Searching for high-rate convolutional codes via binary syndrome trellises

Hug, Florian; Bocharova, Irina; Johannesson, Rolf; Kudryashov, Boris

Published in: [Host publication title missing]

DOI:
10.1109/ISIT.2009.5205916

2009

Citation for published version (APA):
Searching for High-Rate Convolutional Codes via Binary Syndrome Trellises

Florian Hug\textsuperscript{1}, Irina E. Bocharova\textsuperscript{2}, Rolf Johannesson\textsuperscript{1}, and Boris D. Kudryashov\textsuperscript{2}

\textsuperscript{1} Dept. of Electrical and Information Technology, Lund University
P. O. Box 118, SE-22100 Lund, Sweden
Email: \{florian, rolf\}@eit.lth.se

\textsuperscript{2} Dept. of Information Systems
St. Petersburg Univ. of Information Technologies, Mechanics and Optics
St. Petersburg 197101, Russia
Email: \{irina, boris\}@eit.lth.se

Abstract—Rate $R = (c - 1)/c$ convolutional codes of constraint length $\nu$ can be represented by conventional syndrome trellises with a state complexity of $s = \nu$ or by binary syndrome trellises with a state complexity of $s = \nu$ or $s = \nu + 1$, which corresponds to at most $2^s$ states at each trellis level. It is shown that if the parity-check polynomials fulfill certain conditions, there exist binary syndrome trellises with optimum state complexity $s = \nu$.

The BEAST is modified to handle parity-check matrices and used to generate code tables for optimum free distance rate convolutional codes with overall constraint length $\nu$ using the syndrome trellis as proposed in [10]. As an alternative to puncturing, we will consider the binary syndrome trellis representation. In general, this comes at the cost of increasing the state complexity from $s = \nu$ to $s = \nu + 1$, that is, the number of states at each level of the trellis doubles [11].

In Section II, we show that a binary syndrome trellis can be realized with state complexity $s = \nu$, that is, at most $2^s$ different states, if the parity-check polynomials fulfill certain conditions. The BEAST is modified in Section III to handle parity-check matrices. In Section IV, we present tables for optimum free distance rate $R = (c - 1)/c$, $c = 3, 4, 5$, convolutional codes for conventional syndrome trellises as well as for optimum state complexity binary syndrome trellises.

I. INTRODUCTION

High-rate convolutional codes are important for many applications due to the combination of a modest rate loss and the existence of efficient maximum-likelihood (ML) decoding algorithms.

If we ignore the so-called start-up phase, every level of the conventional trellis of such a rate $R = b/c$ convolutional code with overall constraint length $\nu$ consists of $2^s$ states, with $2^s$ branches arriving at and leaving each state. In [1] the state complexity $s$ is called the most widely accepted measure of trellis complexity. The state complexity is given as the maximum value of the logarithm of the number of states at any level of the considered trellis. For a rate $R = b/c$ convolutional code with overall constraint length $\nu$, the state complexity has to be at least greater than or equal to $\nu$, that is,

$$s = \max \{ \log_2 |V_i| \} \geq \nu$$

(1)

where $|V_i|$ denotes the number of states at level $i$ in the trellis.

The ML decoding complexity, e.g., of the Viterbi algorithm [2], depends also on the number of branches arriving at and leaving each state, that is, $2^s$ in the conventional trellis of a rate $R = b/c$ convolutional code. Traditionally this number has been reduced significantly by introducing puncturing which leads to binary trellises with only two branches arriving at and leaving each node [3].

Since Paaske [4] reported on early searches for high-rate convolutional codes using the parity-check matrices, a series of papers reporting on various search techniques for good codes has been published [5]–[9]. In this paper we will report the results of a search for high-rate $R = (c - 1)/c$, $c = 3, 4, 5$ convolutional codes with overall constraint length $\nu$ using the syndrome trellis as proposed in [10]. As an alternative to puncturing, we will consider the binary syndrome trellis representation. In general, this comes at the cost of increasing the state complexity from $s = \nu$ to $s = \nu + 1$, that is, the number of states at each level of the trellis doubles [11].

In Section II, we show that a binary syndrome trellis can be realized with state complexity $s = \nu$, that is, at most $2^s$ different states, if the parity-check polynomials fulfill certain conditions. The BEAST is modified in Section III to handle parity-check matrices. In Section IV, we present tables for optimum free distance rate $R = (c - 1)/c$, $c = 3, 4, 5$, convolutional codes for conventional syndrome trellises as well as for optimum state complexity binary syndrome trellises.

II. COMPLEXITY OF THE SYNDROME TRELLIS

Consider a rate $R = (c - 1)/c$, $c \geq 2$, convolutional code with parity-check matrix

$$H(D) = (h_1(D) \ h_2(D) \ \ldots \ h_c(D))$$

(2)

where $h_i(D)$, $i = 1, 2, \ldots, c$, denotes the $i$th parity-check polynomial. Furthermore, let $\deg(h_i(D))$ and $\deg(h_i(D))$ denote the delay and the degree of $h_i(D)$, respectively. The overall constraint length $\nu$ of this parity-check matrix is given by

$$\nu = \max \{ \deg(h_i(D)) \}.$$ 

For the parity-check polynomials $h_i(D)$, $i = 1, \ldots, c$, in (2), we have

$$h_i(D) = h_i^{(0)} + h_i^{(1)} D + \ldots + h_i^{(\nu)} D\nu.$$ 

Then the parity-check matrix $H(D)$ can be represented by the semi-infinite matrix $H$ consisting of $1 \times c$ sub-matrices $H_l$, $l = 0, 1, \ldots, c$, where

$$H_l = \begin{pmatrix} h_i^{(l)} \end{pmatrix}_{1 \times c}, \quad l = 0, 1, \ldots, \nu.$$ 

For example, for $\nu = 2$, we have the matrix $H$ given in (3). Following [10], a syndrome trellis can be constructed by connecting the identically full syndrome trellis modules corresponding to the parity-check matrix module $\tilde{H}$ [11] given by (4).
In case of a rate $R = (c - 1)/c$ convolutional code, every column in the matrix module $H$ corresponds to one of the $c$ parity-check polynomials $h_i(D)$, $i = 1, 2, \ldots, c$, written in binary notation starting from the top.

$$H = \begin{pmatrix}
H_0 & H_0 & \cdots \\
H_1 & H_0 & \cdots \\
H_2 & H_1 & \cdots \\
& & \ddots & \cdots \\
& & & H_\nu
\end{pmatrix}.$$  \hspace{1cm} (3)

$$\tilde{H} = \begin{pmatrix}
H_0 \\
\vdots \\
H_\nu
\end{pmatrix}.$$ \hspace{1cm} (4)

Every level (after the startup-phase) of the syndrome trellis of a rate $R = (c - 1)/c$ convolutional code, when sectionalized to $c$ bits per branch, consists of $2^c$ different states with $2^{c-1}$ branches arriving at and leaving each state. For the rate $R = 2/3$ convolutional code with parity-check matrix $H(D) = (1 + D^2 \ 1 + D + D^2)$ the corresponding conventional syndrome trellis module is illustrated in Fig. 1.

A binary syndrome trellis is specified using only two branches arriving at and leaving each state. This simplification, however, comes at the cost of $c - 1$ additional intermediate layers in each trellis module, where the $c - 1$ additional layers may consist of as many as $2^{c-1}$ different states. For the same convolutional code as before, the binary syndrome trellis module is depicted in Fig. 2.

Having a closer look at the binary syndrome trellis module, we notice that for this convolutional code it is possible to find an equivalent convolutional code [12] whose maximum number of states at each intermediate layer does not exceed $2^c$. Reordering the parity-check polynomials, we obtain the equivalent parity-check matrix $H_{eq}(D) = (1 + D \ 1 + D + D^2)$. Sectionalizing its binary syndrome trellis module, the number of intermediate layers can be reduced by one while the number of states at each intermediate layer decreases to $2^\nu$, which is illustrated for the given code in Fig. 3.

In the following we will give conditions which determine whether it is possible to decrease the number of states at the intermediate layers of the binary syndrome trellis. We introduce the abbreviation $\text{dgl}(p(D))$ for the difference between the degree and the delay of a polynomial $p(D)$, that is,

$$\text{dgl}(p(D)) = \deg(p(D)) - \text{del}(p(D)).$$ \hspace{1cm} (5)

Every parity-check polynomial $h_i(D)$, $i = 1, \ldots, c$, belongs to at least one of the sets, $\mathcal{H}_I$, $\mathcal{H}_{II}$, and $\mathcal{H}_{III}$, where

$$\mathcal{H}_I = \{h_i(D), i = 1, 2, \ldots, c \mid \deg(h_i(D)) < \nu\}$$ \hspace{1cm} (6)

$$\mathcal{H}_{II} = \{h_i(D), i = 1, 2, \ldots, c \mid \text{dgl}(h_i(D)) = \nu\}$$

$$\mathcal{H}_{III} = \{h_i(D), i = 1, 2, \ldots, c \mid \text{del}(h_i(D)) > 0\}.$$

If a parity-check polynomial fulfills the conditions for two sets, it will be assigned to an arbitrary one.

By reordering the parity-check polynomials in the matrix module $\tilde{H}$ it is possible to obtain an equivalent convolutional code, whose matrix module $\tilde{H}_{eq}$ consists of the columns ordered such that the first columns in $\tilde{H}_{eq}$ belong to $\mathcal{H}_I$, followed by the columns of $\mathcal{H}_{II}$, and finally by those of $\mathcal{H}_{III}$, as illustrated in Fig. 4.

Next, we let the span [13] for each row in (3) denote the interval starting with its first and ending with its last nonzero
value. A certain column of a row is considered to be active if it lies within the span, but is not the last column of the span \([1]\). Directly related, the number of states \(|V_i|\) at the \(i\)th level of the trellis is given by

\[
|V_i| = 2^{a_i}
\]  
(7)

where \(a_i\) denotes the number of active rows in the \(i\)th column, \(i = 1, 2, \ldots, c\), of a parity-check matrix. As the trellis is constructed by connecting the identically full syndrome trellis modules, the state complexity \(s\) is fully determined by a single matrix module. By combining (1) and (7) we obtain

\[
s = \max \{a_i\}, \quad i = 1, 2, \ldots, c.
\]  
(8)

Whether a row in (3) is considered to be active at a certain position is obviously determined by the matrix module \(H\). For a rate \(R = \frac{(c-1)}{c}\) convolutional code, the last nonzero value of a row is determined by the first row of the matrix module, whereas the first nonzero value of a row is determined by the last row of the matrix module. In other words, in each matrix module, its first row ends being active and its last row starts being active (cf. Fig. 5).

Having a closer look at the binary syndrome trellis module, every valid path, that is, every valid partial codeword, corresponds to a linear combination of columns of the parity-check matrix module. Every valid codeword \(v\) has to fulfill the zero-constraint \(vH^T = 0\). Consider an arbitrary row \(j\), \(j = 1, 2, \ldots\). The span of this row ends within the first row of a certain matrix module \(H\). To fulfill the zero-constraint, that is the linear combination of columns determined by the codeword, the syndrome bit \(j\) must be zero already after adding the first \(j\) columns.

Next we take a closer look at the sets \(H_I, H_{II},\) and \(H_{III}\). To store every possible linear combination of parity-check polynomials from \(H_I\) we need at most \(2^\nu\) memory elements, while the same holds for every linear combination of parity-check polynomials from \(H_{II}\). For the set \(H_{III}\), however, we will distinguish four different cases:

- \(H_{III}\) is empty and thus the span of every row of the matrix \(H\) ends with a column of \(H_I\). Having fulfilled the zero-constraint with a linear combination of parity-check polynomials from \(H_I\), we have used at most \(\nu\) memory elements. Due to the zero-constraint the first partial syndrome bit is zero and will stay so. Continuously adding parity-check polynomials from \(H_{III}\), we update at most the last \(\nu\) bits of the current partial syndrome. Consequently there is no need for more than \(\nu\) memory elements at any time.
- \(H_{II}\) contains one parity-check polynomial and the span of the first row of the matrix module ends with the single column in \(H_{II}\). If, after the linear combination of parity-check polynomials from \(H_I\), the first bit is one, the parity-check polynomial in \(H_{II}\) will be added to fulfill the zero-constraint. That is, it forces the first bit to get and stay zero and thereby only the last \(\nu\) bits have to be stored. Proceeding with parity-check polynomials from \(H_{III}\) does not increase the memory requirements as previously explained.
- \(H_{II}\) contains two parity-check polynomials and thereby the span ends in the second of the two columns in \(H_{II}\). For rate \(R = \frac{(c-1)}{c}\) convolutional codes, one codeword bit within a codeword \(c\)-tuple can be determined from the other \(c - 1\) codeword bits. By sectionizing it is possible to combine two parity-check polynomials into a single step, while still preserving the properties of a binary syndrome trellis. If the first bit is already zero after combining the columns from \(H_I\), these two parity-check polynomials are either both added or none of them is added. If the first bit is one after combining the columns from \(H_I\), only one of these two polynomials is added and thereby the zero-constraint is fulfilled. We continue analogously to the previous case with one parity-check polynomial in \(H_{III}\).
- \(H_{II}\) contains more than two parity-check polynomials. As it is not possible to combine those parity-check polynomials by sectionizing without violating the binary syndrome trellis property, we need to have \(\nu + 1\) memory elements at least for those layers, and thereby \(2^\nu + 1\) states.

We will now summarize these results in a theorem:

**Theorem 1:** Consider a rate \(R = \frac{(c-1)}{c}\) convolutional code, \(c \geq 2\), with overall constraint length \(\nu\), whose parity-check polynomials are assigned to the sets \(H_I, H_{II},\) and \(H_{III}\) according to their delay, difference of degree and delay, and degree, respectively. Then, if and only if \(|H_{II}| \leq 2\), the binary syndrome trellis (possibly sectionized) can be realized with \(2^\nu\) different states at every layer. This corresponds to a
maximum number of $\nu$ active rows at any column, that is, a state complexity of $s = \nu$.

### III. THE SYNDROME BEAST

The BEAST—Bidirectional Efficient Algorithm for Searching code Trees—was introduced in [14] and [15]. Based on (binary) trees obtained from generator matrices, it was used both for code search [15] and for decoding of block codes [16].

However, with only minor modifications it is possible to use a (binary) syndrome tree with the BEAST. Consider a rate $R = b/c$ convolutional code and let $\xi$ and $s(\xi)$ denote a node in the syndrome tree and its corresponding partial syndrome, respectively. Every node $\xi$ has a unique parent node $\xi^P$ and $2^b$ children, referred to as $\xi^{\{i\}}$. For every valid codeword $v$ of weight $\omega$ there exists a path $\xi_{\text{root}} \rightarrow \xi$, with $s(\xi_{\text{root}}) = s(\xi_{\text{root}}) = 0$. Hence, for each such path, there exists an intermediate node $\xi$ with $s(\xi) \neq 0$, such that

$$w'_P(\xi) = f_w + j \quad w_B(\xi) = b_w - j \quad j = 0, 1, \ldots, c - 1$$

where $w'_P$ and $w_B$ denote the accumulated branch weights for the sub-paths $\xi_{\text{root}} \rightarrow \xi$ and $\xi \rightarrow \xi_{\text{root}}$, respectively, and

$$f_w = \left\lfloor \frac{\omega}{2} \right\rfloor \quad b_w = \left\lceil \frac{\omega}{2} \right\rceil.$$

Based on these observations, BEAST performs the following steps, searching for the number of codewords of weight $\omega$:

1) **Forward search:** Starting at the zero-weight root, extend the forward syndrome code tree to obtain $c$ sets of nodes, indexed by $j = 0, 1, \ldots, c - 1$,

$$\mathcal{F}_{j} = \left\{ \xi | w'_P(\xi) = f_w + j, w'_P(\xi^{P}) < f_w, s(\xi) \neq 0 \right\}$$

2) **Backward search:** Starting at the zero-weight root, extend the backward syndrome code tree to obtain $c$ sets of nodes, indexed by $j = 0, 1, \ldots, c - 1$,

$$\mathcal{B}_{j} = \left\{ \xi | w_B(\xi) = b_w - j, w_B(\xi^{C}) > b_w, s(\xi) \neq 0 \right\}$$

3) **Matching:** For every pair $\{\mathcal{F}_j, \mathcal{B}_j\}$, $j = 0, 1, \ldots, c - 1$, count the number of matching node pairs $\{\xi, \xi'\}$ with equal partial syndrome, i.e., $s(\xi) = s(\xi')$, $\xi \in \mathcal{F}_j$ and $\xi' \in \mathcal{B}_j$. Thereby, the number of codewords $n_\omega$ of weight $\omega$ is determined by

$$n_\omega = \sum_{j=0}^{c-1} \sum_{\xi, \xi' \in \mathcal{F}_j \times \mathcal{B}_j} \chi(\xi, \xi')$$

where $\chi$ is the match-indicator function defined as

$$\chi(\xi, \xi') = \begin{cases} 1, & \text{if } s(\xi) = s(\xi') \\ 0, & \text{otherwise.} \end{cases}$$

**Remark:** Note that although we can use the binary syndrome tree with the BEAST we always have to complete the processing of a trellis module before storing the nodes in their appropriate sets.

### IV. RESULTS

Using the syndrome BEAST, rate $R = (c - 1)/c$ convolutional codes with optimal free distance are obtained for various overall constraints lengths $\nu$. In Table I we give the first seven spectral components for optimum free distance, rate $R = 2/3$ convolutional parity-check polynomials with $\nu = 1, 2, \ldots, 13$ in the following octal notation: $56 \equiv 101100 \equiv 1 + D^2 + D^3 + D^4$. Table III and Table V give similar results with the first six spectral components for rate $R = 3/4$, and rate $R = 4/5$ convolutional parity-check polynomials with $\nu = 1, 2, \ldots, 10$ and $\nu = 1, 2, \ldots, 9$, respectively.

Searching for convolutional codes fulfilling Theorem 1, we obtain the rate $R = 2/3$ convolutional parity-check polynomials with $\nu = 1, 2, \ldots, 13$ given in Table II, rate $R = 3/4$ convolutional parity-check polynomials with $\nu = 1, 2, \ldots, 10$ given in Table IV, and rate $R = 4/5$ convolutional parity-check polynomials with $\nu = 1, 2, \ldots, 9$ given in Table VI.

Comparing these results, it becomes obvious that by imposing the restrictions in Theorem 1, the performance of convolutional codes is not severely deteriorated. In most cases the same free distance $d_{\text{free}}$ can be achieved and only a minor increase in the number of spectral components has to be accepted. On the other hand, decoding such convolutional

<table>
<thead>
<tr>
<th>$\nu$</th>
<th>polynomials</th>
<th>$d_{\text{free}}$</th>
<th>spectrum</th>
</tr>
</thead>
</table>
| 1     | 6 6 4       | 2              | 1, 2, 4, 10, 20, 40, 80 |}
| 2     | 7 6 5       | 3              | 1, 4, 14, 40, 116, 339, 999 |}
| 3     | 74 64 54    | 4              | 1, 5, 24, 71, 238, 862, 2991 |}
| 4     | 62 56 52    | 5              | 2, 13, 45, 143, 534, 2014, 736 |}
| 5     | 61 55 53    | 6              | 6, 27, 70, 285, 1103, 4063, 15359 |}
| 6     | 634 514 504 | 7              | 17, 53, 133, 569, 2327, 8624, 32412 |}
| 7     | 722 662 576 | 8              | 41, 0, 528, 0, 7497, 0, 11017 |}
| 8     | 631 555 477 | 9              | 6, 42, 153, 150, 1853, 7338, 28378 |}
| 9     | 7264 6214 4504 | 10             | 17, 81, 228, 933, 3469, 13203, 51286 |}
| 10    | 7642 6406 4232 | 11             | 69, 0, 925, 0, 13189, 0, 197340 |}
| 11    | 7741 6667 5175 | 12             | 10, 80, 260, 864, 3336, 13131, 50279 |}
| 12    | 42074 70754 62964 | 13             | 32, 144, 477, 1769, 6718, 25717, 98945 |}
| 13    | 52536 72166 60902 | 14             | 116, 0, 1768, 0, 24984, 0, 370584 |}
| 14    | 71341 64657 40773 | 15             | 22, 134, 464, 1702, 6477, 24767, 94527 |}
codes can be performed with much less complexity, as their binary trellises can be implemented without increasing the state complexity and thereby with a smaller amount of memory elements.

Note that although most of the parity-check polynomials given in Table I-VI and their corresponding generator matrices have been listed in previous publications [3]–[9], [17], [18], their optimum free distance property was mostly unknown.

ACKNOWLEDGMENTS

This research was supported in part by the Swedish Research Council under Grant 621-2007-6281.

REFERENCES


[19] ——, “Rate 3/4 optimum free distance convolutional codes fulfilling Theorem 1.”