, we show that the net path metric of each tail-biting path is lower-bounded during the decoding process of the CVA. This property can be applied to remove unnecessary iterations of the CVA and results in a convergent maximum likelihood (ML) decoder. Simulation results show that the proposed algorithm exhibits higher decoding efficiency compared with other existing ML decoders.

1 Introduction

For linear block codes, conventional trellis and tail-biting trellis representations have gained a great deal of attention in the past decades [1-3]. Trellis representations not only reveal the code structure, but also lead to efficient trellis-based decoding algorithms [4-8]. For the same linear block codes, the number of states in its tail-biting trellis can be as low as the square root of the number of states in its minimal conventional trellis [1,8], e.g., for the (24, 12) extended Golay code, the maximum number of states in its conventional trellis is 512 [8], while the maximum number of states in its time-varying tail-biting trellis is only 16 [1].

In addition, the tail-biting technique has been widely used in convolutional encoders to eliminate the rate loss caused by the known tail bits. For example, the Worldwide Interoperability for Microwave Access (WiMAX) [9] and Long Term Evolution [10] standards both adopted tail-biting convolutional codes in the control channel or broadcasting channel. Consequently, a maximum likelihood (ML) decoder with high decoding efficiency on tail-biting trellises is important and desirable for studying tail-biting codes.

Both the Viterbi algorithm and bidirectional efficient algorithm for searching code trees (BEAST) can achieve ML decoding on conventional trellises [8]. In the case of a tail-biting trellis, due to the lack of *a priori* knowledge about the starting state, the Viterbi and BEAST decoder have to perform an exhaustive search on tail-biting trellises to find the ML codeword. BEAST can be more efficient if applied to the conventional trellis obtained by reducing the tail-biting code generator matrix to the minimum span form [8].

Another kind of decoder, the two-phase ML decoder, has been proposed to reduce the decoding complexity for tail-biting trellises [5,6]. This kind of algorithm performs Viterbi searches on tail-biting trellises in the first phase and records the accumulated path metric of each path at every section for the second phase. In the second phase, heuristic searches are performing based on the result obtained from the first phase. Since the two-phase decoder is based on two distinct algorithms, this makes it difficult for practical application.

The circular Viterbi algorithm (CVA)-based decoder greatly reduces the implementation complexity of a decoder for tail-biting trellises and provides near-optimal block error rate performance. However, the decoding process of the CVA is non-convergent and sub-optimal [4,7]. In this paper, we introduce a CVA-based ML decoder for tail-biting trellises. In this algorithm, the lower bound of the net path metric of each tail-biting path can be

*Correspondence: hua.qian@shrcwc.org
[1]Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China
[3]Shanghai Research Center for Wireless Communications, Shanghai 200335, China
Full list of author information is available at the end of the article

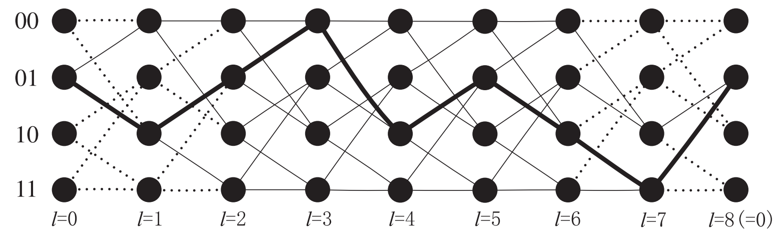

Figure 1 The tail-biting trellis of the (8, 4) convolutional code with octal generator polynomials of {7, 5}.

obtained to exclude impossible starting state candidates, which leads to convergence of the CVA. In addition, the net path metric of survivor paths can be used to terminate redundant searches without performing a full Viterbi iteration.

The following parts of this article are organized as follows: In Section 2, a detailed description of the algorithm is presented. The performance of the proposed algorithm is demonstrated by simulations in Section 3. Section 4 concludes this paper.

2 Algorithm description

2.1 Variable definition

An example of a tail-biting trellis is shown in Figure 1, which has eight sections with four states at each section. For a tail-biting trellis with L sections, denote by $\mathbf{S}_l$ the set of states at location l, where $0 \leq l \leq L$. From the definition of tail-biting trellises, we have $\mathbf{S}_0 \equiv \mathbf{S}_L$. Any path that starts from and terminates at the same state forms a tail-biting path. All tail-biting paths that start from the same state construct a sub-tail-biting trellis of this state. In Figure 1, the branches of thick solid lines form a tail-biting path of state '01,' and all solid lines compose the sub-tail-biting trellis of state '01.'

A conventional CVA-based decoder performs Viterbi iterations around the tail-biting trellis to find the optimal tail-biting path. This algorithm takes advantage of the circular property of tail-biting trellises and employs the path metrics accumulated in the ending states of the trellis to initialize the path metrics in the starting states for a new iteration until a predefined termination condition is fulfilled. In the following parts, we elaborate the decoding process of CVA on tail-biting convolutional codes. The decoding process for general tail-biting trellises can be similarly obtained.

For tail-biting convolutional codes of rate b/c, the length of information bits is bL and the length of the corresponding codeword is cL. Binary code bits $v_l^{(j)} \in \{0, 1\}$ are mapped to $x_l^{(j)} = \left(1 - 2v_l^{(j)}\right)\sqrt{E_s}$ with binary phase-shift keying (BPSK) modulation, where $0 \leq j \leq c-1$ and $0 \leq l \leq L-1$. Without loss of generality, signal energy E_s is normalized to 1. After passing through an additive white Gaussian noise (AWGN) channel with a double-sided noise power spectrum density of $N_0/2$, the corresponding received symbols are $r_l^{(j)}$. The log-likelihood ratio of $r_l^{(j)}$ is given by $\Lambda\left(r_l^{(j)}\right) = 4r_l^{(j)}/N_0$, where for $l \geq L$, $x_l^{(j)} = x_{(l)_L}^{(j)}$, $\Lambda\left(r_l^{(j)}\right) = \Lambda\left(r_{(l)_L}^{(j)}\right)$, and $(l)_L = l \bmod L$.

During the decoding process of the CVA, the accumulated path metric of the survivor path entering state s at location l in the ith iteration is

$$M_l^i(s) = \sum_{k=0}^{(i-1)L+l} \sum_{j=0}^{c-1} \left(x_k^{(j)} \Lambda\left(r_k^{(j)}\right)\right), \ i \geq 1. \tag{1}$$

The weighted Hamming distance between $\Lambda\left(r_l^{(j)}\right)$ and $x_l^{(j)}$ can be defined as in [8]:

$$D\left(\Lambda\left(r_l^{(j)}\right), x_l^{(j)}\right) = \begin{cases} \left|\Lambda\left(r_l^{(j)}\right)\right|, & \text{if } \operatorname{sgn}\left(\Lambda\left(r_l^{(j)}\right)\right) \neq x_l^{(j)} \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $\operatorname{sgn}\left(\Lambda\left(r_l^{(j)}\right)\right)$ denotes the sign of $\Lambda\left(r_l^{(j)}\right)$. Based on (1) and (2), the ML decoding on the tail-biting trellis is equivalent to solving the following equation:

$$\begin{aligned} \widehat{x} &= \arg\max_{\mathbf{x}} \sum_{k=(i-1)L}^{iL-1} \sum_{j=0}^{c-1} \left(x_k^j \cdot \Lambda\left(r_k^{(j)}\right)\right) \\ &= \arg\max_{\mathbf{x}} \sum_{k=(i-1)L}^{iL-1} \sum_{j=0}^{c-1} \left|\Lambda\left(r_k^{(j)}\right)\right| - 2D\left(\Lambda\left(r_k^{(j)}\right), x_k^{(j)}\right) \\ &= \arg\min_{\mathbf{x}} \sum_{k=(i-1)L}^{iL-1} \sum_{j=0}^{c-1} D\left(\Lambda\left(r_k^{(j)}\right), x_k^{(j)}\right). \end{aligned} \tag{3}$$

The term $\left|\Lambda\left(r_k^{(j)}\right)\right|$ can be ignored in the third line of (3) since it is independent of specific codewords $\mathbf{x}$ and consequently is a constant for all paths on the tail-biting trellis.

Denote by $P^i(\beta^i(s), s)$ the survivor path that connects state $\beta^i(s)$ of $\mathbf{S}_0$ with state s of $\mathbf{S}_l$ in the ith iteration.

The corresponding net path metric of $P^i(\beta^i(s), s)$ can be derived from (1) and (3):

$$M^i_{\text{net}}(\beta^i(s), s) = M^i_l(s) - M^i_0(\beta^i(s)). \quad (4)$$

Since the initial path metrics $M^{i+1}_0(s')$ are different from $M^i_0(s')$ for each state $s' \in \mathbf{S_0}$, different survivor paths can be obtained in each iteration. Denote by $\overline{P}^i$ the ML path obtained in the ith iteration, where the ML path obtained from the first iteration has the least net path metric among all possible survivor paths [4]. Similarly, the ML tail-biting path obtained in the ith iteration is denoted by $\widetilde{P}^i$, which has the net path metric of $\widetilde{M}^i$. Among the set of tail-biting paths $\{(\widetilde{P}^i(s,s), \widetilde{M}^i(s,s)) \mid \forall s \in \mathbf{S}_0,\ i \geq 1\}$, the optimal tail-biting path and its net path metric are denoted by $\widetilde{P}^{\text{O}}$ and $\widetilde{M}^{\text{O}}$, respectively.

2.2 Lower bounds of the net path metrics

A conventional CVA-based decoder is non-convergent, and consequently, it cannot guarantee that the tail-biting path obtained is optimal when the decoding process is terminated [4,7]. In order to design a convergent ML decoder on tail-biting trellises, further information needs to be obtained from the decoding process of CVA. Based on the characteristics of CVA, we can derive a lower bound of the net path metric of each tail-biting path. This observation is summarized in Lemma 1.

Lemma 1. *Let $\widetilde{P}(s,s)$ denote the ML tail-biting path on the sub-tail-biting trellis of state s, and the corresponding net path metric is $\widetilde{M}(s,s)$, where $s \in \mathbf{S}_0$. Define $B(s)$ as*

$$B(s) = \max_{i \geq 1}\{M^i_L(s) - M^i_0(s)\}, \quad (5)$$

then we have $B(s) \leq \widetilde{M}(s,s)$, i.e., $B(s)$ is the lower bound of the metrics of all paths on the sub-tail-biting trellis of state s.

Proof. The tail-biting trellis defined on a circular time axis can be split at section $l = 0$ and duplicated on the time axis head-tail. Conventional CVA then becomes a general Viterbi decoder composed of several length L decoding sections, where the Viterbi algorithm searches on the duplicated trellis by recording and repeating the received symbols. Consequently, combining (1) and (3), we find that the survivor path $P^i(\beta^i(s), s)$ has the minimum accumulated path metric $M^i_l(s)$ among all possible paths $P^i(s^*, s)$, where $s^*, \beta^i(s) \in \mathbf{S}_0$, $s \in \mathbf{S}_l$, and $0 \leq l \leq L-1$. Consequently, we have

$$\begin{aligned} M^i_l(s) &= M^i_0(\beta^i(s)) + M^i_{\text{net}}(\beta^i(s), s) \\ &\leq M^i_0(s^*) + M^i_{\text{net}}(s^*, s). \end{aligned} \quad (6)$$

Since (6) holds for $0 \leq l \leq L-1$, we know that for any $s \in \mathbf{S}_L$, (6) also holds. Then for the ML tail-biting path, $\widetilde{P}(s,s)$, on the sub-tail-biting trellis of state s, from (6), we have

$$M^i_L(s) - M^i_0(s) \leq M^i_{\text{net}}(s,s) \leq \widetilde{M}(s,s),\ i \geq 1. \quad (7)$$

Since $B(s) = \max_{i \geq 1}\{M^i_L(s) - M^i_0(s)\}$, then combining (7), we have

$$B(s) \leq \widetilde{M}(s,s). \quad (8)$$

Since $\widetilde{P}(s,s)$ is the ML tail-biting path on the sub-tail-biting trellis of state s, from (3) and (8), we come to the conclusion that $B(s)$ is a lower bound of the metrics of all paths on the sub-tail-biting trellis of state s. □

The lower bound $B(s)$ defined in Lemma 1 is updated as iterations continue, and a more precise estimation of $B(s)$ can be obtained if more iterations are performed on the tail-biting trellis. According to (8), the maximal value of $B(s)$ is $\widetilde{M}(s,s)$.

Based on Lemma 1, we can reduce the decoding complexity of CVA on the tail-biting trellis by removing redundant computations and iterations during the decoding process and control the convergence of CVA. The improvements of the proposed decoder can be summarized into the following two aspects.

Firstly, during the decoding process, if the net path metric $M^i_{\text{net}}(\beta^i(s), s)$ of survivor path $P^i(\beta^i(s), s)$ is not less than $\widetilde{M}^{\text{O}}$, where $\widetilde{M}^{\text{O}}$ is the optimal tail-biting path obtained in the first $i-1$ iterations and $s \in \mathbf{S}_l$, all searches that follow state s can be terminated (refer to Figure 2). In this case, the net path metric of any survivor path that starts from state $\beta^i(s)$ and passes through state s is not less than $\widetilde{M}^{\text{O}}$.

Secondly, denote by $\mathbf{S}^i_{\text{C}}$ the set of survivor starting state candidates in the ith iteration of the CVA, i.e.,

$$\mathbf{S}^i_{\text{C}} = \{s \mid B(s) < \widetilde{M}^{\text{O}}\},$$

where $\mathbf{S}^1_{\text{C}} = \mathbf{S}_0$.

After the ith iteration, $(\widetilde{P}^{\text{O}}, \widetilde{M}^{\text{O}})$ and $B(s)$ will be updated with $(\widetilde{P}^i, \widetilde{M}^i)$ and $B^i(s)$, where $(\widetilde{P}^i, \widetilde{M}^i)$ is the best tail-biting path and its metric obtained in the ith iteration, and $B^i(s)$ is the lower bound of the metrics of all paths on the sub-tail-biting trellis of state s obtained in the ith iteration. For $\forall\ s \in \mathbf{S}^i_{\text{C}}$, if $B(s) \geq \widetilde{M}^{\text{O}}$, then from Lemma 1, we have $\widetilde{M}(s,s) \geq \widetilde{M}^{\text{O}}$. This indicates that the ML tail-biting path on the sub-tail-biting trellis of state s is not better than $\widetilde{P}^{\text{O}}$. State s can be excluded from the set $\mathbf{S}^{i+1}_{\text{C}}$. As iterations continue, the search space on the tail-biting

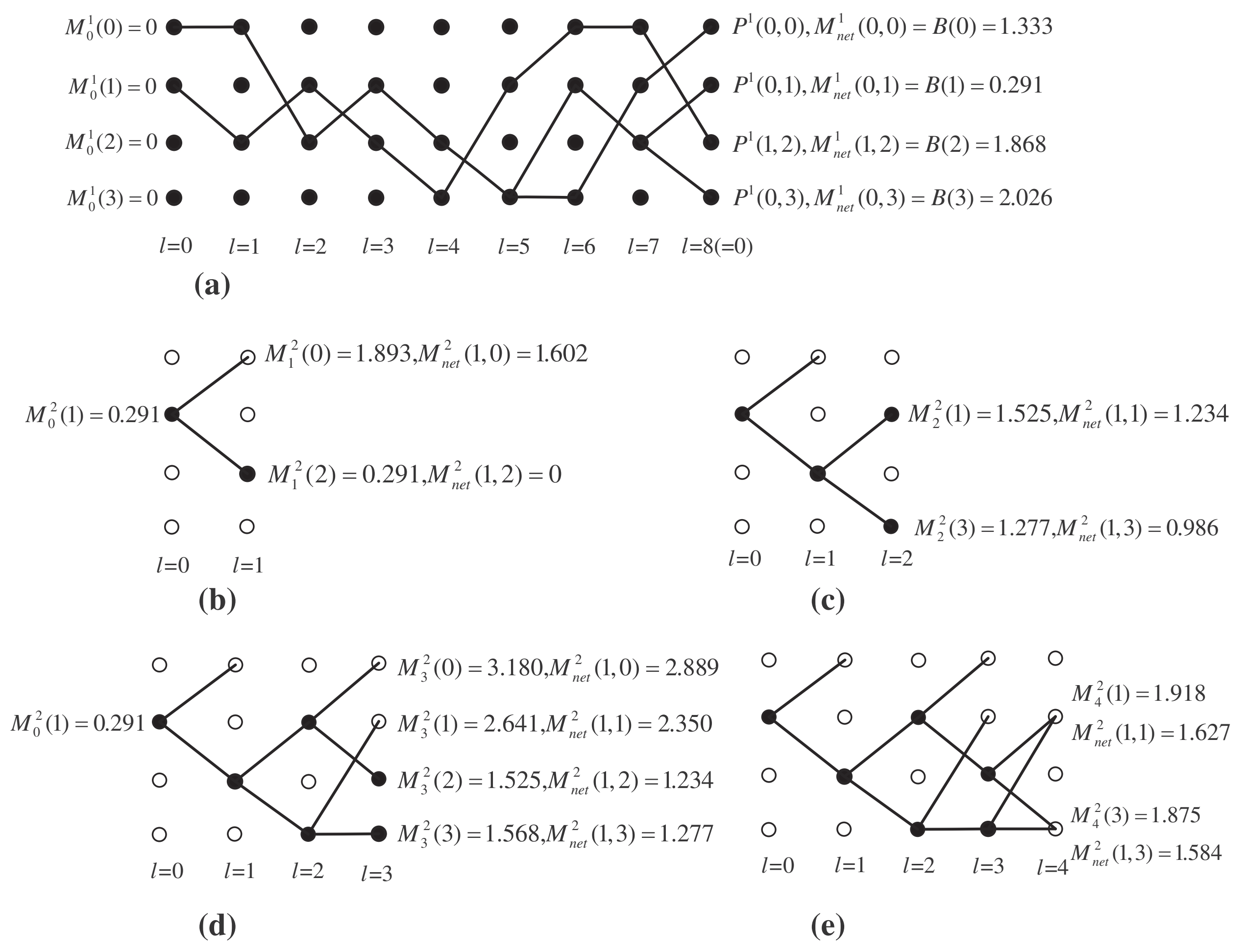
$M_0^1(0)=0$
$M_0^1(1)=0$
$M_0^1(2)=0$
$M_0^1(3)=0$
$P^1(0,0), M_{net}^1(0,0)=B(0)=1.333$
$P^1(0,1), M_{net}^1(0,1)=B(1)=0.291$
$P^1(1,2), M_{net}^1(1,2)=B(2)=1.868$
$P^1(0,3), M_{net}^1(0,3)=B(3)=2.026$
l=0 l=1 l=2 l=3 l=4 l=5 l=6 l=7 l=8(=0)
(a)
$M_0^2(1)=0.291$
$M_1^2(0)=1.893, M_{net}^2(1,0)=1.602$
$M_1^2(2)=0.291, M_{net}^2(1,2)=0$
l=0 l=1
(b)
$M_2^2(1)=1.525, M_{net}^2(1,1)=1.234$
$M_2^2(3)=1.277, M_{net}^2(1,3)=0.986$
l=0 l=1 l=2
(c)
$M_0^2(1)=0.291$
$M_3^2(0)=3.180, M_{net}^2(1,0)=2.889$
$M_3^2(1)=2.641, M_{net}^2(1,1)=2.350$
$M_3^2(2)=1.525, M_{net}^2(1,2)=1.234$
$M_3^2(3)=1.568, M_{net}^2(1,3)=1.277$
l=0 l=1 l=2 l=3
(d)
$M_4^2(1)=1.918$
$M_{net}^2(1,1)=1.627$
$M_4^2(3)=1.875$
$M_{net}^2(1,3)=1.584$
l=0 l=1 l=2 l=3 l=4
(e)

Figure 2 Decoding process of the B-CVA ML decoder. For the first iteration, four survivor paths corresponding to each of the terminal states are shown on the tail-biting trellis in (**a**), where the tail-biting trellis has been split at location $l = 0$. For the second iteration, the decoding process is illustrated from (**b**) to (**e**). Searches following the circles with left incident branch are terminated according to step 2.1. The decoding process is terminated at $l = 4$ since there are no survivor paths with net path metric less than $\widetilde{M}^{\mathrm{O}}$.

trellis shrinks and the decoder will converge to the global optimal solution eventually.

2.3 CVA-based ML decoder on tail-biting trellis

We summarize the above decoding process as follows: In the algorithm description, the operator '$\leftarrow$' denotes value assignment from the right-hand side to the left-hand side, and the operator '=' denotes logic comparison between two operands.

From the description above, we find that the decoding process can only be terminated when $\mathbf{S}_{\mathrm{C}}^{i+1} = \emptyset$ in step 2.4. The number of entries in $\mathbf{S}_{\mathrm{C}}^{1}$ is finite, and as iterations continue, the size of $\mathbf{S}_{\mathrm{C}}^{i}$ will reduce to zero.

Firstly, state s with bound $B(s) > \widetilde{M}^{\mathrm{O}}$ is deleted from $\mathbf{S}_{\mathrm{C}}^{i}$ in step 2.3 since the ML tail-biting path on the sub-tail biting trellis of state s is not better than $\widetilde{P}^{\mathrm{O}}$.

Secondly, after the $(i+1)$th iteration, if no state has been deleted from $\mathbf{S}_{\mathrm{C}}^{i+1}$, i.e., $\forall\ s \in \mathbf{S}_{\mathrm{C}}^{i+1}$, $B(s) < \widetilde{M}^{\mathrm{O}}$, then equation $\mathbf{S}_{\mathrm{C}}^{i+1} = \mathbf{S}_{\mathrm{C}}^{i+2}$ holds and the Viterbi algorithm will be performed on the sub-tail-biting trellis of state $s^{\dagger}$ in step 2.4, where $s^{\dagger} \in \mathbf{S}_{\mathrm{C}}^{i+1}$. If $\widetilde{M}(s^{\dagger}, s^{\dagger}) < \widetilde{M}^{\mathrm{O}}$, $\widetilde{P}^{\mathrm{O}}$ will be updated with $\widetilde{P}(s^{\dagger}, s^{\dagger})$; else, if $\widetilde{M}(s^{\dagger}, s^{\dagger}) \geq \widetilde{M}^{\mathrm{O}}$, this indicates that $\widetilde{P}(s^{\dagger}, s^{\dagger})$ is not better than $\widetilde{P}^{\mathrm{O}}$. Then state $s^{\dagger}$ can be deleted from $\mathbf{S}_{\mathrm{C}}^{i+1}$ since the ML tail-biting path on its sub-tail-biting trellis has been found.

Thirdly, after the ith iteration, if $\widetilde{M}(s', s') < \widetilde{M}^{\mathrm{O}}$, $\widetilde{P}^{\mathrm{O}}$ will be updated with $\widetilde{P}(s', s')$, where $s' \in \mathbf{S}_{\mathrm{C}}^{i}$. Since $B(s') = \max\limits_{i \geq 1}\{M_L^i(s') - M_0^i(s')\} \leq \widetilde{M}(s', s')$, $B(s')$ will be updated with $\widetilde{M}(s', s')$ in step 2.3, and then state s' will be deleted from $\mathbf{S}_{\mathrm{C}}^{i}$ since the equality in $B(s) \geq \widetilde{M}^{\mathrm{O}}$ holds.

All survivor starting state candidates in $\mathbf{S}_{\mathrm{C}}^{1}$ will be deleted eventually, and $\mathbf{S}_{\mathrm{C}}^{i}$ will be empty in a finite number of iterations. When $\mathbf{S}_{\mathrm{C}}^{i}$ is empty, the decoder converges to the global optimal solution that has been recorded in

Algorithm description

```
1. Initialization: $\mathbf{S}_C^1 \leftarrow \mathbf{S_0}$, $M_0^1(s) \leftarrow 0$ for all $s \in \mathbf{S}_C^1$, $\widetilde{M}^O \leftarrow +\infty$;
2. For iteration $i$, $i \geq 1$:
    2.1 Perform Viterbi algorithm on the tail-biting trellis with the set of starting state candidates $\mathbf{S}_C^i$; during the decoding process, $\forall\ P^i(\beta^i(s'), s')$, where $s' \in \mathbf{S}_l$, if $M_{net}^i(\beta^i(s'), s') \geq \widetilde{M}^O$, terminate the search that follows state $s'$ as mentioned above;
    2.2 Find ML tail-biting path $\widetilde{P}^i$ if it exists, update $(\widetilde{P}^O, \widetilde{M}^O)$ with $(\widetilde{P}^i, \widetilde{M}^i)$ if $\widetilde{M}^i < \widetilde{M}^O$;
    2.3 For $\forall s \in \mathbf{S}_C^i$, calculate $B^i(s) = M_L^i(s) - M_0^i(s)$ and update $B(s)$ with $B^i(s)$ if $B^i(s) > B(s)$; generate $\mathbf{S}_C^{i+1}$ by excluding state $s$ with $B(s) \geq \widetilde{M}^O$ from set $\mathbf{S}_C^i$;
    2.4 If $\mathbf{S}_C^{i+1} = \emptyset$, go to step 3; else if $\mathbf{S}_C^{i+1} = \mathbf{S}_C^i$, find the starting state $s^\dagger$ of ML path $\overline{P}^i$ and perform Viterbi algorithm on the sub-tail-biting trellis of state $s^\dagger$, step 2.1 is also applied in this case; delete $s^\dagger$ from $\mathbf{S}_C^{i+1}$; update $\widetilde{P}^O$ with $\widetilde{P}(s^\dagger, s^\dagger)$ if $\widetilde{M}(s^\dagger, s^\dagger) < \widetilde{M}^O$;
    2.5 $\forall\ s \in \mathbf{S}_C^{i+1}$: $M_0^{i+1}(s) \leftarrow M_L^i(s)$; $i \leftarrow i + 1$; go back to step 2;
3. Output the codeword associated with $\widetilde{P}^O$;
```

$\widetilde{P}^O$. In summary, the algorithm presented above is a convergent ML decoder on tail-biting trellises. We call this a bounded CVA (B-CVA) ML decoder.

The following example is used to explain the decoding process of the B-CVA.

Example 1. The tail-biting convolutional code that is represented by the tail-biting trellis in Figure 1 is selected in this example. The information sequence is {01011100}, and the corresponding codeword is {(00), (11), (10), (00), (01), (10), (01), (11)}. After passing through the AWGN channel where the signal-to-noise ratio (SNR) is $E_b/N_0 =$ 0 dB, the received sequence (for one realization) is {(1.144, 0.458), (−0.986, −1.234), (0.291, 1.364), (0.472, 0.350), (1.578, −1.594), (0.050, −0.399), (2.260, 0.359), (−1.501, 0.234)}. For convenience, states {00, 01, 10, 11} are represented by {0, 1, 2, 3}. For the first iteration, set $\mathbf{S}_C^1 = \{0, 1, 2, 3\}$ and the four survivor paths are shown on Figure 2a. We find that there is a tail-biting path $P^1(0, 0)$ with net path metric $M_{net}^1(0, 0) = 1.333$. Then $(\widetilde{P}^O, \widetilde{M}^O)$ are updated with $(P^1(0, 0), M_{net}^1(0, 0))$. The bounds of each terminal state are updated to $B(0) = 1.333$, $B(1) = 0.291$, $B(2) = 1.868$, and $B(3) = 2.026$. Since only $B(1) < \widetilde{M}^O$, according to step 2.3 of the B-CVA, $\mathbf{S}_C^2 = \{1\}$. The decoding process of the second iteration is illustrated in Figure 2b, c, d, e. We find that with the control on the decoding process of the CVA, the second iteration terminates at section $l = 4$; and at each section, searches after state s' that has $M_{net}^2(1, s') \geq \widetilde{M}^O$ are terminated. The state s' corresponds to the circle with left incident branch on the trellis in Figure 2b, c, d, e. The decoding result is the codeword associated with the tail-biting path $\widetilde{P}^O$. In the first iteration, there are 64 real additions and 32 logical comparisons; and in the second iteration, the numbers of real additions and logical comparisons corresponding to each step are {(4, 2), (4, 2), (8, 4), (6, 4)}. In summary, the decoding complexity of B-CVA is denoted by 86 real additions and 44 logical comparisons. As is shown in Appendix 1,, the decoding complexity of the two-phase decoder is denoted by 100 real additions and 101 logical comparisons.

3 Simulation results

In order to show the decoding efficiency, we compare the proposed B-CVA ML decoder with other widely cited tail-biting ML decoders in three aspects: the number of real additions, the number of logical comparisons, and average memory space consumption during the decoding process. The codewords are modulated to BPSK symbols and then passed through an AWGN channel. The results shown in the tables and figures are obtained by observing at least 100 block error events.

The conventional BEAST ML decoder should perform decoding on each sub-tail-biting trellis independently and consecutively to find the ML tail-biting path [8]. To improve its efficiency, we use the upper-bounding technique on the thresholds used by BEAST. During the consecutive decoding process, the metric of the optimal tail-biting path obtained on the first i sub-tail-biting trellises are used to upper-bound the thresholds that will be used for decoding the remaining $|\mathbf{S}_0| - i$ sub-tail-biting trellises, where $|\mathbf{S}_0|$ denotes the size of set $\mathbf{S}_0$. With this

Table 1 BLER performance of WAVA and ML decoders for (64, 32) tail-biting convolutional codes over the AWGN channel

	SNR				
BLER performance	**0.0 dB**	**1.0 dB**	**2.0 dB**	**3.0 dB**	**4.0 dB**
WAVA	3.95×10^{-1}	1.49×10^{-1}	2.80×10^{-2}	2.70×10^{-3}	1.06×10^{-4}
ML decoders	3.80×10^{-1}	1.48×10^{-2}	2.66×10^{-2}	2.50×10^{-3}	0.96×10^{-4}

The ML decoders are the advanced BEAST ML decoder, two-phase ML decoder, or the B-CVA ML decoder.

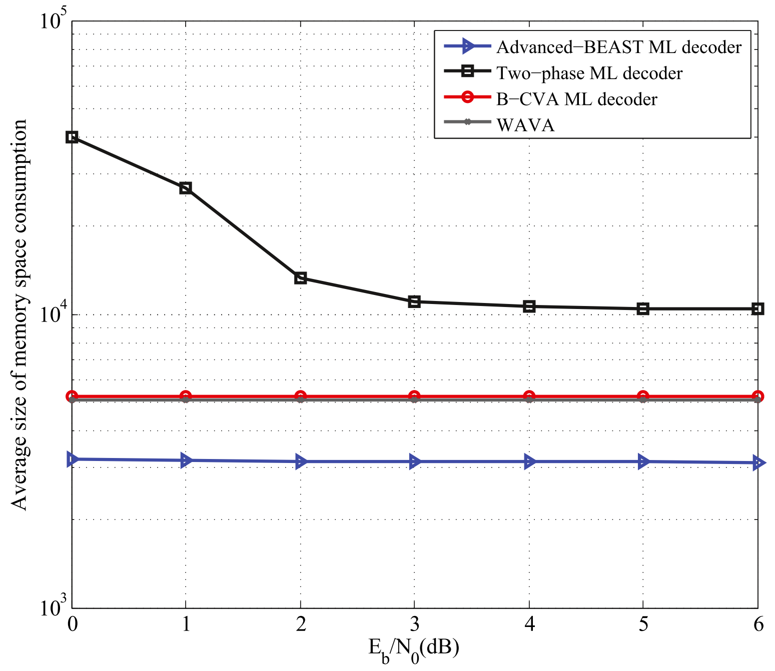

Figure 3 Average memory space required for decoding the (64, 32) tail-biting convolutional codes. Average memory space consumption by the WAVA and different ML decoders: advanced BEAST ML decoder, two-phase ML decoder and B-CVA ML decoder, for decoding the (64, 32) tail-biting convolutional codes over the AWGN channel.

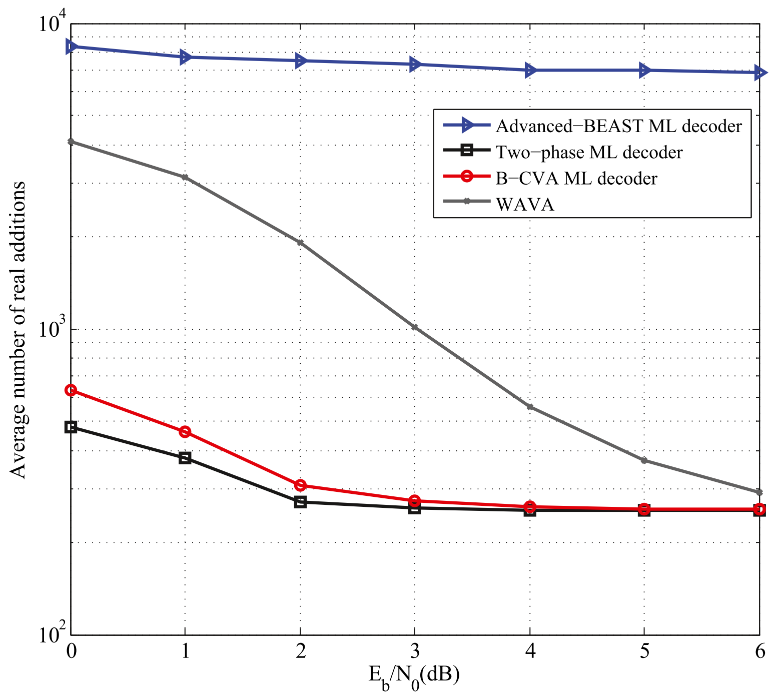

Figure 4 Average number of real additions required for decoding the (64, 32) tail-biting convolutional codes. Average number of real additions that were performed during the decoding process of the WAVA and different ML decoders: advanced BEAST ML decoder, two-phase ML decoder and B-CVA ML decoder, for decoding the (64, 32) tail-biting convolutional codes over the AWGN channel.

Table 2 BLER performance of WAVA and ML decoders for (80, 40) tail-biting convolutional code over the AWGN channel

	SNR				
BLER performance	**1.0 dB**	**2.0 dB**	**3.0 dB**	**4.0 dB**	**5.0 dB**
WAVA	1.60×10^{-1}	3.16×10^{-2}	3.10×10^{-3}	1.45×10^{-4}	3.50×10^{-6}
ML decoders	1.59×10^{-1}	3.05×10^{-2}	3.10×10^{-3}	1.44×10^{-4}	3.50×10^{-6}

The ML decoders are the advanced BEAST ML decoder, two-phase ML decoder, or the B-CVA ML decoder.

upper-bounding technique, redundant searches can be terminated timely or reduced through the whole decoding process. For convenience, we call the BEAST decoder with this upper-bounding technique an advanced-BEAST ML decoder, which should be more efficient than the original BEAST decoder in [8]. Results presented in Appendix 4 are used to demonstrate this point.

In the first example, different decoders were applied for decoding the tail-biting convolutional codes (64, 32) with octal generator polynomials {345, 237}, which can be represented by a 128-state tail-biting trellis [7]. For the demonstration of the block error rate (BLER) performance of the B-CVA, we choose the most cited sub-optimal decoder proposed in [7] for comparison. This decoder is called the wrap-around Viterbi algorithm (WAVA). Because the WAVA is non-convergent, the maximal allowed number of iterations is set as 20 during the decoding process. Table 1 shows the BLER performance of different decoders. We find that of the sub-optimal decoders, WAVA has performance that is close to the optimal results.

Figure 3 shows the average memory space consumption of the different ML decoders during the decoding process. We find that the advanced BEAST decoder and the B-CVA decoder require almost constant memory space from the low- to high-SNR regions. However, we find that the WAVA decoder requires less memory space than the B-CVA. This is due to the fact that WAVA has no need of space for storing $B(s)$ of every state in $\mathbf{S}_0$. The memory space required by the two-phase decoder is about $2 \sim 12$ times more than that required by the B-CVA or BEAST decoders. This is due to the fact that the two-phase decoder has to store path metrics of all survivor paths obtained in the first phase and has to maintain the open stack and close table during the second phase.

Figure 4 shows that the two-phase ML decoder which

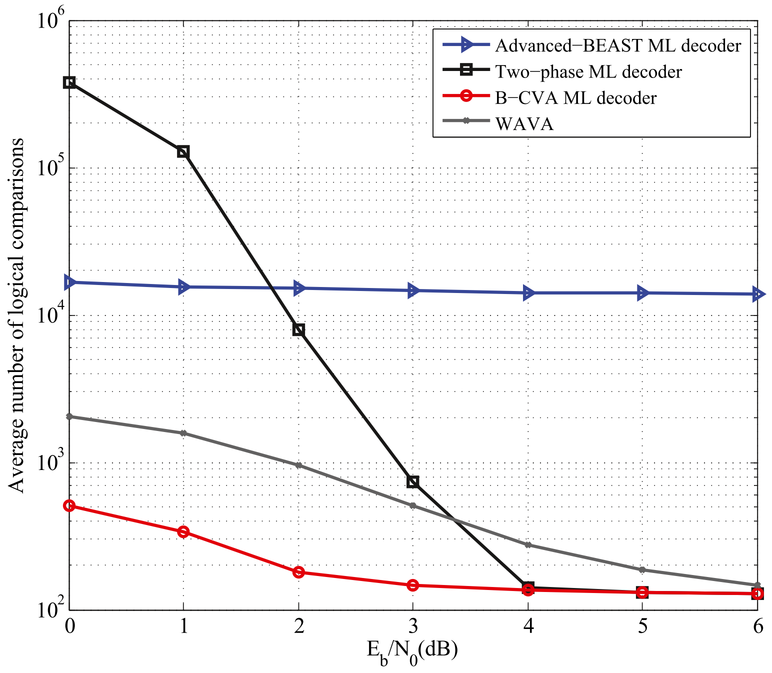

Figure 5 Average number of logical comparisons required for decoding the (64, 32) tail-biting convolutional codes. Average number of logical comparisons that were performed during the decoding process of the WAVA and different ML decoders: advanced BEAST ML decoder, two-phase ML decoder, and B-CVA ML decoder, for decoding the (64, 32) tail-biting convolutional codes over the AWGN channel.

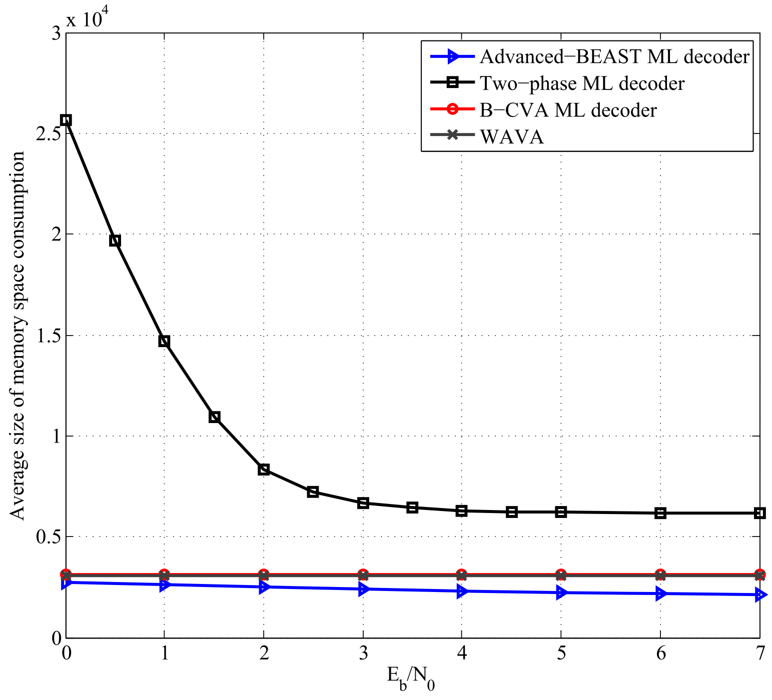

Figure 6 Average memory space required for decoding the (80, 40) tail-biting convolutional code. Average memory space consumption by the WAVA and different ML decoders: advanced BEAST ML decoder, two-phase ML decoder, and B-CVA ML decoder, for decoding the (80, 40) tail-biting convolutional codes over the AWGN channel.

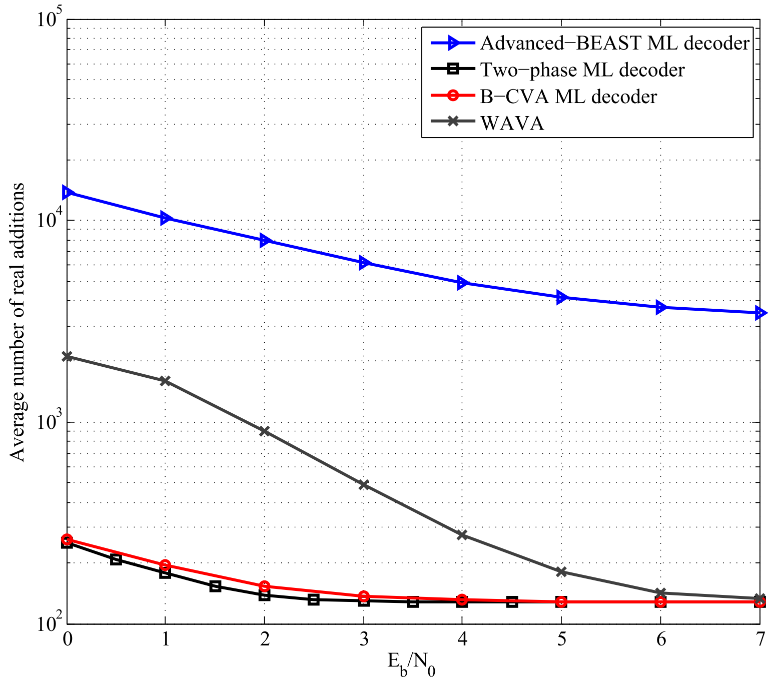

Figure 7 Average number of real additions required for decoding the (80, 40) tail-biting convolutional code. Average number of real additions that were performed during the decoding process of the WAVA and different ML decoders: advanced BEAST ML decoder, two-phase ML decoder and B-CVA ML decoder, for decoding the (80, 40) tail-biting convolutional codes over the AWGN channel.

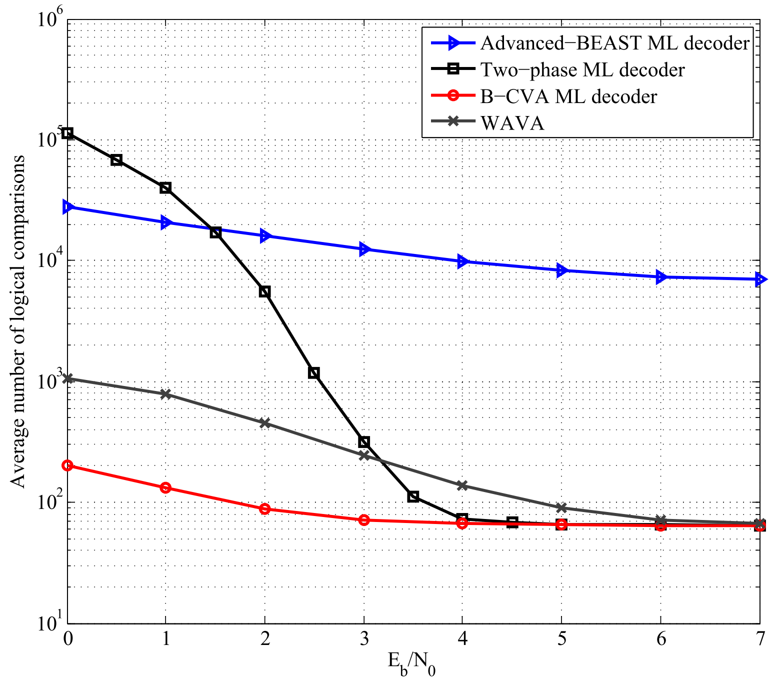

Figure 8 Average number of logical comparisons required for decoding the (80, 40) tail-biting convolutional code. Average number of logical comparisons that were performed during the decoding process of the WAVA and different ML decoders: advanced BEAST ML decoder, two-phase ML decoder and B-CVA ML decoder, for decoding the (80, 40) tail-biting convolutional codes over the AWGN channel.

is based on depth-first searches is a little better than the B-CVA ML decoder in the number of real additions. The BEAST ML decoder which performs exhaustive searches on the tail-biting trellis shows the highest decoding complexity. Figure 5 shows that a larger number of logical comparisons should be performed by the two-phase decoder than that performed by the B-CVA decoder. This is due to the fact that many logical comparisons need to be performed to sort the open stack and to retrieve the close table in the low-SNR region. In fact, the length of the close table grows linearly as the searches continue. Figures 4 and 5 also show that the advanced BEAST decoder, which has to be performed on each sub-tail-biting trellis, exhibits high decoding complexity from the low- to high-SNR regions. Meanwhile, we find that the WAVA exhibits high decoding complexity from the low- to medium-SNR regions; this is caused by circular traps existing in the CVA decoding process [4].

The second example adopts the tail-biting convolutional codes that have been used in WiMAX. The corresponding generator polynomials are {133, 171} in octal form [9]. The length of the information sequence is 40 bits, and the corresponding code is a long tail-biting convolutional code [7], which is denoted as an (80, 40) tail-biting convolutional code. The BLER of the WAVA and ML decoders are presented in Table 2, where the ML decoder in Table 1 refers to any kind of ML decoders mentioned in this paper: the advanced BEAST ML decoder, two-phase ML decoder, or BCVA decoder, since all of them exhibit exactly the same block error rate performance. We find that in the case of long tail-biting codes, the BLER performance of WAVA is close to that of ML decoders.

Table 3 Updates of the open stack and close table during heuristic searches of the two-phase ML decoder

Step	Open stack	Close table	Complexity
1	{(1, 1, 0, 0.291)}	∅	{(6, 4)}
2	{(1, 2, 1, 0.291)}	{(1, 1, 0)}	{(6, 7)}
3	{(1, 3, 2, 0.986), (1, 1, 2, 1.234)}	{(1, 1, 0), (1, 2, 1)}	{(6, 10)}
4	{(1, 1, 2, 1.234), (1, 3, 3, 1.277)}	{(1, 1, 0), (1, 2, 1), (1, 3, 2)}	{(6, 13)}
5	{(1, 2, 3, 1.234), (1, 3, 3, 1.277)}	{(1, 1, 0), (1, 2, 1), (1, 3, 2), (1, 1, 2)}	{(6, 16)}
6	{(1, 3, 3, 1.277)}	{(1, 1, 0), (1, 2, 1), (1, 3, 2), (1, 1, 2), (1, 2, 3)}	{(6, 19)}

Table 4 Average numbers of real additions and logical comparisons in decoding the (80, 40) code used in WiMAX

Decoding complexity		SNR							
		0.0 dB	**1.0 dB**	**2.0 dB**	**3.0 dB**	**4.0 dB**	**5.0 dB**	**6.0 dB**	**7.0 dB**
BEAST	Additions	10,307	10,083	9,989	9,583	9,288	9,222	9,013	8,999
	Comparisons	20,613	20,167	19,978	19,165	18,576	18,443	18,026	17,998
Advanced BEAST	Additions	5,248	4,395	3,793	3,411	3,218	3,048	2,996	2,930
	Comparisons	10,497	8,790	7,585	6,821	6,436	6,096	5,993	5,860

From Figures 6, 7 and 8, we come to almost the same conclusions as that in the first example. In Figure 6, the memory space required by the advanced BEAST decoder decreases slowly as system SNR grows. Figure 8 shows that, in the low-SNR region, many comparisons were performed to maintain the open stack and close table in the second phase of the two-phase ML decoder. In summary, the B-CVA decoder is an efficient ML decoder on tail-biting trellises both in memory space saving and decoding complexity reduction.

4 Conclusion

We propose a convergent CVA-based ML decoder for tail-biting trellises. The proposed algorithm takes advantage of the lower bound of the net path metric of the tail-biting path to control the decoding process of CVA. Simulation results show that, on tail-biting trellises, the B-CVA decoder exhibits high decoding efficiency while requiring relatively small memory space during the decoding process. These advantages make it attractive to practical applications.

Appendices

Appendix 1: decoding process of the two-phase ML decoder

In Example 1, the decoding process of the two-phase ML decoder can be described as follows [6]: (1) During the first phase, the decoder stores the path metric $M_l^1(s)$ of $\forall s \in \mathbf{S}_l$ and $0 \leq l \leq L$; the decoding complexity of the first phase is the same as that of the B-CVA, which contains 64 real additions and 32 logical comparisons; (2) in the second phase, heuristic searches are performed on the sub-tail-biting trellis of state 01; in each step, the heuristic search is performed following the top path in the open stack. There are 3 real additions and 4 logical comparisons for updating the values of the g-function, h-function, and f-function. The two successors will be saved in an open stack if the values of their f-functions are less than $\widetilde{M}^1$. Meanwhile, the starting state, terminal state, and current section l of the top path will be saved in the close table. Then, the top path of the open stack will be compared with each entry of the close table in three aspects: starting state, terminal state, and current location. With six heuristic searching steps, the decoding process is stopped. The decoding complexity in summary is denoted by 100 real additions and 105 logical comparisons. We have ignored the complexity of sorting the open stack according to ascending order of their f-function values. The open stack and the corresponding close table obtained in each step are shown in Table 3, where Open stack $=$ {(starting state, current state, l-section, f-function)}, Close table $=$ {(starting state, current state, l-section)}, and Complexity $=$ {(the number of real additions, the number of logical comparisons)}.

Appendix 2: improvements of the BEAST ML decoder with upper-bounding technique

To show the decoding efficiency improvement of the advanced BEAST decoder, we use it and the original BEAST decoder [8] to decode a tail-biting convolutional code used in WiMAX. The length of the information sequence is 40 bits. The average numbers of real additions and logical comparisons are presented in Table 4. We find that the upper-bounding technique can cut the decoding complexity of the BEAST decoder by $1/2 \sim 2/3$.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This work was supported in part by the Hundred Talents Program of the Chinese Academy of Sciences, the Shanghai Pujiang Talent Program No. 11PJ1408700, the National Science and Technology Major Project No. 2011ZX03003-003-04, the International Science and Technology Cooperation Program of Shanghai Municipality No. 11220705400, and the Science and Technology Commission Foundation of Shanghai No. 12511503400.

Author details
[1]Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China. [2]Graduate University of Chinese Academy of Sciences, Beijing 100049, China. [3]Shanghai Research Center for Wireless Communications, Shanghai 200335, China. [4]Shanghai Internet of Things Co., Ltd., Shanghai 201899, China. [5]Electrical and Computer Engineering Department, University of Michigan-Dearborn, Dearborn, MI 48128-2406, USA.

Received: 16 August 2012 Accepted: 3 May 2013
Published: 17 May 2013

References

1. A Calderbank, G Forney, Vardy A, Minimal tailbiting trellises: Golay code and more. IEEE Trans. Inf. Theory. **45**(5), 1435-1455 (1999)
2. P Stahl, J Anderson, R Johannesson, Optimal and near-optimal encoders for short and moderate-length tailbiting trellises. IEEE Trans. Inf. Theory. **45**(7), 2562–2571 (1999)
3. H Gluesing-Luerssen, E Weaver, Linear tail-biting trellises: characteristic generators and the BCJR-construction. IEEE Trans. Inf. Theory. **57**(2), 738–751 (2011)
4. X Wang, H Qian, J Xu, Y Yang, F Wang, in *IEEE Global Telecommunications Conference*. An efficient CVA-based decoding algorithm for tail-biting codes ((Houston, 5–9 Dec 2011), pp. 1–5
5. P Shankar, P Kumar, K Sasidharan, B Rajan, A Madhu, Efficient convergent maximum likelihood decoding on tail-biting (2007). http://arxiv.org/pdf/cs/0601023v1.pdf. Accessed 10 Aug 2007
6. H Pai, Y Han, T Wu, P Chen, S Shieh, Low-complexity ML decoding for convolutional tail-biting codes. IEEE Commun. Lett. **12**(12), 883–885 (2008)
7. R Shao, S Lin, M Fossorier, Two decoding algorithms for tailbiting codes. IEEE Trans. Commun. **51**(10), 1658–1665 (2003)
8. IE Bocharova, R Johannesson, BD Kudryashov, M Loncar, BEAST decoding for block codes. Europ. Trans. Telecomm. **15**, 297–305 (2004)
9. IEEE, *IEEE Std 802.16–2009, IEEE Standard for, Local and Metropolitan Area Networks Part 16: Air Interface for Broadband Wireless Access Systems*. (IEEE, Piscataway, 2009)
10. 3rd Generation Partnership Project, *3GPP TS 36.212, Technical Specification, Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding (Release 8)*. (ETSI, Sophia Antipolis, 2007)

doi:10.1186/1687-1499-2013-130
Cite this article as: Wang *et al.*: A low-complexity maximum likelihood decoder for tail-biting trellis. *EURASIP Journal on Wireless Communications and Networking* 2013 **2013**:130.