A Puncturing Algorithm for Rate-Compatible LDPC Convolutional Codes

Hua Zhou*, David G. M. Mitchell†, Norbert Goertz*, and Daniel J. Costello, Jr.†
*Institute of Telecommunications, Vienna University of Technology, Vienna, Austria,
{hua.zhou, norbert.goertz}@nt.tuwien.ac.at
†Dept. of Electrical Engineering, University of Notre Dame, Notre Dame, Indiana, USA,
{david.mitchell, costello.2}@nd.edu

***Abstract*—A family of rate-compatible (RC) punctured low-density parity-check convolutional codes (LDPC-CCs) is derived from an LDPC-CC by periodically puncturing encoded bits (variable nodes) with respect to several criteria: (1) ensuring the recoverability of punctured variable nodes, (2) minimizing the number of completely punctured cycle trapping sets (CPCTSs), and (3) minimizing the number of punctured variable nodes involved in short cycles. As an example, a family of RC punctured LDPC-CCs with rates $4/9$, $4/8$, $4/7$, $4/6$, and $4/5$ are obtained from the $(21, 3, 5)$ Tanner LDPC-CC.**

I. Introduction

To adapt to the changing conditions of time-varying channels, rate-compatible (RC) channel codes, a form of variable rate coding, allow transceivers to employ the same encoder/decoder pair. As competitors to RC turbo codes, RC low-density parity-check block codes (LDPC-BCs) have recently been investigated using code modifying techniques such as nulling, extending, and puncturing [1]-[4].

By nulling information bits and puncturing parity check bits, Tian *et al*. [1] obtained RC LDPC-BCs with the same degree distribution as the original code. A combination of puncturing and extending was used in [2] and [3] to overcome the degradation in decoding performance caused when a large number of bits are punctured in order to obtain high rate LDPC-BCs. Ha *et al*. [4] then proposed a puncturing scheme to obtain RC LDPC-BCs based on puncturing nodes that require fewer decoding iterations to be recovered, referred to as the *recoverability* of punctured variable nodes criterion in this paper.

As counterparts to RC block codes, RC punctured convolutional codes were first introduced in [5] by periodically puncturing encoded bits chosen with respect to the distance spectra of the codes. Costello *et al*. [6] have shown that practical LDPC convolutional codes (LDPC-CCs) can be implemented without an increase in computational complexity compared to corresponding LDPC-BCs; therefore, extending the concept of conventional RC convolutional codes to RC punctured LDPC-CCs is of practical interest. A method to estimate the distance spectrum of LDPC-CCs has been proposed in [7]. However, due to the high order of the syndrome former memory of practically interesting LDPC-CCs, it is infeasible to precisely compute the distance spectra of punctured LDPC-CCs. Moreover, when using (sub-optimal) iterative decoding techniques, there are many other important code characteristics that affect decoding performance, such as short cycles in the Tanner graph and trapping sets [8].

Recently, Skoglund *et al*. [9] constructed a family of RC LDPC-CCs for Type-II HARQ systems by successively adding check nodes to the graph of a high-rate mother code to obtain lower rate codes. By contrast, in this paper we propose a method of obtaining a family of RC punctured LDPC-CCs by periodically puncturing encoded bits with respect to the following criteria: (1) ensuring the recoverability of punctured variable nodes, (2) minimizing the number of *completely punctured cycle trapping sets* (CPCTSs), and (3) minimizing the number of punctured variable nodes involved in short cycles, and then analyzing several properties of the resulting codes. One of the advantages of this method compared to designing a puncturing scheme based on the distance spectrum is that these properties can be precisely and efficiently calculated. As proposed in [5], compatible rates are realized by incrementally modifying the puncturing pattern $\mathbf{a}_i$ of the previous lower rate punctured code. As an example, a family of RC punctured LDPC-CCs with rates $4/9$, $4/8$, $4/7$, $4/6$, and $4/5$ are obtained from the $(21, 3, 5)$ Tanner LDPC-CC [10].

This work was partly supported by NSF Grants CCF-0830650 and CCF-1161754.

II. LDPC convolutional codes

A rate $R = b/c$ LDPC-CC [11] can be described as the set of binary sequences $\mathbf{v} = (\mathbf{v}_0,\mathbf{v}_1,\ldots,\mathbf{v}_t,\ldots)$, $\mathbf{v}_t = (v_t^{(1)},v_t^{(2)},\ldots,v_t^{(c)})$, satisfying $\mathbf{v}\cdot\mathbf{H}^T = \mathbf{0}$, where

$$\mathbf{H}^T=\begin{bmatrix} \mathbf{H}_0^T(0) & \mathbf{H}_1^T(1) & \cdots & \mathbf{H}_{m_s}^T(m_s) & & & \\ & \mathbf{H}_0^T(1) & \cdots & \mathbf{H}_{m_s-1}^T(m_s) & \mathbf{H}_{m_s}^T(m_s+1) & & \\ & & \ddots & \vdots & \vdots & \ddots & \\ & & & \mathbf{H}_0^T(m_s) & \mathbf{H}_1^T(m_s+1) & \cdots & \mathbf{H}_{m_s}^T(m_s+m_s) \\ & & & & \ddots & & \ddots \end{bmatrix}, \tag{1}$$

and blank spaces correspond to zeros. The transposed parity check matrix $\mathbf{H}^T$, called the syndrome former matrix, is made up of a set of binary submatrices $\mathbf{H}_i^T(t)$, $t\in\mathbb{Z}^*$, $i = 0, 1, \ldots, m_s$, each of size $c\times q$, $q \triangleq c - b$. The value m_s is called the syndrome former memory. If, beginning with the block of q columns at time $t = m_s$, $\mathbf{H}^T$ contains exactly J ones in each row and K ones in each column, then the code is called an (m_s, J, K)-regular LDPC-CC. In the Tanner graph representation of $\mathbf{H}^T$, a variable node, denoted as $v_t^{(j)}$, $j = 1, 2, \ldots, c$, corresponds to row $tc+j$ of (1), while a check node, denoted as $c_t^{(k)}$, $k = 1, 2, \ldots, q$, corresponds to column $tq+k$ of (1). In this paper, we focus on *time-invariant* LDPC-CCs. Here, the binary submatrices in $\mathbf{H}^T$ are constant on each diagonal, i.e., $\mathbf{H}_i^T(t) = \mathbf{H}_i^T$, $\forall i, t$.

Given the time-domain syndrome former matrix $\mathbf{H}^T$ of a time-invariant LDPC-CC in (1) with rate $R = b/c$ and syndrome former memory m_s, the polynomial-domain syndrome former matrix is given by $\mathbf{H}^T(D) = \sum_{n=0}^{m_s}\mathbf{H}_n^T\cdot D^n$, where the codeword polynomial $\mathbf{V}(D)$ satisfies $\mathbf{V}(D)\mathbf{H}^T(D)=\mathbf{0}(D)$, $\mathbf{V}(D) = [\mathbf{v}^{(0)}(D), \mathbf{v}^{(1)}(D), \ldots, \mathbf{v}^{(c-1)}(D)]$, $\mathbf{v}^{(j)}(D) = v_0^{(j)} + v_1^{(j)}D+\cdots+v_t^{(j)}D^t+\cdots$. Details of the relationship between $\mathbf{H}^T$ and $\mathbf{H}^T(D)$ are described in [12].

III. Cycle enumerators and cycle trapping sets

In this section, we introduce cycle enumerators and so-called cycle trapping sets for LDPC-CCs and review the con-

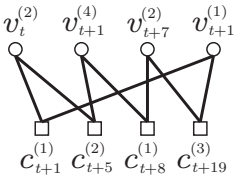


Fig. 1: A cycle $r_8(t)$ in the $(21, 3, 5)$ Tanner LDPC-CC.

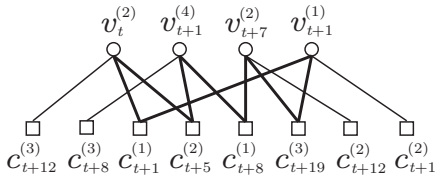


Fig. 2: The (4,4) cycle trapping set derived from $r_8(t)$ in Fig. 1.

cept of recoverability of punctured variable nodes presented in [4]. These three properties form the basis of the design criteria used to obtain the RC punctured LDPC-CCs introduced in Section IV.

A. Cycle enumerators of LDPC-CCs

Definition 1. *A cycle in the Tanner graph of an LDPC-CC is defined as a finite alternating sequence of nodes (variable nodes or check nodes) connected by edges, beginning and ending with the same node, such that no node appears more than once. The length of the shortest cycle is called the girth of an LDPC-CC and is denoted by g.*

A cycle $r_w(t)$ of length w, associated with time t, is denoted by

$$r_w(t) = \{v_{t_1+t}^{(j_1)}, c_{t_2+t}^{(k_1)}, \ldots, v_{t_{w-1}+t}^{(j_{w/2})}, c_{t_w+t}^{(k_{w/2})}\}, \tag{2}$$

where $j_i \in \{1, 2, \ldots, c\}$, $k_i \in \{1, 2, \ldots, q\}$, $i = 1, 2 \ldots, w/2$, $w \in \{4, 6, 8, \ldots\}$, $t_s \in \mathbb{Z}^*$, $s = 1, 2, \ldots, w$, and $t \in \mathbb{Z}$, such that $t_s + t \in \mathbb{Z}^*$, $\forall s$. The cycle $r_w(t)$ contains $w/2$ variable nodes $\{v_{t+t_1}^{(j_1)}, v_{t+t_3}^{(j_2)}, \ldots, v_{t+t_{w-1}}^{(j_{w/2})}\}$ and $w/2$ check nodes $\{c_{t+t_2}^{(k_1)}, c_{t+t_4}^{(k_2)}, \ldots, c_{t+t_w}^{(k_{w/2})}\}$. Since the time-domain syndrome former matrix is semi-infinite, an LDPC-CC has an infinite number of cycles. However, due to the periodically repeating identical structure of the Tanner graph, given a particular cycle $r_w(t)$, the set of periodically shifted cycles $\{r_w(t)\}$ can be considered as one *type*.

Definition 2. *The cycle enumerator for an LDPC-CC is denoted as R_w, where R_w indicates the number of types of cycles of length w, i.e., periodically shifted cycles are only counted once.*

To obtain the cycle enumerators of an LDPC-CC, we use the cycle counting method introduced in [13]. For example, applying this method to the $(21, 3, 5)$ Tanner LDPC-CC with polynomial syndrome former matrix[1]

$$\mathbf{H}^T(D) = \begin{bmatrix} 1 & 1 & D^{18} \\ D & D^5 & D^{12} \\ D^3 & D^{15} & 1 \\ D^7 & D^4 & D^7 \\ D^{15} & D^{13} & D^{21} \end{bmatrix}, \tag{3}$$

the first three non-zero cycle enumerators are given by $R_8 = 11$, $R_{10} = 62$, and $R_{12} = 351$, i.e., the number of types of cycles of length 8, 10, and 12 are 11, 62, and 351, respectively. Consequently, the girth of this code is 8. One of the 11 cycles in R_8 is shown in Fig. 1. The cycle $r_8(t) = \{v_t^{(2)}, c_{t+1}^{(1)}, v_{t+1}^{(1)}, c_{t+19}^{(3)}, v_{t+7}^{(2)}, c_{t+8}^{(1)}, v_{t+1}^{(4)}, c_{t+5}^{(2)}\}$, $t \in \mathbb{Z}^*$, consists of a set of variable nodes $\{v_t^{(2)}, v_{t+1}^{(1)}, v_{t+1}^{(4)}, v_{t+7}^{(2)}\}$ and a set of check nodes $\{c_{t+1}^{(1)}, c_{t+5}^{(2)}, c_{t+8}^{(1)}, c_{t+19}^{(3)}\}$. In the remainder of the paper, we simply refer to R_w as the number of cycles of length w rather than the number of types of cycles of length w.

[1]Note that the common factors of D have been removed from the code in [10] for simplicity.

B. Cycle trapping sets of LDPC-CCs

Definition 3. *A (d, f) trapping set $\tau_{d,f}$ of a Tanner graph is a set of variable nodes of size d which induces a subgraph with exactly f odd-degree check nodes (and an arbitrary number of even-degree check nodes).*[2]

Definition 4. *If the d variable nodes involved in a cycle of length $w = 2d$ induce a subgraph with $f > 0$ odd-degree check nodes, then the $(w/2, f)$ trapping set defined by the subgraph is called a $(w/2, f)$ cycle trapping set.*

Trapping sets derived from short cycles, or the union of short cycles, have been shown to be the most dominant ones in terms of decoding performance in the high signal-to-noise ratio (SNR) region on the binary symmetric channel [14].

Consider the cycle $r_8(t)$ shown in Fig. 1 as an example. By attaching the remaining neighboring check nodes in the original graph to each of the variable nodes, we obtain an induced subgraph (shown in Fig. 2) that defines a $(4, 4)$ cycle trapping set. The degree-one check node set $\{c_{t+1}^{(2)}, c_{t+12}^{(2)}, c_{t+8}^{(3)}, c_{t+12}^{(3)}\}$ corresponds to the $f = 4$ odd-degree check nodes in the trapping set and the set $\{c_{t+19}^{(3)}, c_{t+8}^{(1)}, c_{t+5}^{(2)}, c_{t+1}^{(1)}\}$ are the $w/2 = d = 4$ even-degree check nodes. Thus each cycle in the Tanner graph corresponds to a unique cycle trapping set. Consequently, there are 11, 62, and 351 cycle trapping sets in the code defined by (3), containing exactly 4, 5, and 6 variable nodes, respectively.

C. m-step-recoverable (m-SR) punctured nodes

We define $\mathcal{N}(v_t^{(j)})$ as the neighboring check node set of the variable node $v_t^{(j)}$, and $\mathcal{N}(v_t^{(j)}) \setminus c_{t'}^{(k)}$ as the neighboring check node set excluding check node $c_{t'}^{(k)}$, respectively. Similar definitions apply to $\mathcal{N}(c_t^{(k)})$ and $\mathcal{N}(c_t^{(k)}) \setminus v_{t'}^{(j)}$.

A punctured variable node $v_t^{(j)}$ recovered in the mth iteration by the iterative decoding algorithm is called an m-step-recoverable (m-SR) node [4] if $v_t^{(j)}$ has at least one neighboring check node $c_{t'}^{(k)}$ in $\mathcal{N}(v_t^{(j)})$ such that the set $\mathcal{N}(c_{t'}^{(k)}) \setminus v_t^{(j)}$ contains at least one (m-1)-SR node and the others are n-SR nodes, where $0 \leq n \leq m-1$ (an unpunctured variable node is defined as a 0-SR node). If the punctured variable node cannot be recovered in a finite number of steps, it is called *unrecoverable* and is described as an ∞-SR node. This represents a punctured variable node whose recovery tree has no reliable check nodes. A check node $c_{t'}^{(k)}$ in $\mathcal{N}(v_t^{(j)})$ is called *reliable* with respect to $v_t^{(j)}$ if all the variable nodes in $\mathcal{N}(c_{t'}^{(k)}) \setminus v_t^{(j)}$ are unpunctured nodes. Otherwise, it is called an *unreliable* check node. Fig. 3 illustrates an example of the recovery tree of a punctured 2-SR variable node. It has two unreliable check nodes, and each of them is connected to one 1-SR variable node. The squares filled with u and r are unreliable and reliable check nodes, respectively. Filled circles are punctured variable nodes, while the unfilled ones are unpunctured variable nodes. Further details of the recovery tree and m-SR nodes can be found in [4].

[2]We define a trapping set as a topology of the bipartite graph, i.e., it does not depend on a particular decoder (see [8]).

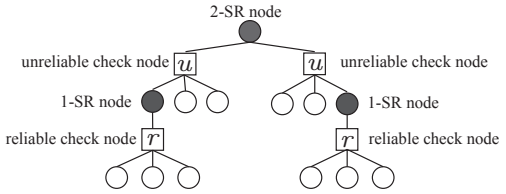


Fig. 3: The recovery tree of a 2-SR punctured variable node.

IV. RC PUNCTURED LDPC-CCS

In this section, we present a method to construct RC punctured LDPC-CCs by analyzing several properties of punctured LDPC-CCs. The advantage of this method over a distance spectrum based technique is that all these properties can be precisely and efficiently calculated.

A. Puncturing patterns of LDPC-CCs

Given a polynomial-domain syndrome former matrix of size $c \times q$, a puncturing pattern with period P is defined by the $P \times c$ matrix

$$\mathbf{a} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,c-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,c-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{P-1,0} & a_{P-1,1} & \cdots & a_{P-1,c-1} \end{bmatrix}, \quad (4)$$

where $a_{x,y} \in \{0,1\}$, $x = 0,1,\ldots,P-1$, $y = 0,1,\ldots,c-1$.[3] Each row of the puncturing pattern corresponds to the c encoded bits (variable nodes) at one time unit. The ones and zeros in (4) imply puncturing and transmission of the associated bits, respectively. If l is the total number of ones in the puncturing pattern, the resulting punctured code has rate $R' = (P \times b)/(P \times c - l)$, where $b = c - q$, $l \leq P \times q$, and $R' < 1$.

B. Criteria used to design RC punctured LDPC-CCs

We now introduce the design criteria used to obtain RC punctured LDPC-CCs from a "mother code" (original unpunctured code). The compatible rates are obtained by a set of nested puncturing patterns, i.e., a higher rate punctured code is obtained from the previous lower rate code by including an extra punctured entry in the puncturing pattern of the previous code. This results in a RC family of codes derived from the same mother code.

Given a total number of punctured entries l in $\mathbf{a}$ and a puncturing period P, there are $\binom{Pc}{l}$ possible puncturing patterns that result in the same punctured code rate. However, their decoding performance varies. The best puncturing pattern will be chosen based on the following three criteria.

1) Ensuring the recoverability of punctured variable nodes: A puncturing pattern $\mathbf{a}_i$ that generates unrecoverable punctured variable nodes, i.e., ∞-SR nodes, results in poor decoding performance, since ∞-SR nodes are unable to offer reliable information to help recover other punctured nodes or to help correct unpunctured nodes received in error. We will demonstrate the dramatic degradation in decoding performance caused by ∞-SR nodes in Section V.

In addition to the extreme case of an ∞-SR node, we also wish to avoid m-SR nodes with large m. The recoverability of an m-SR node is determined by the value of m. The smaller the value of m, the easier it is to recover a punctured variable node.

Definition 5. *We define the element $C_m^{\mathbf{a}_i}$ in the vector $\boldsymbol{E}_c^{\mathbf{a}_i} = \begin{bmatrix} C_1^{\mathbf{a}_i} & C_2^{\mathbf{a}_i} & \cdots & C_{N_c}^{\mathbf{a}_i} \end{bmatrix}$, where $1 \leq m \leq N_c$ and $m, N_c \in \mathbb{Z}^+$, as the enumerator of m-SR nodes in the punctured code within one period of puncturing pattern $\mathbf{a}_i$.*[4]

Given a puncturing pattern $\mathbf{a}_i$ and the number of punctured entries l, the number of elements in the vector $\boldsymbol{E}_c^{\mathbf{a}_i}$ is

$$N_c = \min \left\{ N_c' \left| \sum_{m=1}^{N_c'} C_m^{\mathbf{a}_i} + C_\infty^{\mathbf{a}_i} = l, N_c' \in \mathbb{Z}^+ \right. \right\}. \quad (5)$$

Note that sum of the elements in $\boldsymbol{E}_c^{\mathbf{a}_i}$ and $C_\infty^{\mathbf{a}_i}$ equals l.

A puncturing pattern $\mathbf{a}_{i_1}$ is said to be superior to puncturing pattern $\mathbf{a}_{i_2}$ with respect to the recoverability of punctured variable nodes if there exists an $M \in \mathbb{Z}^*$ such that

$$\begin{cases} C_m^{\mathbf{a}_{i_1}} = C_m^{\mathbf{a}_{i_2}}, & 1 \leq m \leq M-1 \\ C_m^{\mathbf{a}_{i_1}} > C_m^{\mathbf{a}_{i_2}}, & m = M \end{cases}. \quad (6)$$

In other words, puncturing pattern $\mathbf{a}_{i_1}$ is superior to pattern $\mathbf{a}_{i_2}$ if the enumerator vector of $\mathbf{a}_{i_1}$ is more weighted toward smaller values of m than that of $\mathbf{a}_{i_2}$.

2) Minimize the number of CPCTSs: If all of the variable nodes in a cycle trapping set are punctured, then, because of the uncertainty of the nodes and the limited connectivity available to obtain helpful information from nodes outside the set, the probability that the associated symbols are decoded correctly decreases.

Definition 6. *Given a puncturing pattern $\mathbf{a}_i$, if a cycle $r_w(t)$ has all of its $w/2$ variable nodes punctured, it corresponds to a CPCTS $\tau_{w/2,f}$, $f \in \mathbb{Z}^*$. We define the element $\mathcal{T}_{w/2}^{\mathbf{a}_i}$ in the vector $\boldsymbol{E}_\tau^{\mathbf{a}_i} = \begin{bmatrix} \mathcal{T}_{g/2}^{\mathbf{a}_i} & \mathcal{T}_{(g+2)/2}^{\mathbf{a}_i} & \cdots & \mathcal{T}_{N_\tau/2}^{\mathbf{a}_i} \end{bmatrix}$, where $w = g, g+2, g+4, \ldots, N_\tau$ and $g/2$, $N_\tau/2 \in \mathbb{Z}^+$, as the enumerator of the number of CPCTSs $\tau_{w/2,f}$, $\forall f$, corresponding to the puncturing pattern $\mathbf{a}_i$.*

The number of CPCTSs associated with a particular puncturing pattern is obtained by checking each cycle $r_w(t)$ to see if it has all its variable nodes punctured. In Algorithm 1, the sum of the elements of $\boldsymbol{E}_\tau^{\mathbf{a}_i}$ is used to compare different puncturing patterns.

3) Minimize the number of punctured variable nodes involved in short cycles: Short cycles in LDPC-CCs are known to prevent the suboptimal iterative sum-product decoding algorithm from converging [15]. Intuitively, puncturing variable nodes in short cycles makes the situation even worse, since unreliable information is passed around a short loop. Consequently, we seek puncturing patterns that have few punctured nodes involved in short cycles.

Definition 7. *We define the element $B_w^{\mathbf{a}_i}$ in the vector $\boldsymbol{E}_b^{\mathbf{a}_i} = \begin{bmatrix} B_g^{\mathbf{a}_i} & B_{g+2}^{\mathbf{a}_i} & \cdots & B_{N_b}^{\mathbf{a}_i} \end{bmatrix}$, where $w = g, g+2, g+4, \ldots, N_b$ and g, $N_b \in \mathbb{Z}^+$, as the enumerator of the total number of punctured variable nodes involved in all cycles of length w for the puncturing pattern $\mathbf{a}_i$.*

Given a cycle $r_w(t)$ and a puncturing pattern $\mathbf{a}_i$ with period P, $B_w^{\mathbf{a}_i}$ is obtained by calculating the number of punctured variable nodes in each of the cycles in the set $\{r_w(t), r_w(t+1), \ldots, r_w(t+P-1)\}$. For example, consider the cycle $r_8(t)$ in Fig. 1. Given the puncturing pattern $\mathbf{a}_0 = [10000; 00010]$ with $P = 2$, the set of punctured variable nodes is $\{v_{t+nP}^{(1)}, v_{t+1+nP}^{(4)} | n = 0, 1\}$. Therefore, in the sets of variable nodes $\{v_t^{(2)}, v_{t+1}^{(1)}, v_{t+7}^{(2)}, v_{t+1}^{(4)}\}$ and $\{v_{t+1}^{(2)}, v_{t+2}^{(1)}, v_{t+8}^{(2)}, v_{t+2}^{(4)}\}$ belonging to cycles $r_8(t)$ and $r_8(t+1)$, variable nodes $v_{t+1}^{(4)}$ and $v_{t+2}^{(1)}$ are punctured when $n = 0$ and $n = 1$, respectively. Applying this concept to the 11 cycles

[3] In the remainder of the paper, the puncturing pattern in (4) is presented as a vector with rows separated by ";".

[4] The enumerator vector $\boldsymbol{E}_c^{\mathbf{a}_i}$ does not include the enumerator $C_\infty^{\mathbf{a}_i}$ of unrecoverable punctured variable nodes.

of length 8 for this code, there are 8 and 12 punctured variable nodes in the cycles $r_8(t)$ and $r_8(t+1)$, respectively. Therefore, the total number of punctured variable nodes involved in cycles of length 8 is $B_8^{\mathbf{a}_0} = 8+12 = 20$. When comparing puncturing patterns, we will compare the cumulative number of punctured nodes involved in short cycles, i.e., $\sum_w B_w^{\mathbf{a}_i}$ for small w.

C. Designing RC punctured LDPC-CCs

Based on the three criteria introduced in Section IV-B, we propose to search for puncturing patterns $\mathbf{a}_i$ with no ∞-SR nodes and the smallest possible entries for the enumerators $\boldsymbol{E}_c^{\mathbf{a}_i}$, $\boldsymbol{E}_\tau^{\mathbf{a}_i}$, and $\boldsymbol{E}_b^{\mathbf{a}_i}$. Given a mother LDPC-CC of rate $R = b/c$, a search procedure to obtain a rate $R' = (P \times b)/(P \times c - l)$ punctured LDPC-CC based on these three criteria is summarized in Algorithm 1.

Algorithm 1 Search for RC punctured LDPC-CCs

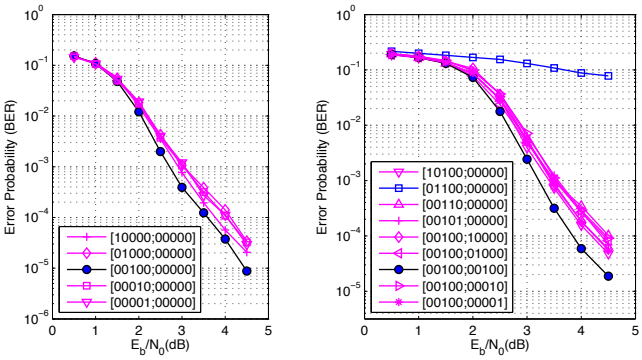
Initialization: Set the value of the desired puncturing period P and the number of punctured entries l such that $R' = (P \times b)/(P \times c - l)$. If a puncturing pattern $\mathbf{a}$ with $k < l$ non-zero entries has been selected at a previous stage, form $A_P^l = \{\mathbf{a}_i\}$, where A_P^l is the set of all non-equivalent puncturing patterns $\mathbf{a}_i$ that can be obtained from $\mathbf{a}$ by including $l - k$ additional punctured entries.[5] If no previous pattern has been selected, set $\mathbf{a} = \mathbf{0}$ and generate A_P^l as described above. Set the parameters for the girth g and the maximum cycle lengths N_τ and N_b that will be used to calculate enumerator vectors $\boldsymbol{E}_\tau^{\mathbf{a}_i}$ and $\boldsymbol{E}_b^{\mathbf{a}_i}$, respectively. For each puncturing pattern $\mathbf{a}_i \in A_P^l$, calculate the enumerators $C_\infty^{\mathbf{a}_i}$, $\boldsymbol{E}_c^{\mathbf{a}_i}$, $\boldsymbol{E}_\tau^{\mathbf{a}_i}$, and $\boldsymbol{E}_b^{\mathbf{a}_i}$ and store them. Initialize $\mathcal{A}_j = \varnothing$, where $j = 1, 2, 3, 4$.

- Step 1: *Ensure the recoverability of punctured variable nodes.*
  - Step 1.1: *Choose the puncturing patterns with the fewest ∞-SR nodes.* Calculate $\bar{C}_\infty = \min\{C_\infty^{\mathbf{a}_i} | \mathbf{a}_i \in A_P^l\}$. For each puncturing pattern $\mathbf{a}_i \in A_p^l$, if $C_\infty^{\mathbf{a}_i} = \bar{C}_\infty$, $\mathcal{A}_1 \leftarrow \mathcal{A}_1 \cup \mathbf{a}_i$.
  - Step 1.2: *Choose puncturing patterns with the best recoverability of punctured variable nodes.* Based on $\boldsymbol{E}_c^{\mathbf{a}_i}$, if a pattern $\mathbf{a}_i \in \mathcal{A}_1$ is superior to (ref. (6)) or has the same recoverability as any other patterns in $\mathcal{A}_1 \setminus \mathbf{a}_i$, $\mathcal{A}_2 \leftarrow \mathcal{A}_2 \cup \mathbf{a}_i$.
- Step 2: *Minimize the number of CPCTSs.* Calculate $\bar{\tau} = \min\{\sum_w \mathcal{T}_{w/2}^{\mathbf{a}_i} | \mathbf{a}_i \in \mathcal{A}_1\}$, where $w = g, g+2, \ldots, N_\tau$. For each puncturing pattern $\mathbf{a}_i \in \mathcal{A}_2$, if $\sum_w \mathcal{T}_{w/2}^{\mathbf{a}_i} > \bar{\tau}$, $\mathcal{A}_1 \leftarrow \mathcal{A}_1 \setminus \mathbf{a}_i$; else, $\mathcal{A}_3 \leftarrow \mathcal{A}_3 \cup \mathbf{a}_i$. If $\mathcal{A}_3 = \varnothing$, go to Step 1.2; else, go to Step 3.
- Step 3: *Minimize the number of punctured variable nodes involved in short cycles.* Calculate $\bar{B} = \min\{\sum_w B_w^{\mathbf{a}_i} | \mathbf{a}_i \in \mathcal{A}_3\}$, where $w = g, g+2, \ldots, N_b$. For each puncturing pattern $\mathbf{a}_i \in \mathcal{A}_3$, if $\sum_w B_w^{\mathbf{a}_i} = \bar{B}$, $\mathcal{A}_4 \leftarrow \mathcal{A}_4 \cup \mathbf{a}_i$. If $|\mathcal{A}_4| = 1$, select this pattern; otherwise, if $|\mathcal{A}_4| > 1$, randomly choose a puncturing pattern in $\mathcal{A}_4$.

Steps 1 and 3 have a strong effect on decoding performance in the waterfall region, while Step 2 prevents the existence of a high error floor. Regarding Steps 1.2 and 3, for low rate punctured codes, even though Step 1.2 is more critical, the selection of puncturing patterns is typically dominated by Step 3, since there is often no difference in $\boldsymbol{E}_c^{a_i}$ among the various puncturing patterns in $\mathcal{A}_1$. As the punctured code rate increases, the choice of the best puncturing pattern in A_P^l relies more heavily on Step 1.2, since the enumerator vectors $\boldsymbol{E}_c^{\mathbf{a}_i}$ begin to show more variation in this case.

The complexity of Algorithm 1 is dominated by the computation involved in determining the variable nodes that participate in each of the short cycles. This computation only needs to be done once, however, for the mother code, since the list of variable nodes involved in each cycle is stored. Subsequently, for each puncturing pattern, the enumerators $C_\infty^{\mathbf{a}_i}$, $\boldsymbol{E}_c^{\mathbf{a}_i}$, $\boldsymbol{E}_\tau^{\mathbf{a}_i}$, and $\boldsymbol{E}_b^{\mathbf{a}_i}$ can be efficiently processed and compared, even for large puncturing period P. For example, it took approximately half a minute to obtain the list of variable nodes involved in the cycles of length 8, 10, and 12 for the $(21, 3, 5)$ Tanner LDPC-CC using a computer with a 2 GHz Processor and 2 GB of memory.

[5]Row permutations of a puncturing pattern $\mathbf{a}$ are considered to be *equivalent*. For example, the puncturing pattern $[10000; 00000]$ is equivalent to $[00000; 10000]$.



(a) $R' = 4/9$ (b) $R' = 4/8$

Fig. 4: BER of punctured LDPC-CCs of (3) with rates R'=4/9 and 4/8.

V. RC PUNCTURED LDPC-CCS DERIVED FROM THE $(21, 3, 5)$ TANNER LDPC-CC

In this section, the $(21, 3, 5)$ Tanner LDPC-CC is chosen as the mother code to illustrate the process of obtaining RC punctured LDPC-CCs. The polynomial syndrome former matrix is given by (3) and the code has rate $R = 2/5$, or equivalently $R = 4/10$. We set the puncturing period $P = 2$ and denote the puncturing pattern of the unpunctured code by $[00000; 00000]$. Then we can obtain a RC family of LDPC-CCs with rates 4/9, 4/8, 4/7, 4/6, and 4/5 by incrementally including an extra punctured entry in the puncturing pattern of the previous lower rate code, i.e., by changing one element in the previous puncturing pattern from "0" to "1". For this example, only the cycles of length 8, 10, and 12 were used to calculate elements in $\boldsymbol{E}_b^{\mathbf{a}_i}$ and $\boldsymbol{E}_\tau^{\mathbf{a}_i}$, i.e., we set $N_b = N_\tau = 12$ and $g = 8$ in Algorithm 1.

A. Rate $R' = 4/9$ punctured codes

We begin by placing $l = 1$ punctured entry in the puncturing pattern $[00000; 00000]$. Consequently, there are $\binom{Pc}{l} = \binom{10}{1} = 10$ different possible puncturing patterns; but only 5 are not equivalent, i.e., $A_2^1 = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5\}$. The enumerators calculated for A_2^1 are shown in Table I. Following Algorithm 1, we find that all the puncturing patterns contain zero ∞-SR nodes and one 1-SR node. Thus, all the patterns have the same recoverability of punctured nodes and are equally good choices in Step 1. Moreover, none of the patterns contain any CPCTSs (Step 2). However, according to Step 3, we find that puncturing pattern $\mathbf{a}_3$ (marked by an asterisk in Table I) is superior to the others, since it has the least number of punctured variable nodes involved in short cycles.

Simulation curves for the five puncturing patterns of the $R' = 4/9$ $(21, 3, 5)$ Tanner LDPC-CC are given in Fig. 4a. All the simulations were carried out assuming binary phase-shift keyed (BPSK) modulation on an additive white Gaussian noise (AWGN) channel with 50 iterations of the on-demand variable node activation [16] sum-product pipeline [11] decoding algorithm for LDPC-CCs. The simulation results confirm the superiority of puncturing pattern $\mathbf{a}_3$ compared to the other patterns.

B. Rate $R' = 4/8$ punctured codes

Based on the previous rate $R' = 4/9$ code with puncturing pattern $\mathbf{a}_3 = [00100; 00000]$, a punctured code with rate $R' = 4/8$ is obtained by replacing one "0" in $\mathbf{a}_3$ by a "1". Note that

A_2^1		$C_\infty^{\mathbf{a}_i}$	$\boldsymbol{E}_c^{\mathbf{a}_i}$	w = 8,10, and 12		
i	$\mathbf{a}_i$			$\boldsymbol{E}_\tau^{\mathbf{a}_i}$	$\boldsymbol{E}_b^{\mathbf{a}_i}$	$\sum_w \boldsymbol{E}_b^{\mathbf{a}_i}$
1	[10000; 00000]	0	[1]	[0 0 0]	[8 59 412]	479
2	[01000; 00000]	0	[1]	[0 0 0]	[10 72 447]	529
3	∗[00100; 00000]	0	[1]	[0 0 0]	[6 48 367]	421
4	[00010; 00000]	0	[1]	[0 0 0]	[12 71 426]	509
5	[00001; 00000]	0	[1]	[0 0 0]	[8 60 454]	522

TABLE I: Enumerators for the puncturing patterns of the $(21, 3, 5)$ Tanner LDPC-CC of rate $4/9$ obtained using cycles of length $w = 8$, 10, and 12.

A_2^2		$C_\infty^{\mathbf{a}_i}$	$\boldsymbol{E}_c^{\mathbf{a}_i}$	w = 8,10, and 12		
i	$\mathbf{a}_i$			$\boldsymbol{E}_\tau^{\mathbf{a}_i}$	$\boldsymbol{E}_b^{\mathbf{a}_i}$	$\sum_w \boldsymbol{E}_b^{\mathbf{a}_i}$
6	[10100; 00000]	0	[2]	[0 0 0]	[14 107 779]	900
7	[01100; 00000]	2	[0]	[0 0 1]	[16 120 814]	950
8	[00110; 00000]	0	[2]	[0 0 0]	[18 119 793]	930
9	[00101; 00000]	0	[2]	[0 0 0]	[14 108 821]	943
10	[00100; 10000]	0	[2]	[0 0 0]	[14 107 779]	900
11	[00100; 01000]	0	[2]	[0 0 0]	[16 120 814]	950
12	∗[00100; 00100]	0	[2]	[0 0 0]	[12 96 734]	842
13	[00100; 00010]	0	[2]	[0 0 0]	[18 119 793]	930
14	[00100; 00001]	0	[2]	[0 0 0]	[14 108 821]	943

TABLE II: Enumerators for the puncturing patterns of the $(21, 3, 5)$ Tanner LDPC-CC of rate $4/8$ obtained using cycles of length $w = 8$, 10, and 12.

the rate $4/8$ code is rate-compatible with the rate $4/9$ code. There are 9 possibilities for A_2^2 as shown in Table II.

Fig. 4b presents the simulation results for the $R' = 4/8$ punctured codes with puncturing patterns in A_2^2. Due to the existence of two ∞-SR nodes, puncturing pattern $\mathbf{a}_7$ has the worst performance, with a visibly inferior BER curve compared to the others. At Step 1, we find that all the puncturing patterns in $A_2^2 \backslash \mathbf{a}_7$ have the same recoverability of punctured nodes, i.e., they all have zero ∞-SR nodes and two 1-SR nodes. In addition, no CPCTSs exist for puncturing patterns in $A_2^2 \backslash \mathbf{a}_7$. However, at Step 3, we observe that pattern $\mathbf{a}_{12}$ has the smallest number of punctured nodes involved in short cycles. We see in Fig. 4b that this pattern also results in the lowest BER, confirming our choice.

C. RC LDPC-CCs derived from the $(21, 3, 5)$ Tanner LDPC-CC

For each higher rate, we follow the design procedure of Algorithm 1 to choose the best candidate puncturing pattern. As in the $R' = 4/9$ and $4/8$ cases, our choice was confirmed to be the best via code simulations. Fig. 5 shows the simulated performance of the family of RC punctured codes derived from the $(21, 3, 5)$ Tanner LDPC-CC. As expected, we observe that the decoding performance gets worse as the punctured code rate increases. We also see that, due to the large number of punctured entries in the puncturing patterns of rate $R' = 4/5$ codes, ∞-SR nodes are unavoidable. As a result, the BER curve of the associated codes is significantly worse than the others. In addition, the chosen $R' = 4/5$ punctured code has many CPCTSs, in particular, $\boldsymbol{E}_\tau^{\mathbf{a}_i} = [1\ 1\ 6]$, resulting in the existence of a high error floor. To avoid creating ∞-SR nodes in high rate RC punctured LDPC-CCs, the puncturing period must be increased. For example, by setting $P = 4$ and applying Algorithm 1, we obtain a family of RC codes whose puncturing pattern $[10100; 00110; 11101; 00110]$ for rate $R' = 8/10$ is free of ∞-SR nodes, resulting in about 2.1 dB gain at a BER of 10^{-3} and no discernible error floor compared to the $R' = 4/5$ code shown in Fig. 5. Details of obtaining families of improved RC punctured codes by extending the puncturing period P will be available in [17].

VI. Conclusions

Given a mother LDPC-CC and a puncturing period P, a novel method has been presented to find RC punctured LDPC-CCs. The positions of the punctured entries in the puncturing pattern are determined by several criteria: (1) ensure good recoverability of punctured variable nodes, (2) minimize the number of CPCTSs in short cycles, and (3) minimize the number of punctured variable nodes involved in short cycles. Crucially, the criteria can be efficiently and precisely computed, even for practically interesting time-invariant LDPC-CCs with large syndrome former memories, and we do not require the computation of code distance spectra. As an example, a family of RC punctured LDPC-CCs with puncturing period $P = 2$ and compatible rates $4/9$, $4/8$, $4/7$, $4/6$, and $4/5$ was obtained from the $(21, 3, 5)$ Tanner LDPC-CC.

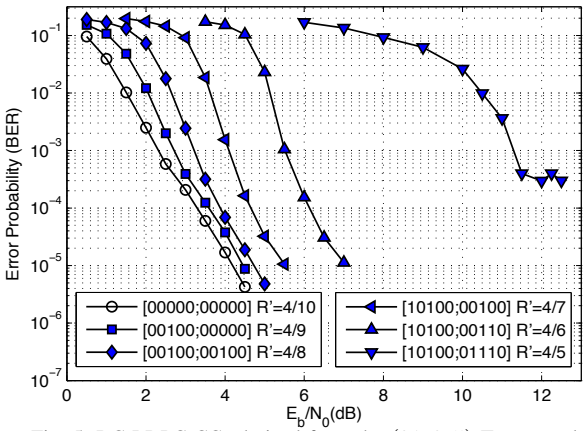


Fig. 5: RC LDPC-CCs derived from the $(21, 3, 5)$ Tanner code.

References

[1] T. Tian, C. Jones, and J. D. Villasenor, "Rate-compatible low-density parity-check codes," *in Proc. IEEE Int. Symp. Information Theory*, Chicago, USA, June 2004.

[2] M. R. Yazdani and A. H. Banihashemi, "On construction of rate-compatible low-density parity-check codes," *IEEE Trans. Commun. Letter*, vol. 8, no. 3, Mar. 2004.

[3] J. Li and K. R. Narayanan "Rate-compatible low-density parity-check codes for capacity-approaching ARQ scheme in packet data communications," *Proc. International Conference on Communications, Internet and Information Technology*, pp. 201-206, Virgin Islands, USA, Nov. 2002.

[4] J. Ha, J. Kim, D. Klinc, and S. W. McLaughlin, "Rate-compatible punctured low-density parity-check codes with short block lengths," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 728-738, Feb. 2006.

[5] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC Codes) and their applications," *IEEE Trans. Commun.*, vol. 36, no. 4, pp. 389-400, Apr. 1988.

[6] D. J. Costello, Jr., A. E. Pusane, S. Bates, and K. Sh. Zigangirov, "A comparison between LDPC block and convolutional codes," *Proc. Information Theory and Applications Workshop*, San Diego, USA, Jan. 2006.

[7] H. Zhou, D. G. M. Mitchell, N. Goertz, and D. J. Costello, Jr., "Distance spectrum estimation of LDPC convolutional codes," submitted to the *IEEE Int. Symp. Information Theory*, Cambridge, MA, USA, July 2012.

[8] T. Richardson, "Error-floors of LDPC codes," *in Proc. Forty-first Annual Allerton Conference*, Allerton, IL, USA, Sep. 2003.

[9] Z. Si, M. Andersson, R. Thobaben, and M. Skoglund, "Rate-compatible LDPC convolutional codes for capacity-approaching hybrid ARQ," *in Proc. IEEE Information Theory Workshop*, Paraty, Brazil, Oct. 2011.

[10] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello Jr., "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 2966-2984, Dec. 2004.

[11] A. J. Felström and K. Sh. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2181-2191, Sept. 1999.

[12] A. E. Pusane, R. Smarandache, P. O. Vontobel, and D. J. Costello, Jr., "Deriving good LDPC convolutional codes from LDPC block codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 835-857, Feb. 2011.

[13] H. Zhou and N. Goertz, "Cycle analysis of time-variant LDPC convolutional codes," *in Proc. 6th International Symposium on Turbo Codes and Iterative Information Processing*, Brest, France, Sept. 2010.

[14] M. Ivcovic, S. K. Chilappagari, and B. Vasic, "Eliminating trapping sets in low-density parity-check codes by using Tanner graph covers," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3763-3768, Aug. 2008.

[15] S. Lin and D. J. Costello, Jr., "Error control coding," 2^{nd} Ed., Pearson Prentice Hall, 2004.

[16] A. E. Pusane, M. Lentmaier, K. Sh. Zigangirov, and D. J. Costello, Jr., "Reduced complexity decoding strategies for LDPC convolutional codes," *in Proc. IEEE Int. Symp. Information Theory*, Chicago, USA, June 2004.

[17] H. Zhou, D. G. M. Mitchell, N. Goertz, and D. J. Costello, Jr., "Robust rate-compatible punctured LDPC convolutional codes," in preparation.