Efficient Ranking of Rate-Compatible Puncturing Patterns for a Given LDPC Code Matrix

Conference Paper · January 2010

DOI: 10.1109/GLOCOM.2009.5425627 · Source: IEEE Xplore

Wai Ho Mow
The Hong Kong University of Science and Technology
217 PUBLICATIONS **3,795** CITATIONS

Efficient Ranking of Rate-Compatible Puncturing Patterns for a Given LDPC Code Matrix

Sissi X. Wu, and Wai Ho Mow
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology
Email: {eesissi, eewhmow}@ust.hk

***Abstract*—In this paper, we introduce a novel criterion to rank puncturing patterns for rate-compatible LDPC codes. Specifically, based on Gaussian approximation density evolution, a cost function is devised to characterize the degree distribution of the punctured code matrices, which are derived from a mother code matrix by matrix transformation. This cost function allows us to effectively compare the expected performance of candidate puncturing patterns and to sort out good ones. Combined with well-designed search algorithms, the proposed criterion can be applied on both standardized Block-LDPC codes and generic binary LDPC codes to get good puncturing patterns with manageable complexity. Numerical simulation results verify the effectiveness of the proposed ranking criterion, and demonstrate that a series of good rate-compatible LDPC codes can be obtained by the proposed ranking criterion.**

***Index Terms*—Low-density parity-check (LDPC) codes, rate-compatible puncturing pattern, cost function, Gaussian approximation, stack algorithm.**

I. Introduction

Low-density Parity-check (LDPC) codes have been intensively researched due to the capacity approaching property and novel graph-based iterative decoding method. With superior performance and manageable decoding complexity, LDPC codes have been adopted as a mandatory element in recently developed industrial standards, such as DVB-S2, 3GPP2, IEEE802.20, IEEE802.11n, IEEE802.16e [1], and play a significant role in various communication systems. To overcome the fading effects as well as block-size mismatch issue in the practical wireless systems, the channel coding protocols shall be flexible in terms of the coding rate as well as the coding length. Typical examples include the application of incremental redundancy hybrid automatic-repeat-request (IR-HARQ) and the puncturing adjustment as specified in the IEEE 802.16m standard [2]. Such requirements motivate the developments of various rate-compatible LDPC codes.

Among the existing technologies, puncturing provides an efficient and simple way to acquire a series of rate compatible codes from the original LDPC code (the mother code). For LDPC codes, given a mother code, the error performance of the punctured code is determined by the corresponding *puncturing pattern*. It has been proven that capacity-approaching puncturing distributions exist for cycle-free code with infinite codeword length [3]. Current literatures show some pioneering works in finding puncturing codes for a given mother code. In [4], a bit-wise puncturing method is introduced by grouping all the codebits into different recovery groups (k-SR group). In [5], a puncturing scheme is provided for a specific mother code matrix with dual-diagonal block structure. However, the overall arrangement in [4] of the recovery groups are not deterministic but highly depends on random seeds and the structure of the mother code, hence, punctured codes with high rates are difficult to acquire by this approach. Meanwhile, the unpunctured bits are not guaranteed to keep an information set, so the encoder might not exist for some of the resultant groups arrangement. [5] only can be applied on LDPC codes with a dual-diagonal structure where only degree-two codebits are punctured, besides, it only considers the affect of the row degree distribution, which is obviously insufficient, since both the column degree distribution and row degree distribution can decide the performance of the LDPC code.

In this paper, we study a more general LDPC puncturing problem. The puncturing process is formulated as a code design problem subject to the mother code matrix and target code rates of the punctured code. Compared with the existing works, the proposed scheme has the following advantages: 1) rate-compatible codes with a large range of target rates can be obtained; 2) the design of puncturing patterns and the design of encoding matrix are decomposed so the encoding matrix always exists for the proposed algorithm, and for irregular code, high degree codebits can be assigned as information bits for unequal protection purpose; 3) the criterion is more precise since both the row and column degree distribution of the equivalent matrix for the punctured code are taken into account; and 4) a general case is considered without the constrains of dual-diagonal structure or block code structure.

The rest of the paper is as follows. The puncturing code model was introduced in Section II. In Section III, we formulated the puncturing problem as an optimization problem for the punctured code matrix, and based on this, an efficient cost function was devised to rank puncturing patterns. In Section IV, best-first stack search algorithm and non-greedy exhaustive search algorithm are proposed to facilitate sorting out good puncturing patterns by the proposed cost function. Numerical simulation results and conclusions are given in Section V, VI.

This work was supported by the Hong Kong Research Grants Councils under project number 617706.

$$H = \begin{array}{c|ccccc|} & v_1 & v_2 & v_3 & v_4 & v_5 \\ e_1 & 0 & 1 & 0 & 1 & 1 \\ e_2 & 1 & 0 & 1 & 0 & 1 \\ e_3 & 1 & 1 & 0 & 1 & 1 \end{array} \qquad \hat{H} = \begin{array}{c|ccccc|} & v_1 & v_2 & v_3 & v_4 & v_5 \\ \hat{e}_1 & 0 & 0 & 0 & 0 & 0 \\ \hat{e}_2 & 1 & 1 & 1 & 1 & 0 \\ \hat{e}_3 & 1 & 0 & 0 & 0 & 0 \end{array}$$

Fig. 1: An example for eliminating one bit by subtraction.

II. Puncturing Pattern Candidate Set

The decoding of the LDPC code can be performed on the corresponding Tanner Graph (TG) [6]. decoding an LDPC code can be performed on the corresponding Tanner Graph [6], since every parity check matrix is in a one to one correspondence with its Tanner Graph. Puncturing a parity bit from a codeword is equivalent to eliminating a variable node from the Tanner Graph of the mother code. From the view of check equations, a punctured node is not different from the "intermediate bit variable" involved in calculating the checksums associated with k parity check equations (i.e. k is the degree of the punctured bit). These intermediate nodes are the so called "invisible nodes" in the Factor Graph [6]. Hence, we can combine the parity check equations into $k-1$ parity check equations by eliminating the corresponding invisible dumb node. This corresponds to the "substraction" operation for solving an equation set. The Tanner Graph of the resultant parity check matrix is the equivalent decoding graph for the punctured code. Without loss of generality, consider the example as shown in Fig. 1. Given a codeword $(v_1, v_2, v_3, v_4, v_5)$, puncturing v_5 is equivalent to subtracting one of its associated equations (neighboring check nodes) such as e_1 from the parity check matrix (Tanner graph) H. Since v_5 is associated with three check equations, there are other two equivalent punctured matrices of different formats[1].

For the purpose of illustration, consider code C_0 with length n and rate r_0. In order to obtain a child code of rate $r_1 \geq r_0$, $x_{01} = \lceil n(1-r_0/r_1) \rceil$ bits need to be punctured. Let $\Omega(C_0, r_1)$ denote the *puncturing pattern candidate set* which contains all candidate puncturing patterns for the predecessor code C_0 and target rate r_1. We can get a child code C_1 with the puncturing patten $\mathbf{p} \in \Omega(C_0, r_1)$, where $\mathbf{p} = (p_0, p_1, p_2, ...p_{x_{01}})$ is a row vector denoting the puncturing bits' positions in the original codeword. The cardinality of $\Omega(C_0, r_1)$ is the number of combinations to choose x_{01} bits out of n codebits, i.e. $|\Omega(C_0, r_1)| = \binom{n}{n(1-r_0/r_1)}$. The candidate set Ω can be adjusted by judiciously deciding which bits are preserved and the order at which bits are punctured. Hence, the proposed puncturing scheme can flexibly keep useful bits (information set), or combine with other puncturing schemes. In general, exhaustively searching for good puncturing patterns from the candidate set Ω requires a complexity that grows exponentially with respect to the code length and the number of puncturing bits. In this paper, we want to provide criteria to efficiently sort out good patterns with manageable complexity.

[1]In this paper, we only consider subtracting with the first check equation.

III. Ranking Criterion

A. Optimized degree distribution

Consider a regular code with column degree d_v and row degree d_c, under Log-likelihood ratios (LLRs) message-passing (assuming all-zero codewords transmitted), and define s_{vc} and s_{cv} are LLR messages passing though the edges of the decoding TG. Thus, one message-passing decoding iteration can be divided into two rounds. In the first round, the variable nodes send message to the check nodes, e.g.

$$s_{vc}^{(l)} = s_{v_0} + \sum_{i=1}^{d_v-1} s_{cv,i}^{(l-1)} \tag{1}$$

where d_v is the degree of the variable node v, $s_{cv,i}, i = 1, 2...d_v - 1$, are $d_v - 1$ incoming LLRs from the neighbors of v except the check node c, s_{v_0} is the observed LLR of the output bit associated with v, and $l(> 1)$ is the current iteration time. In the second round, the check nodes pass messages to variable nodes, which can be expressed by the "tanh rule" as:

$$\tanh \frac{s_{cv}^{(l)}}{2} = \prod_{j=1}^{d_c-1} \tanh \frac{s_{vc,j}^{(l)}}{2} \tag{2}$$

where d_c is the degree of the check node c and $s_{vc,j}, j = 1, 2, ...d_c - 1$ are incoming LLRs from $d_c - 1$ neighbors of c except v.

The individual output of a variable or a check node can be well approximated by Gaussian distribution [7]. By keeping the mean of the Gaussian, the message passing decoding process can be analyzed as a one-dimension recursive density evolution model, which is known as Gaussian Approximation Density Evolution (GADE) [7]. Hence, in the first round of a decoding iteration, (1) can be simplified as

$$m_{vc}^{(l)} = m_{v_0} + (d_v - 1) m_{cv}^{(l-1)} \tag{3}$$

where m_{vc}, m_{v_0}, and m_{cv} are the mean of the corresponding LLRs. By taking the expectation on both sides of (2), we have

$$1 - \phi(m_{cv}^{(l)}) = [1 - \phi(m_{vc}^{(l)})]^{(d_c-1)} \tag{4}$$

where,

$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_R \tanh \frac{u}{2} e^{-\frac{(u-x)^2}{4x}} du, & \text{if } x > 0; \\ 1, & \text{if } x = 0. \end{cases}$$

We can write the equation (4) in a recursive fashion and obtain the following relations,

$$m_{cv}^{(l)} = \phi^{-1}(1 - [1 - \phi(m_{v_0} + (d_v - 1) m_{cv}^{(l-1)})]^{d_c-1}) \tag{5}$$

For an ensemble of random codes with irregular degree distributions $\lambda(x) = \sum_{i=2}^{d_v} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{i=2}^{d_c} \rho(x)_i x^{i-1}$, with d_v and d_c denoting maximum column and row degree, the above results can be extended to irregular codes by linearly combining the means from different degree nodes with weight

λ_i or ρ_i. Thus, the decoding process for irregular code can be expressed recursively by the following equation:

$$p_l = q(s, p_{l-1}) \tag{6}$$

where $q(s,p) = \sum_{j=2}^{d_r} \rho_j \phi^{-1}(1 - [1 - \sum_{i=2}^{d_l} \lambda_i \phi(s + (i-1)p)]^{j-1})$ and $s = m_{v_0} = 2/\sigma_n^2$ is the mean of the message from the AWGN channel (noise variance is σ_n^2), $p_l = m_{cv}^l$ and $p_0 = 0$.

At the lth iteration, the error probability $P_b^{(l)}$ is given by

$$P_b^{(l)} = \sum_{i=2}^{d_l} \frac{\lambda_i / i}{\sum_{j=2}^{d_l} \lambda_j / j} Q(\sqrt{\frac{s + ip_l}{2}}) \tag{7}$$

which means that the frame error probability of the code will decrease with respect to $p_b^{(l)}$. Let $\Delta = q(s,p) - p$ denote the convergence rate of $p_b^{(l)}$, as $s > 0$, $p \to \infty$, we have

$$\begin{aligned} \Delta &= s + (i_1 - 2)p - 4log\lambda_{i_1} - 4\sum_{j=2}^{d_r} \rho_j log(j-1) \\ &\quad + \mathcal{O}(p^{-1}) \end{aligned} \tag{8}$$

where λ_{i_1} is the first nonzero λ_i. Thus, the optimized degree distribution can be obtained by maximizing Δ subject to different constraints, which equivalently, to minimize $p_b^{(l)}$.

B. The cost function

Given a code matrix with degree distribution (λ, ρ) and target supporting rate r, let Ψ denote the matrix transformations (substraction) for achieving the target rate from the given code matrix, then $\Psi\{\lambda(x), \rho(x)\}$ represents the candidate set for all the possible resultant degree distribution of the equivalent punctured code matrix. We define a cost function as

$$f(\hat{\lambda}_{i_1}, \hat{\rho}(x)) := \log \hat{\lambda}_{i_1} + \sum_{j=2}^{d_r} \hat{\rho}_j \log(j-1) - c_0 \cdot \hat{i}_1 \tag{9}$$

where c_0 is a sufficiently large positive constant which is determined by the specific codelength. This cost function can be regarded as a sufficient performance measure of the candidate patterns. Hence, the optimized degree distribution of the transformed code can be obtained by the following optimization problem

$$\min_{\{\hat{\lambda}(x), \hat{\rho}(x)\} \in \Psi\{\lambda(x), \rho(x)\}} f(\hat{\lambda}_{i_1}, \hat{\rho}(x)) \tag{10}$$

With the efficient cost function, the challenge of the problem remains as how to effectively represent the candidate space. In the next section, we will provide some efficient ranking algorithms for the puncturing problems, which aim to reduce the search complexity and provide good rate compatible codes for different applications.

Note that the punctured code need to be decoded on at least girth-4 free decoding matrix so that the effective decoding performance can be guaranteed. In this paper, all the punctured codes are decoded on the original mother code matrix, which is delicately designed without short cycles, and the punctured bits are set as LLRs equal to "0". However, in general, it has not to be always referring to the mother code, especially in the scenario that only the parity bits are punctured. Instead of decoding on the mother code, any girth-4-free predecessor code matrix of the final punctured code could be fine to be the effective decoding matrix, which provides flexibility and fast convergence on the decoder.

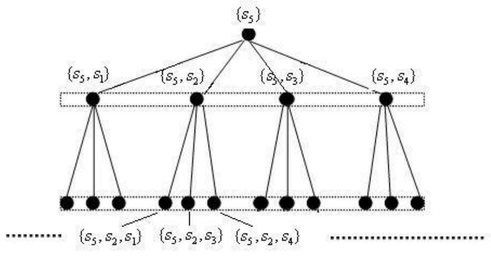


Fig. 2: An example of a pattern tree.

IV. Ranking Algorithm

The puncturing problem can be formulated as: for a given code C_0 with rate r_0, its child codes C_1, C_2... C_k with code rate r_1, r_2... r_k (where $r_1 \leq r_2 ... \leq r_k$) are obtained by consecutive puncturing. In the following, a tree-based search algorithm and a non-greedy (NG) exhaustive ranking algorithm will be introduced. Specifically, the tree search algorithm can handle the puncturing problem for generic LDPC codes for which the candidate space is rather huge and the NG algorithm is applicable for the standardized LDPC codes, for which the puncturing process can be applied in block-wise level and all the candidate space can be easily listed out.

A. Tree search stack algorithm

For a mother code with codelength N, enumerating all the codebit from 1 to N. Define $G(C_0)$ as a pattern tree, whose root node denotes the mother code C_0 or the already decided puncture pattern for the mother code, and all the other nodes represent the candidate puncturing codebits in the mother code matrix. Starting from the root, every time the tree extends one level, the corresponding puncturing pattern includes one more bit. Every partial path in this tree has a corresponding equivalent punctured matrix, which is obtained by puncturing the mother code matrix in the root node by the pattern consisting of the nodes (codebits) along this path. Fig.2 shows the pattern tree for the example code $\hat{H}$ in Fig.1. The root node corresponds to the mother code $\hat{H}$. If we see the code H as the mother code, then the root denotes the existing predecessor code with pattern s_5 under mother code H. Apparently, for code H, we can puncture at the most 3 codebits altogether (It is meaningless to puncture more than parity set), then, this pattern tree can extend two levels.

By defining the branch metric as the substraction of two involved partial path cost, G can be formulated as an additive tree [8]. To well measure every path in the pattern tree, we define an evaluation function as

$$e(k) = f(k) + h(k) \tag{11}$$

where k is the number of the current puncturing bits, $f(k)$ is the same as (9), denoting the cost of the currently evaluated path and $h(k)$ is a heuristic estimate of the cost for the remaining path [8]. To make the evaluation function efficient, we assume that the optimal path is always with decreasing cost $f(k)$. Suppose after the root node, p codebits need to be punctured, then at the level k (where we already punctured $k-1$ codebits) of G, the heuristic function is expressed as

$$h(k) = -c_1(p-k) \tag{12}$$

Here, c_1 is an empirical constant which indicates the branch metric (decreasing cost) between the two successive levels. With the tree extending, the function h can heuristically estimate the evaluated function for paths with different length. When $k = p$, the tree reaches its leaves, so the distance function is equal to 0, and the evaluation function just present the real cost of the path.

Using the evaluation function, the best-first stack algorithm [8] can be applied on the pattern tree to find good patterns as follows:

```
Algorithm I  Stack Algorithm
Initialization: : Set stack size and the value of $t$.
Step 1 : Load the stack with the cost of the root node
Step 2 : Calculate $e(k)$ of the successors of the top path,
         save $t$ successors with the least cost and delete
         the top path from the stack.
Step 3 : Insert the $t$ new paths in the stack, and rearrange
         the stack in order of increasing cost values.
Step 4 : If the top path in the stack is with path length
         less than $p$, go to Step 2. Else, Stop.
```

In Algorithm I, t is an adjustable parameter which decides how many new successor paths to keep in every step. There are two issues associated with the implementation of the stack algorithm: First, since the efficiency of the stack algorithm depends on the efficiency of the evaluated function $e(k)$, especially, the heuristic function $h(k)$, the empirical parameter c_1 is crucial to the stack algorithm. Here we suggest using the average value of all the successor branch for the current level. Second, the search performance can be substantially improved by increasing the stack size at the expense of complexity. Numerical results can be found in Section V to verify that very good puncturing patterns can be found by the stack algorithm.

B. *Non-greedy ranking algorithm*

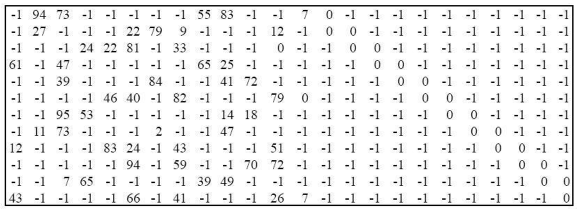
The aforemention tree search algorithm can greatly reduce the complexity of finding good puncturing patterns in the candidate set. However, in practical scenarios, the candidate set is limited and the exhaustive search algorithm could be applied to find better puncturing patterns. Next, a NG ranking algorithm for the standardized LDPC code will be shown, in which we can set priorities for different target supporting rates, if they are supposed to be used more frequently. We refer this method as a NG approach since normally the greedy approach first optimizes the higher rate code and the lower rate code is subsequently optimized based on the resultant higher rate code. Apparently, the NG algorithm is more general.

In the industrial standards, Block-LDPC (B-LDPC) codes, whose cyclic shifting structure is friendly to hardware architecture, are adopted. The parity check matrix for B-LDPC code can be seen as an array of square sub-block matrices, which are obtained by circular shifting an identity matrix, or are all-zero matrices. The parity check matrix is hence fully determined by means of these circular shifts and the square sub-block matrices dimension z. The LDPC codes defined with such structure rely on the concept of a base model matrix. Fig.3 gives the base model matrix of rate-1/2 LDPC code in IEEE 802.16e standard [1]. The base model matrix has a dimension of 12×24, each element in the matrix denotes the number of shifts, and "-1" denotes the all-zero matrix. With various values of z, different block lengths with the same rate can be achieved. The base model matrix is constructed to make sure that the parity check matrix with the smallest value of z has no short girth (at least girth 4). Apparently, for this type of LDPC codes, the puncturing can be operated block by block, therefore, the size of candidate set is not so huge and since in practice only the parity check matrix are punctured, which further reduces the size of candidate set. Hence, exhaustive search algorithm can be applied to acquire better performance.

Given a predecessor code C_0 of length n_0, we need to puncture $x_{0,i}$ bits to achieve the target rate $r_i \geq r_0$, where $x_{0,i} = \lceil n_0(1 - r_0/r_i) \rceil$, and there are $|\Omega(C_0, r_i)| = \binom{n_0}{n_0(1-r_0/r_i)}$ candidate puncturing patterns in set $\Omega(C_0, r_i)$. All the candidate puncturing patterns in set Ω can be ranked with cost function f in descending order. The top t puncturing patterns form the *winner set* of that code rate r_1 denoted by $w(C_0, r_1, t)$.

```
Algorithm II  Non-Greedy Ranking
Initialization: : $i = 1$
Step 1 : Calculate the cost $f$ for every candidate patterns
         in $\Omega(C_0, r_i)$, ranking the patterns in a descending
         order of cost values.
Step 2 : Choose the top $t$ puncturing patterns to form
         the winner set $w(C_0, r_i, t)$.
Step 3 : If $i \leq k$, $i = i + 1$, go to Step 1.
         Else, go to Step 4.
Step 4 : If $\exists p_j \in w(C_0, r_j)$, for $j = 1, 2, ...k$, satisfying
         $p_1 \subseteq p_2 \subseteq p_3, ... \subseteq p_k$, then $p_j (j = 1, 2...k)$ is
         just the series of final solutions.
```

In Algorithm II, Step 4, every puncturing pattern p_j is a set of parity bits. Whether a series of compatible p_j ($j =$



-1	94	73	-1	-1	-1	-1	-1	55	83	-1	-1	7	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	27	-1	-1	-1	22	79	9	-1	-1	-1	12	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	24	22	81	-1	33	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
61	-1	47	-1	-1	-1	-1	-1	65	25	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1
-1	-1	39	-1	-1	-1	84	-1	-1	41	72	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	46	40	-1	82	-1	-1	-1	79	0	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1
-1	-1	95	53	-1	-1	-1	-1	-1	14	18	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1
-1	11	73	-1	-1	-1	2	-1	-1	47	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1
12	-1	-1	-1	83	24	-1	43	-1	-1	-1	51	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1
-1	-1	-1	-1	-1	94	-1	59	-1	-1	70	72	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1
-1	-1	7	65	-1	-1	-1	-1	39	49	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0
43	-1	-1	-1	-1	66	-1	41	-1	-1	-1	26	7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

Fig. 3: The rate-$1/2$ LDPC code in IEEE 802.16e standard [1].

$1, 2...k$) exists depends on t: large t gives us high probability to find the rate-compatible patterns. Besides, since the cost function f is derived from GADE, it predicts the average error performance for the code ensemble. For individual code, the actual performance would deviate from the threshold. Thus, in $\Omega(C_0, r_i)$, for two different candidate puncturing patterns with comparable cost, they might be with similar error performance. Hence, simulated error performance (not too many) of the shortlisted patterns can be used to determine the more desirable one. The non-greedy ranking algorithm is typically applied in the case where the puncturing patterns are required for HARQ application, in which the original transmission and first retransmission are expected to succeed with higher probability, so we could choose punctured patterns with relative smaller cost for higher rates.

Example: For IEEE 802.16e HARQ application, the rate 1/2 code can be used as mother code to obtain target rate $2/3$, $3/4$ and $5/6$. The 24 column blocks in Fig.3 are enumerated as col.0-col.23 from the left to right, and combinations out of 12 parity column blocks (col.12-col.23) will be chosen from the parity part. Table I list the requirements for the puncturing problem. Table II lists the final solutions for the puncturing problem solved by Algorithm II. P1-P11-P111 and P2-P22-P222 are two groups of rate-compatible puncturing patterns with ranking orders listed. It is noted that the patterns for rate 5/6 code have highest priority, since this rate code is often used as the original transmission.

TABLE I: PUNCTURING EXAMPLE FOR IEEE 802.16e.

Targeted rate	pun. blocks	degree of freedom	priority
2/3	6	$\binom{12}{6}$=924	3 (low)
3/4	8	$\binom{12}{8}$=495	2
5/6	9.6	$\binom{12}{10}\binom{10}{1}$=660	1 (high)

V. Numerical Simulation

In this section, three types of simulation results will be provided to verify the effectiveness of the proposed schemes.

In Fig.4, examples are given to show how precise the cost function match the numerical frame error rate (FER) performance for different puncturing patterns. We exhaustively simulate all the possible puncturing patterns for IEEE802.16e rate 2/3B (3/4B) code removing 4 (3) column blocks and list their FERs at the target Eb/No as well as the cost value, so we have 70 (20) patterns altogether with exhaustive search. This

TABLE II: THE RESULTANT PUNCTURING PATTERNS FOR THE EXAMPLE OF IEEE802.16e.

Pattern	column block	cost ranking order
Patterns for rate 2/3 code-puncturing 6 blocks of 12 column blocks.		
Pattern	column block	cost ranking order
P1	13,15,16,19,21,23	8 of 924
P2	12,14,16,19,21,23	2 of 924
Patterns for rate 3/4 code-puncturing 8 blocks of 12 column blocks.		
Pattern	column block	cost ranking order
P11	12,13,15,16,19,20,21,23	6 of 495
P22	12,14,16,17,19,20,21,23	18 of 495
Patterns for rate 5/6 code-puncturing 9.6 blocks of 12 column blocks.		
Pattern	column block	cost ranking order
P111	12,13,15,16,17,19,20,21,23,22*	1 of 660
P222	12,13,14,16,17,19,20,21,23,22*	3 of 660

'*': Fractional part of the block is punctured.

is a typical puncturing problem for downlink partial used sub-channel (PUSC) configurations in IEEE802.16m [2]. When space-time codes (STC) are used, data allocation to subcarrier cluster is changed. In particular, when a four-antenna STC are used, some data subcarriers are reassigned as pilots to facilitate channel estimation, so we need to puncturing some parity bits when the LDPC code is applied. The simulation setting assumes the AWGN channel, a maximum of 50 iterations, a transmission block number of 10e8, and the belief propagation decoding algorithm. To show the result, we show the relationship between actual numerical FERs and the corresponding cost value for all the possible puncturing patterns[2]. It can be seen that, the cost values are consistent with the actual FERs. The embosses of the FER curves are always corresponding to large cost, and low FERs are always corresponding to small cost. So very good puncturing patterns can always be sorted out by the cost function.

Fig. 5 demonstrates that, for a given mother code matrix, when the locations of information bits are already fixed, the proposed algorithm performs better than the algorithm in [4]. The puncturing patterns are obtained by tree search stack algorithm with stack size 100. The proposed criterion separate the pattern design from the mother code matrix design and the encoding process, hence the encoding matrix always exists for our proposed ranking algorithms. Thus, for irregular code, high degree codebits can be assigned as information bits for unequal protection purpose. In this plot, the mother code is a (3,6)-LDPC code generated by Progressive Edge Growth (PEG) algorithm [9] with blocklengths 1024, rate 0.5, and the punctured codes with target rate 0.6 and 0.7. The simulation setting is the same as the previous simulation except

[2]In the plot, the order of the pattern index doesn't matter since our purpose is showing the correspondence between FERs and costs. Some scaling is made for FERs so that relationship can be obvious.

the transmission block number is 10e6. Note that, [4] gives different puncturing patterns for different random seeds so we show the best one by randomly simulating 50 times.

In Fig 6, simulated FER curves are given for the puncturing patterns listing in Table II compared with the puncturing patterns obtained by k-SR group algorithm in [4]. In Table II, we list two sets of rate-compatible puncturing patterns with comparable cost, and this plot tells that P2-P22-P222 should be adopted as the final puncturing pattern. It can be seen that, our algorithm outperforms [4] with gains in Eb/No of 0.25-0.3dB at rate 2/3 and 5/6, and be comparable at rate 3/4. We show the best curve by randomly simulating five times for [4]. Meanwhile, [4] can not acquire rate 5/6 directly, so several parity bits have to be randomly punctured to get rate 5/6. In this simulation, the superiority of the proposed algorithm is that we can set priories for different target rates, which is more considerate for practical requirements.

Numerical results verify that the proposed ranking algorithm is efficient and it can provide rate-compatible puncturing patterns with better performance than the previous puncturing schemes for different applications.

VI. Conclusions

In this paper, the LDPC rate-compatible puncturing problem has been formulated as an optimization problem subject to the mother code and target code rates. Based on the GADE, an efficient cost function was devised to shortlist puncturing patterns. The proposed ranking scheme considers both row degree and column degree distributions and can sort out good puncturing patterns from the candidate puncturing pattern set. Moreover, well-designed search algorithms are also introduced to obtain good rate-compatible puncturing patterns for different application requirements. Numerical simulation results show that the cost function is in very good consistency with the FER performance for the punctured codes. The proposed schemes can efficiently obtain a series of good rate-compatible punctured LDPC codes for practical communication systems at an affordable computational complexity.

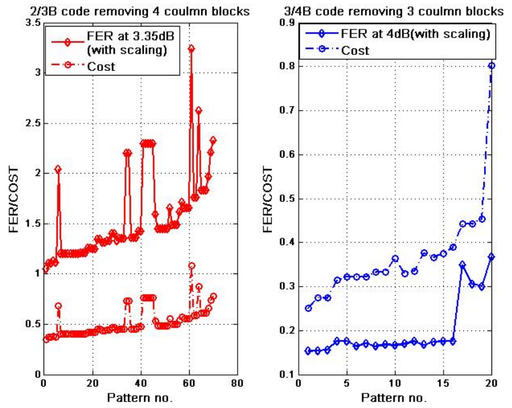


Fig. 4: Matching relationship between cost function and FERs.

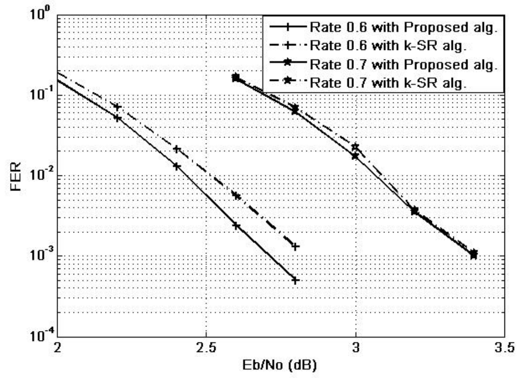


Fig. 5: Comparison between k-SR algorithm and proposed scheme for the mother code with given information bits' locations.

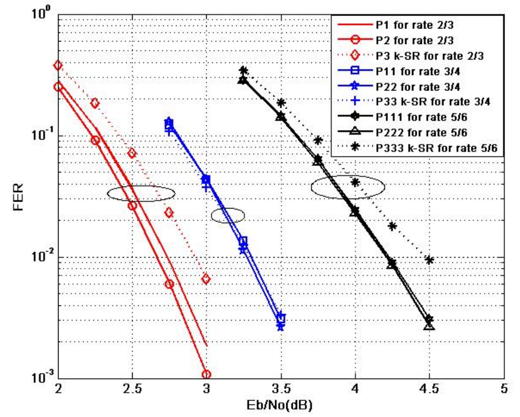


Fig. 6: FER curves for puncturing patterns under mother code (960,480) in IEEE802.16e.

References

[1] *IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems*, IEEE Std. 802.16e-2005, 2005.

[2] *IEEE 802.16m System Requirements*, IEEE Std. 802.16m-07/002r4, 2007.

[3] J. Ha, J. Kim, and S. W. McLaughlin, "Optimal puncturing distribution for rate compatible low density parity check code," *IEEE Trans. Inf. Theory*, vol. 50, pp. 159–161, Nov. 2004.

[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, pp. 728–738, Feb. 2006.

[5] S. N. Hong, H. J. Goo, and D. J. Shin, "Optimal puncturing of block-type LDPC codes and their fast convergence decoding," in *Proc. of IEEE Information Theory International Symposium (ISIT)*, 2006, pp. 826–830.

[6] T. J. Richardson and R. L. Urbanke, *Modern Coding Theory*, 1st ed. New York: Combridge, 2008.

[7] S. Y. Chung, T. J. Richardson, and R. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, pp. 657–670, Feb. 2001.

[8] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of A*," *Journal of the Association for Computing Mathinery*, vol. 32, pp. 505–536, 1985.

[9] X. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, pp. 386–398, Jan. 2005.