Systematic encoders for convolutional codes and their application to turbo codes
Sergio Benedetto, Roberto Garello, Guido Montorsi

Dipartimento di Elettronica, Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
e-mail: authorname@polito.it

Abstract - Systematic recursive convolutional encoders have been shown to play a crucial role in the design and implementation of parallel concatenated codes ("turbo codes"). We present here a canonical structure of minimal linear systematic rate k/n encoders and use it in the search for good constituent codes of parallel concatenated codes. Tables of the best encoders for various rates are also presented.

I. INTRODUCTION

The algebraic structure of convolutional codes has been thoroughly illustrated by Forney in the basic paper [1]. Among other results, he generalized a Costello observation [2] on the fact that feedback convolutional encoders can be described as minimal feedforward equivalents. Encoders are equivalent in the sense that they generate the same set of code sequences. As such, the error event probability is the same for both classes of encoders. In terms of bit error probability, however, the behavior is different, since different encoders yield different input-output mappings. Easy ways to pass from feedforward to feedback encoders and vice versa have been presented in [3].

Recently, equivalent non systematic and systematic recursive rate 1/n convolutional encoders have been compared in [4], with the aim of using systematic codes as constituent codes (CCs) of parallel concatenated convolutional codes (PCCCs), also known as "turbo codes". The importance for a CC of turbo codes to be both systematic (for decoding simplicity) and recursive (to maximize the interleaver gain) has been assessed in [5], where design guidelines were also proposed for rate 1/n CCs. In [6] the results of [5] were extended to rate k/n CCs and tables of "optimum" CCs presented.

In this paper, we use group code theory [7] to introduce a canonical structure of minimal linear systematic recursive encoders for rate k/n convolutional codes and use it in the search of "optimum" CCs of various rates to be embedded in turbo codes. An analytical example of performance of the obtained turbo codes is also shown.

II. CONVOLUTIONAL CODES AS GROUP CODES

A binary linear convolutional code C with rate k/n and 2^r states, is a time-invariant, finite-state, (complete, controllable), group code over Y = Z_2^n. We can use the theory developed in [7] to state several properties for C:

1. C is a subgroup of the direct product space Y times Z_2; every code sequence c E C is a two-sided infinite sequence c = (..., c_{-1}, c_0, c_1, ...).
2. C is characterized by a canonical state group Sigma, that inherits a group structure from C and is isomorphic to Z_2^[r] : Sigma times Z_2^[r]. Every code sequence c E C passes through a unique state sigma_k(c) E Sigma at each time k and any c E C corresponds to a unique state sequence sigma.
3. The canonical state-output trellis section of C is the group T E Sigma times Y times Sigma of edges (sigma_k(c), c_k, sigma_{k+1}(c)) for all c E C. There exists an edge (sigma, y, sigma') E T if there is a sequence c E C that passes through state sigma at time k, state sigma' at time k + 1 and has output c_k = y at time k. Any c E C corresponds to a unique edge sequence. The set of edges leaving the zero state sigma_0: T_0 = {(sigma_0, y, sigma_0)} is a 2^r-element group and the set of edges leaving the other states are cosets of T_0. The elements y that label the edges of T_0 are all distinct and forms a group U E Z_2^[r].
4. The trellis section of C can be interpreted as a directed, labeled graph. An edge (sigma, y, sigma') E T corresponds to a directed arc from node sigma to node sigma' with label y; the same number of edges enter and exit each node. The state transition structure is that of a shift-register (de Bruijn) graph. 5. C may be characterized by a set of k minimal length generators g^{(1)}, ..., g^{(k)} E C; they are code sequences that cannot be obtained as superpositions of shorter sequences. Every c E C can be written, in a unique way, by adding shifted version of the generators:

\[ c = \sum_{i=1}^{k} \sum_{j \in S_i} D_{ij} g^{(i)} \] (1)

with S_i subset Z. The cardinality 2^r of the state group Sigma is strictly tied to the generator lengths: if g^{(i)} = (g_{0}^{(i)}, ..., g_{r}^{(i)}), then v = sum_{i=1}^{k} l_i.

Example 1: The trellis section T of a 4-state, rate 2/3 convolutional code C is shown below. The sixteen edges are listed in tabular form that shows the coset structure of T.

<table>
<thead>
<tr>
<th>sigma</th>
<th>y</th>
<th>sigma'</th>
<th>sigma</th>
<th>y</th>
<th>sigma'</th>
<th>sigma</th>
<th>y</th>
<th>sigma'</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>00</td>
<td>11</td>
<td>10</td>
<td>01</td>
<td>10</td>
<td>10</td>
<td>01</td>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>00</td>
<td>10</td>
<td>01</td>
<td>01</td>
<td>11</td>
<td>01</td>
<td>11</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>00</td>
<td>01</td>
<td>11</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>01</td>
<td>00</td>
<td>11</td>
</tr>
</tbody>
</table>

The two generators are g^{(1)} = (111,101) and g^{(2)} = (100,011).

This work has been supported by Agenzia Spaziale Italiana.
III. Minimal encoders for convolutional codes

A minimal encoder \( E \) for \( C \) is a finite-state machine that causally maps unconstrained input sequences \( u \in (\mathbb{Z}_2^n)^* \) into code sequences \( c \in C \). \( E \) is characterized by a state space (that we will also call \( \Sigma \)) in a one-to-one correspondence with the state group \( \Sigma \) of \( C \), an input set \( U = \mathbb{Z}_2^n \) and a one-to-one function \( e : \Sigma \times U \rightarrow T \), \( e(\sigma, u) = (\sigma, y, \sigma') \). At time \( k \in \mathbb{Z} \) the encoder \( E \), which is in state \( \sigma_k \in \Sigma \), receives an input element \( u_k \in U \) and selects the edge \( e(\sigma_k, u_k) = (\sigma_k, y_k, \sigma_{k+1}) \in T \) that determines the output element \( y_k \) and the future state \( \sigma_{k+1} \). Since any code sequence \( c \in C \) corresponds to a unique edge sequence, and any edge corresponds to a unique \( u \in U \), then any \( c \in C \) is generated by \( E \) via a unique input sequence \( u \in U^2 \); the encoder is invertible and the set of all pairs \( (u, c) \) is denoted by \( \tilde{C} \).

We remark here that a minimal encoder \( E \) cannot be catastrophic; given two code sequences that differ in a finite number of symbols, the input sequences generating them also differ in a finite number of symbols.

Starting from the \( 2^{(k+\nu)} \) elements of the canonical minimal state-output trellis section \( T \) of \( C \), we introduce the input-state-output extended trellis section \( T_E \) as the \( 2^{(k+\nu)} \) set of 4-tuples \( (\sigma, u, y, \sigma') \) where \( (\sigma, y, \sigma') \in T \) and \( e(\sigma, u) = (\sigma, y, \sigma') \).

Again, \( T_E \) can be interpreted as the directed graph introduced for \( T \), extended by adding the input symbols to the labels of the arcs. This approach permits to enumerate all different minimal encoders \( E \) for a given code \( C \); they are \( [2^k]^2 \). In fact, for each \( \sigma_1 \in \Sigma \), we have \( 2^k \) different possible associations in \( e \) between elements of \( u \in \mathbb{Z}_2^n \) and triplets \( (\sigma_1, y, \sigma') \in T \).

A minimal encoder for \( E \) is linear (homomorphic) if

\[
\forall (u_1, c_1) \in \tilde{C}, \quad \forall (u_2, c_2) \in \tilde{C} \quad \rightarrow \quad (u_1 + u_2, c_1 + c_2) \in \tilde{C}.
\]

As a particular case of a theorem holding for general group codes, we have:

**Theorem 1:** A minimal encoder \( E \) is linear if and only if \( T_E \) is a group. In this case

\[
T_E = T_{\Sigma} \times T_U
\]

where

\[
T_{\Sigma} = \{(\sigma_0, u_i, y_i, \sigma_j)\} \quad \text{for all} \quad u_i \in U
\]

is the group of edges leaving the identity state \( \sigma_0 \) and

\[
T_U = \{(\sigma_j, u_0, y, \sigma'_j)\} \quad \text{for all} \quad \sigma_j \in \Sigma
\]

is the group of edges labeled by the identity input element \( u_0 \).

**Example 2:** The extended trellis section \( T_E \) of a minimal linear encoder \( E \) for the code of Example 1 is shown below in tabular form.

<table>
<thead>
<tr>
<th>( \sigma )</th>
<th>( u )</th>
<th>( y )</th>
<th>( \sigma' )</th>
<th>( \sigma )</th>
<th>( u )</th>
<th>( y )</th>
<th>( \sigma' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>000</td>
<td>00</td>
<td>10</td>
<td>01</td>
<td>101</td>
<td>00</td>
</tr>
<tr>
<td>00</td>
<td>01</td>
<td>111</td>
<td>10</td>
<td>10</td>
<td>00</td>
<td>010</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>11</td>
<td>100</td>
<td>01</td>
<td>10</td>
<td>10</td>
<td>001</td>
<td>01</td>
</tr>
<tr>
<td>00</td>
<td>10</td>
<td>011</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>110</td>
<td>11</td>
</tr>
</tbody>
</table>

Lemma 1: Given \( C \), there exists

\[
\prod_{i=0}^{k-1} (2^k - 2^i) \cdot (2^{(k+\nu)})
\]

distinct minimal linear encoders for it.

**Proof:** The first factor, which is the number of automorphisms of \( Z_2^k \), comes from all possible choices for \( e(\sigma_0, u) \), where \( u_1 \) is one of the \( k \) generators of \( U \). The second factor comes from all possible choices for \( e(\sigma_1, u_0) \), where \( \sigma_1 \) is one of the \( \nu \) generators of \( \Sigma \). For a generic \( \sigma \in \Sigma \) and \( u \in U \), \( e(\sigma, u) \) is then uniquely determined by the group properties of \( T_E \).

If \( E \) is linear, \( C \) is completely characterized by \( k \) pairs \((u^{(i)}, g^{(i)})\), where \( g^{(i)} \) are the generators of \( C \). In fact, \( \forall c \in C \) we recall (1)

\[
c = \sum_{i=1}^k \sum_{j \in S_i} D^i g^{(i)}.
\]

Since \( E \) is linear, the sequence \( u \in U^2 \) such that \((u, c) \in C\) is

\[
u = \sum_{i=1}^k \sum_{j \in S_i} D^i u^{(i)}.
\]

IV. Minimal linear systematic encoders

A minimal encoder is said to be systematic if

\[
\forall (\sigma, u, y, \sigma') \in T_E, \quad u = p_k(y), \quad \text{where} \quad p_k(y) \quad \text{is the projection onto the first} \quad k \quad \text{components of} \quad y.
\]

Owing to the group properties, a necessary condition for a group code \( C \) to admit a systematic encoder is that \( p_k(U) = Z_2^k \). This condition is not a serious limitation: if a code \( C' \) does not satisfy it, there always exists a permutation \( \rho \) of \( Z_2^k \) (that does not alter the code properties) such that the code \( C = \rho(C') \) satisfies the condition. In the following, we will assume that \( C \) satisfies this condition.

**Lemma 2:** Given \( C \), there exists one and only one minimal linear systematic encoder \( E \) for it.

**Proof:** Owing to the group properties of \( T \), for every \( \sigma \in \Sigma \) there is only one admissible choice for \( e(\sigma, u) \). In fact, there is only one edge \((\sigma, y, \sigma') \in T \) such that \( p_k(y) = u \). The resulting \( T_E \) is clearly a group.

The one-to-one correspondence between a code \( C \) and the extended trellis section \( T_E \) of its minimal linear systematic...
encoder will be used in the following for code/encoder search. The description of an encoder by its extended trellis section $T_E$ is sufficient for implementation: there exist algorithms (also available within many digital synthesis commercial programs) that translate finite directed labeled graphs into hardware description. However, mainly for historical reasons, convolutional encoders are more often described as algorithms (also available within many digital synthesis commercial programs) that translate finite directed labeled graphs into hardware description. However, mainly for historical reasons, convolutional encoders are more often described as finite state machines.

The description of an encoder by its extended trellis section $T_E$ is sufficient for implementation: there exist algorithms (also available within many digital synthesis commercial programs) that translate finite directed labeled graphs into hardware description. However, mainly for historical reasons, convolutional encoders are more often described as algorithms (also available within many digital synthesis commercial programs) that translate finite directed labeled graphs into hardware description. However, mainly for historical reasons, convolutional encoders are more often described as finite state machines.

Theorem 2: Given a code $C$ with $k$ generators

$$g^{(i)} = (g_0^{(i)}, \ldots, g_{l}^{(i)}) = (u_0^{(i)} | \ldots | u_{k}^{(i)} | \ldots | r_{n-k}^{(i)}),$$

if we choose $f$ such that

$$f^{-1}(u_0^{(i)}, \ldots, f^{-1}(u_{k} \ldots, 0, 1)) = v_0^{(k)},$$

and $f_{ij}$ such that

$$f^{-1}(f_{ij}) = u_0^{(i)} \ldots, f^{-1}(f_{ik}) = u_0^{(k)},$$

then the structure of Fig. 1 realizes the minimal linear systematic encoder for $C$.

Proof: The encoder is minimal, because the cardinality of its state space depends on the generator lengths. It is clearly systematic and linear by construction. It only remains to prove that every generator $g^{(i)}$ is generated exactly by $u^{(i)} = p_k(g^{(i)})$. A sequence $u^{(i)}$ produces a sequence $y^{(i)}$, where the first symbol has a 1 on the $i$-th line and 0's on the others, while all other symbols are 0's over all lines. As a consequence, the generated sequence $y'$ is precisely $(y_0^{(i)}, \ldots, y_{n-k}^{(i)}).$ □

A. Systematic recursive encoders

As previously outlined, the component convolutional codes in a turbo code must be recursive. An encoder is recursive if, given any sequence $u \in \mathbb{Z}^2$ with Hamming weight $w_H(u) = 1$, no finite-weight sequences $c \in C$ exist, such that $(u, c) \in C$.

A sufficient condition to ensure that the canonical encoder of Fig. 1 is recursive is that at least one of the coefficients $f_{ij}$, $j = 1, \ldots, k$, is different from zero $\forall i = 1, \ldots, k$.

V. APPLICATION TO TURBO-CODES

In [5] we have proved that the performance of a PCCC depends mainly on the minimum-weight code sequences of the CCs generated by input sequences of weight 2. According to [5], we define as effective free distance of the CC, $d_{f, eff}$, this minimum weight. The canonical form of rate $k/n$ recursive encoders previously introduced has been used for an exhaustive search of "optimum" CCs.

The search program is based on the following steps:

1. The parameters $k, n$ defining the code rate, and the set of $l_i, i = 1, \ldots, k$ defining the code memory $\nu = \sum_{i=1}^k l_i$ are chosen.

2. Defining as $d_i$ the minimum weight of code sequences generated by input sequences of weight $i$ (notice that $d_2 = d_{f, eff}$), the algorithm searches the codes that maximize sequentially $d_i$, $i = 1, 2, 3, 4$. Among those yielding the largest values for $d_1, \ldots, d_4$, it retains those with the largest $d_{f, eff}$.

3. In the search performed in [6], the authors only maximized $d_1, d_2, d_3$.

4. Several properties, based on group code theory and symmetry/structural considerations, can be incorporated into the search algorithm, yielding a significant reduction of the number of codes to be generated. This makes the $T_E$ approach appealing with respect to other convolutional codes representation, especially for high rates and number of states.

To characterize the codes obtained in the search, we used two sets of binary vectors that resemble the generating vec-

\[1\text{Notice that the definition of } d_i \text{ in [6] is different, in the sense that it represents only the minimum Hamming weight of the redundant bits.}\]
The 8-state rate 2/3 encoder of Table 4 in its canonical (A) and implementation-oriented (B) versions.

The vectors $h_{i,j}^m, i = 1, \ldots, k, j = 1, \ldots, l_i + 1$ and $z_{mt}^m, m = 1, \ldots, k, t = 1, \ldots, (n - k)$, expressed in octal notation (least significant bit on the right corresponding to the boldface multiplier in Fig. 2) refer to the feedback and feedforward connections respectively. As an example, in Fig. 2 we draw the encoder according to the vectors characterization, and the corresponding canonical structure.

In Tables 1, 2, 3, 4, 5, 6, we report the best codes found for rates 1/2, 1/3, 1/4, 2/3, 2/4 and 3/4. In terms of $d_{f,eff}$, the results agree with those of [6]. However, since our optimization takes into account more weights, it leads sometimes to improved codes for higher input weights.

The upper bound to the bit error probability described in [8] has been applied to rate 1/2 PCCCs that employ as CCs the encoders of Table 4 (for 4,8,16,32 states) and an interleaver of length 1000. The results are shown in Fig. 3. We also report the rate 1/2 PCCCs obtained from the best punctured rate 1/2 CCs with 8 and 16 states. The curves show a significant advantage of rate 2/3 CCs with respect to the punctured rate 1/2.

VI. CONCLUSIONS

We have presented here a canonical structure of minimal linear systematic rate $k/n$ encoders and used it in the search for good constituent codes of parallel concatenated codes. The results obtained through an upper bounding technique show that higher rate constituent codes yield better performance to the concatenated code than punctured rate 1/2 constituent codes. This makes it worthwhile searching for optimum high rate systematic recursive convolutional encoders.

REFERENCES

Table 1: Best systematic encoders with rate \( R = 1/2 \) (The boldface number denotes the \( d_{free} \)).

<table>
<thead>
<tr>
<th>( N )</th>
<th>Part.</th>
<th>( z_{(1)}^{(1)} )</th>
<th>( h_{(1)}^{(1)} )</th>
<th>( d_{eff} )</th>
<th>( d_3 )</th>
<th>( d_4 )</th>
<th>( d_5 )</th>
<th>( d_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>13</td>
<td>13</td>
<td>14</td>
<td>14</td>
<td>10</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>23</td>
<td>23</td>
<td>22</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>67</td>
<td>67</td>
<td>38</td>
<td>17</td>
<td>16</td>
<td>11</td>
<td>14</td>
</tr>
<tr>
<td>64</td>
<td>6</td>
<td>147</td>
<td>147</td>
<td>70</td>
<td>23</td>
<td>14</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 2: Best systematic encoders with rate \( R = 1/3 \).

<table>
<thead>
<tr>
<th>( N )</th>
<th>Part.</th>
<th>( z_{(1)}^{(1)} )</th>
<th>( h_{(1)}^{(1)} )</th>
<th>( d_{eff} )</th>
<th>( d_3 )</th>
<th>( d_4 )</th>
<th>( d_5 )</th>
<th>( d_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>2.3</td>
<td>3</td>
<td>6</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>7.5</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>14</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>17,15</td>
<td>13</td>
<td>20</td>
<td>12</td>
<td>14</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>37,27</td>
<td>23</td>
<td>32</td>
<td>15</td>
<td>14</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>51,45</td>
<td>67</td>
<td>56</td>
<td>23</td>
<td>20</td>
<td>15</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 3: Best systematic encoders with rate \( R = 1/4 \).

<table>
<thead>
<tr>
<th>( N )</th>
<th>Part.</th>
<th>Output</th>
<th>Feedback</th>
<th>( h_{(1)}^{(1)} )</th>
<th>( h_{(2)}^{(1)} )</th>
<th>( d_{eff} )</th>
<th>( d_3 )</th>
<th>( d_4 )</th>
<th>( d_5 )</th>
<th>( d_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1,0</td>
<td>1,0</td>
<td>3,0,1,0</td>
<td>3,0,1,0</td>
<td>2</td>
<td>( \infty )</td>
<td>6</td>
<td>( \infty )</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1,1</td>
<td>3,3</td>
<td>2,3,3,3</td>
<td>2,3,3,3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2,1</td>
<td>7,2</td>
<td>5,4,3,1</td>
<td>5,4,3,1</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>2,2</td>
<td>3,6</td>
<td>1,7,5,0</td>
<td>8</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>3,2</td>
<td>11,7</td>
<td>6,11,7,0</td>
<td>12</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>3,3</td>
<td>1,11</td>
<td>13,12,16,1</td>
<td>20</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Best systematic encoders with rate \( R = 2/3 \).

<table>
<thead>
<tr>
<th>( N )</th>
<th>Part.</th>
<th>Output</th>
<th>Feedback</th>
<th>( h_{(1)}^{(1)} )</th>
<th>( h_{(2)}^{(1)} )</th>
<th>( d_{eff} )</th>
<th>( d_3 )</th>
<th>( d_4 )</th>
<th>( d_5 )</th>
<th>( d_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1,0</td>
<td>2,1,1</td>
<td>3,1,1,0</td>
<td>3,1,1,0</td>
<td>4</td>
<td>( \infty )</td>
<td>6</td>
<td>( \infty )</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1,1</td>
<td>3,3,3</td>
<td>2,3,3,3</td>
<td>2,3,3,3</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1,2</td>
<td>4,4,0</td>
<td>1,3,7,0</td>
<td>9</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>2,2</td>
<td>3,6,6</td>
<td>1,7,5,0</td>
<td>14</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Best systematic encoders with rate \( R = 2/4 \).

<table>
<thead>
<tr>
<th>( N )</th>
<th>Part.</th>
<th>Output</th>
<th>Feedback</th>
<th>( h_{(1)}^{(1)} )</th>
<th>( h_{(2)}^{(1)} )</th>
<th>( d_{eff} )</th>
<th>( d_3 )</th>
<th>( d_4 )</th>
<th>( d_5 )</th>
<th>( d_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1,0,0</td>
<td>1,0</td>
<td>3,0,1,0</td>
<td>3,0,1,0</td>
<td>2</td>
<td>( \infty )</td>
<td>6</td>
<td>( \infty )</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1,1,0</td>
<td>1,1</td>
<td>1,3,0,1</td>
<td>1,3,0,1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1,1,1</td>
<td>2,1</td>
<td>0,2,3,0</td>
<td>3,0,0</td>
<td>4</td>
<td>( \infty )</td>
<td>4</td>
<td>( \infty )</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Best systematic encoders with rate \( R = 3/4 \).