Optimal Rate-Compatible Punctured Concatenated Zigzag Codes

Song-Nam Hong and Dong-Joon Shin
Division of Electrical and Computer Engineering
Hanyang University, Seoul, Korea
sunny795@ccrl.hanyang.ac.kr, djshin@hanyang.ac.kr

Abstract—In the next-generation mobile communication systems for various high-speed data services, the error correcting codes are required to have rate-compatibility, low decoding complexity and good performance for various frame lengths. In this paper, new rate-compatible punctured concatenated zigzag (RCPCZ) codes are proposed and analyzed by using the density evolution. As their application, Type-II HARQ using RCPCZ code is shown to have better throughput at short frame lengths than Type-II HARQ using Turbo code.

Keywords – Concatenated zigzag codes, density evolution, rate compatibility, puncturing, HARQ.

I. INTRODUCTION

The next-generation mobile communication systems should support high data rate and asymmetric transmission as in the HSDPA (High Speed Downlink Packet Access) system [1]. Therefore, FER (frame error rate) should be down to $10^{-4}$~$10^{-5}$ at the target SNR range and the latency should be within 80ms. To satisfy these requirements, the error-correcting codes should have rate compatibility and good performance for various frame lengths.

The concept of rate-compatible codes was presented for the first time in [2], where rate-compatible convolutional codes are obtained by adding a rate-compatibility restriction to the puncturing rule. This restriction requires that the rates are organized in the way that all the unpunctured coded bits of a higher rate code should also be the unpunctured bits of the lower rate codes. This concept was extended to Turbo codes, called RCTC [3]. Turbo codes have an error floor in the region of moderate-to-high SNR range due to low-weight codewords [4]. Especially, at high code rates and short frame lengths, the effect of error floor becomes more serious, which is an obstacle to the reliable data transmission. However, concatenated zigzag (CZZ) code shows little error floor problem even at high code rates and short frame lengths [5]. Therefore, it is better suited for short packet transmission with various code rates. The rate-compatible CZZ code constructed by multiple zigzag codes is proposed in [12].

In this paper, we propose new rate-compatible codes, called rate-compatible punctured concatenated zigzag (RCPCZ) codes, which are constructed by combining the best CZZ code for a fixed rate with the optimal rate-compatible puncturing patterns. For a fixed rate, various CZZ codes can be constructed by using different number of component codes (zigzag codes). To find the best CZZ code for a fixed rate, we obtain the threshold values by using the density evolution, similarly as the LDPC and IRA codes [6], [7], [8]. Further, by modifying the recursion equations, this analysis is extended to the punctured CZZ codes and threshold values for various puncturing patterns are obtained. The criteria for designing good puncturing patterns are suggested and accordingly the rate-compatible puncturing patterns are obtained. By employing RCPCZ codes in Type-II HARQ based on incremental redundancy (IR) retransmission, we can obtain better throughput at short frame lengths than Type-II HARQ using rate-compatible Turbo code (RCTC-HARQ) adopted in 3GPP.

II. OVERVIEW OF CONCATENATED ZIGZAG CODES

A zigzag code is discovered by Ping [5], which has simple tree structure without cycle. The error correcting capability of this code itself is very weak but it is very useful as a component code to construct concatenated codes. The CZZ code is illustrated in Fig. 1. The dashed box represents an overall interleaver consisting of smaller sub-
interleavers, \( \pi_1, \pi_2, \ldots \), and white and black circle nodes represent information and parity bits, respectively. Both nodes are commonly called variable nodes and the square nodes are called check nodes. The parity bits are chosen such that the variable nodes connected to the check node contain an even number of 1’s. The \((N_c, d_c)\)-CZZ code denotes the parallel concatenation of \( N_c \) \((d_c - 2, 0)\)-zigzag codes, which is of code rate \((d_c - 2)/(d_c + N_c - 2)\). The CZZ codes have lower decoding complexity than turbo codes, faster convergence speed than LDPC codes, and linear time encoding. The overall encoding and decoding algorithms of CZZ codes can be found in [5], [9].

III. ANALYSIS OF CONCATENATED ZIGZAG CODES

We analyze the CZZ codes using density evolution and derive important properties. Further, this analysis is applied to the punctured zigzag codes and the criteria for designing good rate-compatible puncturing patterns are suggested.

A. Optimal component codes

Since CZZ code can have various structures for a fixed code rate, it is important to find the optimal \((N_c, d_c)\)-CZZ code. We will apply the density evolution with Gaussian approximation to the CZZ codes when two-way scheduling is used, which is used for deriving important properties of CZZ code. To simplify the calculation, we assume that all-1 codeword is transmitted as in [7]. We denote that \( m_{u_0} \) represents the mean of the message at a variable node from the channel, and \( N_{ch} \) is the number of check nodes in each zigzag code. If the information nodes are connected to the same check node, the means of messages passed from information nodes to the check node are the same. Therefore, we do not distinguish the information nodes connected to the same check node and only consider \( N_{ch} \) information nodes. We assume that the zigzag interleaver randomizes the information bits perfectly and the number of information bits is sufficiently large. The density evolution analysis is done by two levels considering messages passed between zigzag decoders and messages passed in each zigzag decoder.

First, we consider the messages passed between zigzag decoders. In the \( l \)th component decoder at the \( l \)th iteration, \( m_{u(k)}^{(l,i)}(j) \) and \( m_{r(k)}^{(l,i)}(j) \) denote the means of messages passed from the \( j \)th check node to information node and vice versa. \( m_{u(k)}^{(l,i)}(j) \) and \( m_{r(k)}^{(l,i)}(j) \) correspond to (a) and (b) in Fig. 2, respectively. Let \( u^{(l,j)} \) be the randomly selected message passed from check node to information node. Then, the probability of \( u^{(l,k)} = u^{(l,k)}(j) \) is equal to \( 1/N_{ch} \) for all \( j \). Therefore, \( u^{(l,k)} \) has the following Gaussian mixture density \( f_u^{(l,k)}(\cdot) \):

\[
f_{u^{(l,k)}}(x) = \frac{1}{N_{ch}} \sum_{j=1}^{N_{ch}} f_{u^{(l,k)}(j)}(x), k = 1, \cdots, N_c,
\]

where \( f_{u^{(l,k)}(j)}(\cdot) \) means the Gaussian probability density with mean \( m_{u^{(l,k)}(j)} \) and variance \( 2m_{u^{(l,k)}(j)} \).

The message \( v^{(l+1,k)} \) is computed by

\[
v^{(l+1,k)} = \sum_{n=1}^{N_c} u^{(l,n)}, \quad k = 1, \cdots, N_c.
\]

\( v^{(l+1,k)} \) is the sum of \( u^{(l,n)} \) which can be assumed Gaussian since \( u^{(l,k)}(j) \)'s have the almost same mean for all \( j \). Therefore, \( v^{(l+1,k)} \) is well approximated by Gaussian random variable and only the mean and variance are needed, which can be calculated as follows.

\[
E[v^{(l+1,k)}] = \sum_{n=1}^{N_c} E[u^{(l,n)}] = (N_c - 1)E[u^{(l,1)}]
\]

where \( n \) is eliminated since \( E[u^{(l,n)}] \) does not depend on \( n \).

\[
\text{VAR}[v^{(l+1,k)}] = (N_c - 1)\text{VAR}[u^{(l,1)}]
\]

Since \( v^{(l+1,k)} \) also satisfies the symmetry condition [3], we only need to track the mean.

Second, we consider the message passed in each zigzag decoder. We denote the means of messages passed from the \( j \)th check node to the left-parity node (\( p_{l,j} \) in Fig. 2) and the right-parity node (\( p_{r,j} \) in Fig. 2) at the \( l \)th iteration by \( m_{p_{l,j}}^{(l,0)} \) and \( m_{p_{r,j}}^{(l,0)} \), respectively. These corresponding to (c) and (d) in Fig. 2 are only used in each zigzag decoder and not passed to any other zigzag decoder. Therefore, the superscript to denote the zigzag decoder can be omitted.

Two-way scheduling used in each zigzag decoder is divided into three levels [5]. Three recursion equations obtained by density evolution are derived by considering these three levels separately as follows.

\[
m_{p_{l,j}}^{(l,0)} = \phi^{-1}\left[ -\phi(m_{u_{l,j}} + (N_c - 1)m_{p_{r,j}}) \right],
\]

\[
m_{p_{r,j}}^{(l,0)} = \phi^{-1}\left[ -\phi(m_{u_{r,j}} + (N_c - 1)m_{p_{l,j}}) \right],
\]

\[
m_{u_{l,j}}^{(l,0)} = \phi^{-1}\left[ -\phi(m_{u_{r,j}} + (N_c - 1)m_{u_{l,j}}) \right],
\]

where \( m_{u^{(l,j)}} = \frac{1}{N_{ch}} \sum_{j=1}^{N_{ch}} m_{u^{(l,j)}} \).
In the third equation, \( m_{u(j)}^{(i)} \) is used since it does not depend on \( k \). By using these equations recursively, we can compute the threshold values for various CZZ codes.

In most of mobile communication systems, the code rate of mother error-correcting code is usually \( 1/n \) and by modifying this code (e.g. using puncturing), higher code rates can be achieved. Therefore, we only consider the \((N_c,d_c)\)-CZZ codes of rate \( 1/n \). From Table I, we can see that CZZ codes having four component codes show the best performance for code rates \( 1/2, 1/3 \) and \( 1/5 \) and the CZZ codes having the minimum number of component codes are optimal for the rates less than \( 1/5 \).

B. Selection of punctured nodes

By puncturing CZZ codes, higher code rates can be achieved and since the performance of punctured codes depends on the puncturing patterns, the analysis should be done for various puncturing patterns. Among variable nodes, only the parity nodes are punctured, which have degree 2 for CZZ codes. Then, since the number of edges connected to the punctured nodes is minimized, the number of erasure messages which are initially propagated through the edges in the iterative decoding is also minimized and the performance degradation due to puncturing can be reduced. We show the asymptotic performance according to the puncturing positions by using density evolution and derive some results necessary for the performance according to the puncturing positions by using recursion equations in III-A. The messages \( u(i)(j) \)'s do not have the same mean values because of the punctured parity nodes. Therefore, Gaussian approximation of \( v(i) \) is not valid anymore. However, the mean values of \( u(i)(j) \)'s can take a few values and the probability density function of \( v(i) \) becomes a simple Gaussian mixture density function. Suppose that \( u(i)(j) \)'s can take \( n \) different mean values \( m_1 > m_2 > \cdots > m_n \) and \( f_i \) denotes the number of \( u(i)(j) \)'s having the mean \( m_i \) in a puncturing period. \( v(i) \) is the sum of messages from the \((N_c-1)\) check nodes other than the check node receiving the messages and the number of distinct means and their occurring probabilities can be calculated by considering the puncturing patterns. Let \( \mu_i \) be the arbitrary mean value of \( v(i) \) and \( P_i \) be the occurring probability of \( \mu_i \). Then, the mixture probability density function of \( v(i) \) becomes,

\[
 f_{\mu(i)}(x) = \sum_i P_i f_{\mu(i)}(x)
\]

where \( f_m(\cdot) \) means the Gaussian probability density function with mean \( m \) and variance \( 2m \).

The above procedure will be explained in detail by considering the optimal \((4,4)\)-CZZ code of rate-\(1/3\). Suppose that the same puncturing is performed periodically on the blocks of eight parity bits from each zigzag encoder. To obtain the rate-\(2/3\) code by puncturing, 6 bits should be punctured from each block. Let's consider the puncturing pattern \([10010000]\) where 1 and 0 represent unpunctured and punctured parity nodes, respectively. Fig. 3 represents the subgraph corresponding to one puncturing period of a zigzag code. White and black circle nodes correspond to punctured and unpunctured variable nodes, respectively. Denote a puncturing pattern by \((L_1,L_2,\ldots,L_k)\) or \((l_1,l_2,\ldots,l_k)\) where \( L_i \) is the set of \( i \)th and \((i+1)\)st unpunctured parity nodes and punctured nodes between them and \( l_i \) is the distance between \( i \)th and \((i+1)\)st unpunctured parity nodes. Then, the puncturing pattern in Fig. 3 can be denoted by \((\{0,1,2,3\},\{3,4,5,6,7,0\})\) or \((3,5)\).

By using three recursion equations, we can compute the mean of message \( u(i)(j) \)'s from check node to information node and we get the following two facts.

(i) The messages \( u(i)(j) \)'s from the check nodes connected to the parity nodes in \( L_i \) have the same mean values.

(ii) The messages \( u(i)(j) \)'s from the check nodes connected to the parity nodes in \( L_i \) at any

---

### Table I

<table>
<thead>
<tr>
<th>Rate</th>
<th>( \sigma^* )</th>
<th>( N_c )</th>
<th>( d_c )</th>
<th>Rate</th>
<th>( \sigma^* )</th>
<th>( N_c )</th>
<th>( d_c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/8</td>
<td>1.952</td>
<td>7</td>
<td>3</td>
<td>1/5</td>
<td>1.541</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>1.830</td>
<td>14</td>
<td>4</td>
<td>4</td>
<td>1.502</td>
<td>8</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>1.840</td>
<td>21</td>
<td>5</td>
<td>5</td>
<td>4.131</td>
<td>12</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1.837</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>0.891</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1.733</td>
<td>12</td>
<td>4</td>
<td>4</td>
<td>1.159</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1.645</td>
<td>18</td>
<td>5</td>
<td>5</td>
<td>1.145</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1.702</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>0.877</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1.623</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>0.894</td>
<td>4</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>1.541</td>
<td>15</td>
<td>5</td>
<td>5</td>
<td>0.887</td>
<td>5</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>Rate</th>
<th>Puncturing Pattern</th>
<th>Threshold value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/3</td>
<td>(1,7), (7,1)</td>
<td>0.662</td>
</tr>
<tr>
<td></td>
<td>(2,6), (6,2)</td>
<td>0.694</td>
</tr>
<tr>
<td></td>
<td>(3,5), (5,3)</td>
<td>0.708</td>
</tr>
<tr>
<td></td>
<td>(4,4)</td>
<td>0.712</td>
</tr>
</tbody>
</table>
puncturing period have the same mean values.

In Fig. 3, there are two different mean values of \( u^{(i)}(j) \)'s since \((l_1, l_2) = (3, 5)\). Suppose that in Fig. 2, \( u^{(i)}(j) \)'s in the dotted circle have the same mean \( m_1 \) and others \( m_2 \). Then, \( f_1 \) and \( f_2 \) are 6 and 10, respectively. To derive the probability density function of \( u^{(i)} \), define an \( N_c \times N_{ch} \) in which the \( j \)th element of the \( i \)th row represent the mean of message passed from the check node in the \( j \)th zigzag code to the \( i \)th information node in a puncturing period. The order of elements in the \( i \)th row is determined by the interleaver connected to the \( i \)th zigzag code, which affects the performance of punctured CZZ code. To evaluate the interleaver performance, we define a random variable \( R \) having the mean of \( u^{(i)} \) as its values. Through calculating the threshold values for varying \( R \), we can see that the performance improves as the variance of \( R \) decreases. The 4×8 mean matrix in Fig. 4 can be a good choice for this example. From Fig. 4, we get three different \( m_v(i) \)'s such as \( m_v(1) = 2m_1 + m_2 \), \( m_v(2) = m_1 + 2m_2 \) and \( m_v(3) = 3m_2 \).

The corresponding occurring probabilities \( P_1 \), \( P_2 \) and \( P_3 \) are 8/32, 20/32 and 4/32, respectively. By using the above results, we get the following probability density function of \( u^{(i)} \).

\[
f_{m_v(1)}(x) = \frac{8}{32} f_{m_v(2)}(x) + \frac{20}{32} f_{m_v(2)}(x) + \frac{4}{32} f_{m_v(3)}(x).
\]

By obtaining the threshold values for various puncturing patterns, we acquire the following two results.

1. CZZ codes punctured by two patterns \( L \) and \( L' \) show the same performance if \( L \) can be obtained by cyclically shifting.

2. As the unpunctured parity bits are located as regularly as possible, the performance becomes better.

The above results are verified by Table II and the corresponding simulation results are shown in Fig. 5.

IV. RCPCZ CODES AND THEIR APPLICATION

In this section, the RCPCZ codes are constructed and Type-II HARQ using RCPCZ code is compared with Type-II HARQ using RCTC.

![One puncturing period=8](image)

Fig. 3. A subgraph corresponding to one puncturing period of length 8 for (2,0)-zigzag code

![Mean matrix for the example](image)

Fig. 4. Mean matrix for the example

TABLE III

<table>
<thead>
<tr>
<th>Overall Rate</th>
<th>Puncturing pattern for 8-parity-bit block</th>
</tr>
</thead>
<tbody>
<tr>
<td>4/5</td>
<td>100000000</td>
</tr>
<tr>
<td>4/6</td>
<td>1001000</td>
</tr>
<tr>
<td>4/7</td>
<td>10100100</td>
</tr>
<tr>
<td>4/8</td>
<td>1010100</td>
</tr>
<tr>
<td>4/9</td>
<td>110101000</td>
</tr>
<tr>
<td>4/10</td>
<td>111011100</td>
</tr>
<tr>
<td>4/11</td>
<td>1111101</td>
</tr>
<tr>
<td>4/12</td>
<td>1111111</td>
</tr>
</tbody>
</table>

A. Construction of RCPCZ Codes

New rate compatible punctured CZZ (RCPCZ) codes are constructed by combining the optimal CZZ code derived in subsection III-A and good rate compatible puncturing patterns. The criteria to design good rate-compatible puncturing patterns are given as follows, which are based on the results in the previous section. Suppose that we want to find the puncturing patterns for the optimal CZZ code of rate \( k/n_{\text{max}} \) to achieve code rates \( k/n_1 > k/n_2 > \ldots > k/n_{\text{max}} \).

(a) Determine the puncturing period as \( n_{\text{max}} = k \)

(b) To achieve lower rates, unpunctured parity bits are inserted into a puncturing period such that the unpunctured parity bits are located as regularly as possible.

We show how to construct a RCPCZ code using an example. Suppose that we want to design an RCPCZ code achieving eight code rates 4/5, 4/6, ..., 4/12. Since the lowest target code rate is 1/3, the optimal CZZ code of rate 1/3 code, the (4,4)-CZZ code is chosen as the mother code. The puncturing period is determined by (a) such as 12−4=8 and the corresponding puncturing patterns are obtained by (b) as in TABLE III.

B. Comparison of Type-II HARQ Using RCPCZ Code and RCTC

It is natural to use RCPCZ codes in Type-II HARQ system based on incremental redundancy and it is compared with Type-II HARQ with RCTC which is adopted in 3GPP2.

For the simulation, 496-information bits, BPSK modulation, Min-sum algorithm with scaling, random interleaver, 20 iterations and AWGN channel are assumed. It is also assumed that the noiseless feedback link is available so that the receiver can reliably inform the transmitter of the decoding result. The round trip delay is not considered. The CRC code of length 496 with the generator polynomial \( x^{16} + x^{15} + x^2 + 1 \) is used. RCPCZ code given in IV-A with the eight puncturing patterns of Table II and the RCTC with eight puncturing patterns given in [10] are used for the simulation.
Fig. 6 shows that RCPCZ code has good rate compatibility and no error floor at all code rates. In Fig. 7, we show that HARQ-RCPCZ has better throughput than HARQ-RCTC. When the decoding delay and complexity are considered, our scheme has smaller decoding delay and lower complexity than RCTC-HARQ scheme.

V. CONCLUSIONS

In this paper, by applying density evolution to the CZZ codes when two-way scheduling is used, the threshold values are obtained and the optimal CZZ codes for the given code rates are derived. Further, density evolution is applied to punctured CZZ codes and the requirements to find the optimal puncturing patterns are obtained. As their application, new Type-II HARQ is constructed and compared with the existing Type-II HARQ using Turbo code.

REFERENCES


