

INFSCO-ICT-216203 DA VINCI

D4.7

“Rate-Compatible Non-Binary LDPC Codes: Flexible and Robust Design”

Contractual Date of Delivery to the CEC:	30/06/2010 (as specified in the contract)
Actual Date of Delivery to the CEC:	28/02/2010
Author(s):	Weigang Chen, Charly Poulliat and David Declercq
Participant(s):	ENSEA
Workpackage:	WP4 Advanced Channel Coding
Estimated person months:	16 MM
Security:	PU (PU/PP/RE/CO)
Nature:	R (R/P/D/O)
Version:	1.0
Total number of pages:	22

Abstract: This Deliverable deals with the design and optimization of punctured coding schemes with a non-binary LDPC code as mother code, in order to obtain the property of rate adaptability for time varying channels or QoS.

In a first part of the deliverable, we propose a new design scheme of rate-compatible codes using $(2, d_c)$ non-binary LDPC codes over $GF(q)$ as mother codes and compares it with its counterpart in the literature. Both the two rate-compatible schemes use the bitwise puncturing methods. On the one hand, the proposed puncturing scheme is performed on parity-check symbols in a systematic form in order to have an efficient linear-time encoder. On the other hand, it punctures the parity-check symbols cluster by cluster to increase the achievable rate range. Finally, we verify the superiority of the systematic puncturing scheme in complexity and performance using computer simulations.

Although interesting in terms of performance, the proposed method assumes bitwise puncturing of the non-binary code, regardless of the constellation order (M-QAM). This strategy gives rise to an important complexity increase in the demodulation which computes the log-likelihood ratios (LLRs) of the punctured symbols. In order to bypass this issue, we propose in a second part a more pragmatic approach, based on symbolwise puncturing, which performs only slightly worse than the bitwise puncturing when the puncturing patterns and the code design are optimized in a joint fashion. We will use a protograph based approach in order to make the method valid for all codeword lengths.

Keyword list: DA VINCI, deliverable, internal report.

Document history	Ver.	Date
Document created.		25/01/10

Rate-Compatible Non-Binary LDPC Codes: Flexible and Robust Design

Weigang Chen, Charly Poulliat and David Declercq

Abstract

This Deliverable deals with the design and optimization of punctured coding schemes with a non-binary LDPC code as mother code, in order to obtain the property of rate adaptability for time varying channels or QoS.

In a first part of the deliverable, we propose a new design scheme of rate-compatible codes using $(2, d_c)$ non-binary LDPC codes over $GF(q)$ as mother codes and compares it with its counterpart in the literature. Both the two rate-compatible schemes use the bitwise puncturing methods. On the one hand, the proposed puncturing scheme is performed on parity-check symbols in a systematic form in order to have an efficient linear-time encoder. On the other hand, it punctures the parity-check symbols cluster by cluster to increase the achievable rate range. Finally, we verify the superiority of the systematic puncturing scheme in complexity and performance using computer simulations.

Although interesting in terms of performance, the proposed method assumes bitwise puncturing of the non-binary code, regardless of the constellation order (M-QAM). This strategy gives rise to an important complexity increase in the demodulation which computes the log-likelihood ratios (LLRs) of the punctured symbols. In order to bypass this issue, we propose in a second part a more pragmatic approach, based on symbolwise puncturing, which performs only slightly worse than the bitwise puncturing when the puncturing patterns and the code design are optimized in a joint fashion. We will use a protograph based approach in order to make the method valid for all codeword lengths.

CONTENTS

I	Introduction	2
II	Rate-Compatible Code Design by Puncturing	4
II-A	Non-Binary LDPC Codes as Mother Codes	4
II-B	Puncturing for Rate-Compatible Code Design	4
III	Puncturing via Node Grouping and Sorting	6
III-A	Node Grouping and Sorting	6
III-B	Puncturing Non-Binary LDPC Codes	6
IV	Encoding of Rate-Compatible Non-Binary Codes	8
IV-A	Linear-Time Encoding Methods	9
IV-B	Non-Systematic Mode and Complexity	9
V	Systematic Bitwise Puncturing for Non-Binary Codes	11
VI	Simulation Results for Bitwise puncturing and Comparisons	14
VII	Protograph Based Approach: Joint Design of Code Structure and Symbolwise Puncturing	17
VIII	Performance Results of the Proposed Symbolwise Puncturing Patterns	18
IX	Conclusion	20
	References	21

I. INTRODUCTION

In time-varying wireless channel, adaptive modulation and coding (AMC) and HARQ (Hybrid Automatic Retransmission reQuest) are often adopted to improve the transmission rate in wireless communications when channel state information or retransmission is available [1]. These two techniques have been both adopted in the next generation communication systems, such as, IEEE 802.16e (WiMAX), IEEE 802.11n, LTE, etc. Meanwhile, in order to approach the channel capacity, high performance channel codes are also needed.

The low-density parity-check (LDPC) code is a very promising candidate in the future wireless communications due to their relatively low implementation complexity and good error-correction performance. However, even LDPC codes can be implemented in low complexity, when AMC or HARQ is used, several encoder and decoders are needed to be integrated in a single terminal, and thus the silicon area and power consumption cause a great challenge. As a consequence, rate-adaptive LDPC codes with only one pair of mother encoder and decoder is a practical tradeoff for performance and complexity.

Puncturing is a very efficient design method to obtain rate-adaptivity by canceling transmission of different number of bits or symbols to achieve different code rates. Usually, it is assumed that the decoder knows the locations of punctured bits or symbols in the mother codes. If the punctured bits or symbols in a lower rate code are also punctured in a higher rate code, the rate adaptive channel code is called rate-compatible code. The asymptotic analysis of puncturing schemes for LDPC codes was investigated in [2], [3] and [4]. Although these results are all based on infinite code length, puncturing methods could be analyzed for finite length code ensembles using finite-length scaling in [5]. These asymptotical approaches are however not completely practical since they do not give explicit algorithms to design puncturing patterns for a fixed code. It is still a tough task to puncture a specific finite-length LDPC code with minimal performance degradation compared with independently designed codes. A noticeable improvement is introduced in [6], where the authors proposed a greedy algorithm to decompose the variable nodes in the Tanner graph into $p + 1$ sets $G_0, G_1, G_2, \dots, G_p$ by different recovery steps and also the nodes in each set are ordered. In this way, puncturing is executed in the order of G_1, G_2, \dots, G_p and also is performed in the sorted order for each set. Analysis and simulation both verify the efficiency of this design method. An improved scheme based on the method in [6] was proposed to improve the performance further in [7]. A special puncturing scheme for finite-length irregular repeat accumulate codes was also proposed in [8].

Non-binary cycle LDPC codes over high order fields $GF(q)$ have shown promising performance in short and intermediate code lengths compared with the state-of-art error correction codes in the literature [9], [10], [11], [12], [13] and they have been proposed in the DaVinci project [14]. Therefore, this kind of codes are very suitable to serve as the mother codes for rate-compatible codes. The puncturing method based on nodes grouping and sorting is also extended to the non-binary LDPC codes in [15] and [16]. However, no specific LDPC code structure is not considered in their design, and we wish to improve the design of puncturing patterns by taking into account a particular topological structure of the mother Tanner graph.

In the first part of this deliverable, we investigate the method in [15] and [16]. We point out that the puncturing method in [15] does not considering the structure of $(2, d_c)$ codes and may increase the implementation complexity. Based on this, we propose an even simplified method to achieve the rate-compatibility and has even lower complexity. The main contributions of the proposed method are concluded into two points: 1) consider the efficient encoding structure of $(2, d_c)$ non-binary LDPC codes and complexity; 2) puncturing the parity-check symbols by consecutive bits also called cluster to extend the code rate range. The proposed method also has the identical or even improved performance with reduced complexity compared with the puncturing method via node grouping and sorting in [15]. In Section II, the framework of rate-compatible code design by puncturing is presented. In Section III, puncturing via node

grouping and sorting is introduced and this will serve as a benchmark for the new proposed schemes. In Section IV, we investigate the efficient encoding problem for the codes proposed in DAVINCI project, and propose the systematic puncturing schemes based on the structure of the DAVINCI codes. In Section VI, the simulation results of the optimized scheme are presented and compared to the different existing puncturing schemes.

In the second part of the deliverable, we propose a very simple puncturing scheme which is based on the results obtained in the first part, and which takes into account the specificity of the hardware compliant code design that has been proposed in deliverable D4.5. Based on a particular protograph approach, we show that symbolwise puncturing is easy to implement with the “grouping and sorting” method, so that the puncturing pattern becomes trivial. Section VII explain the principle of the approach, and section VIII verifies by simulations that the proposed scheme is indeed close to optimal coding performance.

II. RATE-COMPATIBLE CODE DESIGN BY PUNCTURING

In order to design a rate-compatible coding scheme. We should first choose a good mother code or code ensemble. Then, the code or code ensemble is punctured to achieve a set of code rates, at which the codes also exhibit good practical performance compared to independently designed codes. We also follow this design frame. we choose $(2, d_c)$ non-binary LDPC code defined over $GF(q)$ as mother codes and bitwise puncturing to achieve a set of code rates. In this section, we first introduce the $(2, d_c)$ non-binary LDPC codes which serve as mother codes in this design frame and then introduce the puncturing methods to design higher code rates.

A. Non-Binary LDPC Codes as Mother Codes

Non-binary LDPC codes are linear block codes usually defined over high order finite fields $GF(q)$ and can also be represented with the associated Tanner graphs [9]. It has been proved that they are asymptotically good when defined on high order fields [10], [11]. Also, some improved design methods have been proposed to strengthen the performance [12], [13]. Therefore, in this deliverable, we design code compatible schemes based on this kind of codes.

Let $\mathbf{H} = (h_{i,j})_{M \times N}$ be the parity-check matrix of the non-binary LDPC code over $GF(q)$ ($q \geq 2$) and T the associated Tanner graph. Any non-zero entry $h_{i,j}$ in the parity-check matrix $\mathbf{H} = (h_{i,j})_{M \times N}$ of the code over $GF(q)$ ($q \geq 2$) is an element $\alpha^k \in GF(q)$, $k \in \{0, 1, 2, \dots, q-2\}$. In the Tanner graph, a non-zero value $\alpha^k \in GF(q)$ is assumed on each edge $e_{i,j} \in T$.

Non-binary LDPC codes can also be decoded using message passing decoding algorithms [9]. Also, some improved decoding algorithms have been proposed [17], [18], [19], [20], [21], [22]. In this deliverable, we first use the basic decoding algorithm based on FFT transformation to evaluate the relation between the structure and performance [9], [20]. It should be mentioned that in this deliverable all the punctured codes are decoded on the Tanner graphs of their mother codes in order to use only one encoder and decoder pair.

In this deliverable, we choose the $(2, d_c)$ non-binary LDPC codes as mother code to design rate-compatible schemes, for this codes exhibit significant performance compared with the state-of-the-art codes [13], [14]. We choose the mother code rate of 1/2 and puncture to 2/3 and 3/4. With this design scheme, the number of information bits in each frame is constant and it is very suitable for such techniques as HARQ.

B. Puncturing for Rate-Compatible Code Design

Puncturing is one of the most common method to construct rate-compatible codes. Another popular method is extending to achieve lower code rates. In this deliverable, we view extending as a reverse process of puncturing from the extended lowest-rate code. In this part, we first survey the fundamental properties of punctured LDPC codes to verify that puncturing is an efficient design method for rate-compatible code design.

Below, we list some basic properties for punctured LDPC codes [3], [4].

- 1) There is a punctured threshold p_{th} for randomly and intentionally punctured LDPC code ensembles.
- 2) For any rate R_1 and R_2 that $0 < R_1 < R_2 < 1$, there exists an LDPC code ensemble, which can be punctured from rate R_1 to R_2 resulting in asymptotically good codes for all rates $R_1 \leq R \leq R_2$.
- 3) For any ensemble of (λ, ρ) LDPC codes with rate $R_1 > 0$, and any R_2 satisfying $R_2 < R_1 < 1$, there exists an ensemble of punctured LDPC codes of R_1 with parent rate R_2 having the same threshold under the belief propagation algorithm.

From the conclusions above, we believe that good punctured codes can be obtained from the proper lower rate mother codes, though there exists a puncturing rate threshold. Also, it has been proved that properly designed puncturing codes can also asymptotically achieve the identical threshold as the independently designed codes and these codes are also asymptotically good [3], [4]. However, for finite-length case,

there are still some problems. The coding gain of a punctured code at each code rate is usually poorer than that of the corresponding dedicated code for a certain code rate, i.e. independently optimized code for the code rate. Therefore, punctured bits must be chosen carefully to minimize performance gap between dedicated and punctured codes in finite-length case. A selection pattern of punctured symbols is called a puncturing distribution for a required code rate. Then, how to choose the puncturing distribution is the central problem for rate-compatible code design. A very efficient puncturing design scheme based on node grouping and sorting has been proposed in [6]. We will briefly introduce this method in the next section.

However, it is even more complex to design rate-compatible codes by puncturing non-binary LDPC codes. For non-binary LDPC codes, each variable node (variable symbol) is represented by p bits if the non-binary LDPC code is defined over $GF(2^p)$. Therefore, for puncturing of non-binary LDPC codes, there exists a tradeoff between the number of punctured symbols and the number of punctured bits in each punctured symbol. In [15], a bitwise puncturing method of non-binary LDPC codes is proposed to achieve good performance especially in intermediate code rates. Their method can be decomposed into two steps [6], [15].

- 1) Determine the punctured symbols using symbol grouping and sorting method proposed in [6] and [15].
- 2) Determine the number of bits in each punctured symbol by bitwise spreading [15].

Simulation results have revealed that bitwise puncturing can achieve better performance compared to symbolwise puncturing for non-binary LDPC codes. Therefore, in a first part of the study, we mainly focus on bitwise puncturing schemes. This bitwise puncturing method will also be introduced in the next section.

III. PUNCTURING VIA NODE GROUPING AND SORTING

In this section, we survey the puncturing method using node grouping and sorting for finite-length binary LDPC codes proposed in [6] and the specific bitwise puncturing method when using non-binary LDPC codes as mother codes in [15].

A. Node Grouping and Sorting

In [6], the authors proposed a specific way to find the locations of punctured symbols in LDPC codes, which attempts to minimize the performance loss due to the puncturing and maintains the rate-compatibility.

The main contribution of this method is to introduce the concept of recoverable step. It is stemmed from the recovery in the BEC channel and also used in AWGN channel. First, we review some definitions in [6]. For example, a punctured variable node that has check nodes whose remaining neighboring variable nodes are unpunctured will have nonzero messages from the check nodes in the first iteration. The process for a punctured node to have messages from check nodes is called as recovery by analogy with the one over binary-erasure channels. The punctured node in the preceding example will be called a one-step-recoverable (1-SR) since the node is recovered in the first iteration. The 1-SR nodes and unpunctured nodes will help recover some of the remaining punctured nodes in the second iteration, and so on. In general, the punctured nodes recovered in the k -th iteration are called k -SR nodes.

The main idea hidden in the puncturing method via node grouping and sorting is based on the conjectures that the more iterations a punctured node needs for its recovery, the less statistically reliable the recovery message is and that a k -SR node connected with more survived check nodes will be recovered with a more reliable message. Thus, it is better to puncture nodes that require a smaller number of iterations and possess lower node degree, which results not only in less iterations to decode codewords but also in better performance at a given code rate.

In this deliverable, use $G_0, G_1, G_2, \dots, G_p$ to denote unpunctured variable node set, 1-SR variable node set, 2-SR variable node set, ..., p -SR variable node set. Use $C_0, C_1, C_2, \dots, C_p$ to denote survived check node set in 0-SR, 1-SR, 2-SR, ..., p -SR. Here, we use 0-SR to represent the initial information before recovery.

B. Puncturing Non-Binary LDPC Codes

For non-binary codes, each symbol (or each variable node) is represented by p bits if the non-binary codes are defined on $GF(2^p)$. Thus, there is a new freedom to puncturing each symbol entirely or partially.

In [6], based on the symbol grouping algorithm and observation on the LLR initialization process, a two-step method is presented to puncture the code bitwisely, especially for moderate code rates [15]. Specifically, first symbol grouping is executed on the underlying Tanner graph. With this scheme, we reduce the search space from all bits in a codeword to those bits included in the punctured symbols. Then, some bits are punctured using a bitwise pattern. When choosing the punctured bits, two basic principles are adopted. First, puncturing many bits per symbol should be performed exclusively on variable nodes with low levels of recoverability. At the same time, puncturing should be spread uniformly over all variable nodes of the same level of recoverability.

However, there are two problems in this scheme. One is the complexity problem considering encoding and this will be addressed in the next section. The other is that, many bits involved in the set G_0 can not be punctured. Thus, the number of the symbols which can be punctured (especially the symbols in G_1 which are recovered in only one step) is so limited and for the punctured symbols almost all the bits should be punctured. This reduces the performance of the rate-compatible codes and also limits the code rate range, especially when we consider the efficient encoding problem and only puncturing the parity-check symbols. In the following, we give an example.

Example 1:

Let us take as an example the non-binary code on the $(2, 4)$ regular Tanner graph with 48 check nodes

and 96 variable nodes. This non-binary code is defined over $GF(64)$. It has a rate of $1/2$ and code length 576 bits. Using the node grouping and sorting method in [6] and [15], we classify all the variable node into three sets G_0, G_1, G_2 and $|G_0| = 59, |G_1| = 32, |G_2| = 5$. In order to design a good higher rate code by puncturing, we give the following puncturing method. In this design, we puncture 192 bits bitwisely to achieve a code with code rate of $3/4$. In Table I and II, we give two puncturing distributions. In the first puncturing distribution, we puncture both 1-SR and 2-SR nodes. In the second one, we only puncture 1-SR nodes. Indeed, this is a symbolwise puncturing method.

Table I
PUNCTURED BIT DISTRIBUTION I

Type	Num.	Punct Sym	Punct. bit	all bits
0-SR	59	0	0	0
1-SR	32	32	5 or 6	167
2-SR	5	5	5	25
sum	96	37	/	192

Table II
PUNCTURED BIT DISTRIBUTION II

Type	Num.	Punct Sym	Punct. bit	all bits
0-SR	59	0	0	0
1-SR	32	32	6	192
2-SR	5	0	0	0
sum	96	32	/	192

These are the optimal puncturing methods according to the principles mentioned above. However, they don't consider the specific structure of $(2, d_c)$ non-binary codes and the efficient encoding problem. This will be addressed in the next sections.

IV. ENCODING OF RATE-COMPATIBLE NON-BINARY CODES

In this deliverable, we consider the efficient encoding and puncturing method of $(2, d_c)$ non-binary LDPC codes. Reducing the encoding complexity is a basic requirement in designing LDPC codes [23], including rate-compatible codes [24], [25].

For the DaVinci project, all the codes are the $(2, d_c)$ regular non-binary LDPC codes, which have the uniform variable node degree of 2 and check node degree of d_c [14]. In the design of these codes, all the check nodes in the underlying Tanner graphs are imposed to form a biggest single cycle only considering the check nodes. Without considering the variable nodes, the biggest cycle is called Hamiltonian cycle. In another viewpoint, we can view the $(2, d_c)$ non-binary LDPC codes as a kind of multi-edge type LDPC codes [26]. This can be observed in Fig. 1. In Fig. 1(a), the underlying Tanner graph of $(2, d_c = 4)$ non-binary LDPC code is presented, which includes 8 check nodes and 16 variable nodes. Note that in Fig. 1(b) all the variable nodes in the edges are eliminated for simplicity in this graph and all the 8 check nodes form a biggest cycle, also known as Hamiltonian cycle [27], [28]. Therefore, using the encoding method proposed in [29] and [30], these code can be encoded in linear time. The specific encoding method can be found in [29], [30]. With this design constraint, the description complexity of the graph is also reduced, for the link of subgraph associated with parity-check symbols doesn't need record.

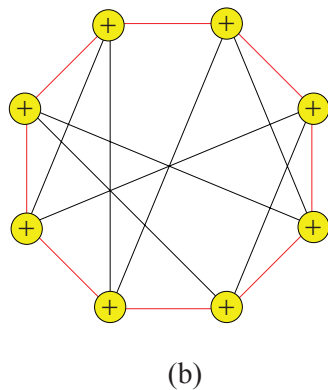
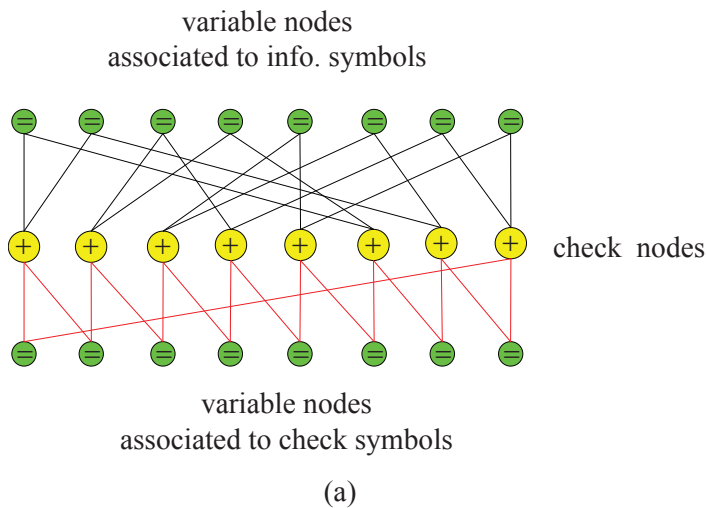


Figure 1. The underlying Tanner graph for efficiently encodable $(2, d_c)$ regular non-binary LDPC codes. Here, we only give a small example with 8 check nodes and 16 variable nodes.

A. Linear-Time Encoding Methods

In this part, the efficient encoding of the non-binary LDPC codes in DaVinci Project is investigated. First, we arrange the parity-check matrix as $\mathbf{H} = [\mathbf{H}_i, \mathbf{H}_c]$ and guarantee the matrix \mathbf{H}_c has an inversion with proper non-zero entries. In order to guarantee the full rank of \mathbf{H}_c , the full rank criterion introduced in [13] can be used to choose the non-zero entries. In this way, we configure the information symbols on the edges not included in the biggest cycle while the parity-check symbols on the edges in the biggest cycle as in Equation 1. The main advantage of encoding based on decomposing the graph into hamiltonian cycle and chord edges is that only little data need be stored and efficient encoding can be easily implemented.

$$\mathbf{H} = \begin{bmatrix} \alpha^{k_{0,0}} & 0 & 0 & \cdots & 0 & \alpha^{k_{0,1}} \\ \alpha^{k_{1,0}} & \alpha^{k_{1,1}} & 0 & \cdots & 0 & 0 \\ \mathbf{H}_i, & 0 & \alpha^{k_{2,0}} & \alpha^{k_{2,1}} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha^{k_{(M-2),1}} & 0 \\ 0 & 0 & 0 & \cdots & \alpha^{k_{M-1,0}} & \alpha^{k_{M,1}} \end{bmatrix} \quad (1)$$

Then, the non-binary $(2, d_c)$ LDPC code can be encoded using the parity-check matrix directly. Let $\mathbf{c} = [\mathbf{m}, \mathbf{p}]$ denote a codeword, where \mathbf{m} and \mathbf{p} denote information bit vector and check bit vector, respectively. Thus, we have

$$\mathbf{H} \cdot \mathbf{c}^T = [\mathbf{H}_i, \mathbf{H}_c]_{M \times N} \cdot [\mathbf{m}, \mathbf{p}]_{1 \times N}^T = \mathbf{0}. \quad (2)$$

Here, \mathbf{H}_c is the right part in Eq. (3), then

$$\mathbf{p}^T = \mathbf{H}_c^{-1} (\mathbf{H}_i \cdot \mathbf{m}^T). \quad (3)$$

Obviously, the computation of $\mathbf{H}_i \cdot \mathbf{m}^T$ can be executed directly for the sparseness of \mathbf{H}_i . However, for direct encoding with Eq. 3, the inversion matrix \mathbf{H}_c^{-1} is usually dense. Then, if Eq. 3 is used for encoding, the number of multiplication computation is M^2 and the number of the addition computation is $M(M-1)$ if accumulator is used. For very long code, this is not an efficient encoding method. It is obviously that the direct encoding has the computation complexity of $O(M^2)$.

In order to implement the linear encoding of the $(2, d_c)$ LDPC code over $GF(q)$, we have the following representation

$$\mathbf{H}_c \cdot \mathbf{p}^T = \mathbf{H}_i \cdot \mathbf{m}^T = \mathbf{s} \quad (4)$$

and then the recursive computing method for \mathbf{p} from \mathbf{s} can be used as proposed in [29] and [30]. We also assume the coefficients have been stored before computing and all the division is computed using the inversion symbols which are also stored. Then the computation includes $(2M-2)$ additions, $(3M-1)$ multiplications over $GF(q)$. It can be observed the $(2, d_c)$ LDPC code is encoded in linear time. Indeed, we can even lower the complexity of the encoder by very small modification [31].

B. Non-Systematic Mode and Complexity

However, there are some problems when the codes are used as mother codes with puncturing method surveyed in Section 3. When using the puncturing method in [6] and [15], the punctured symbols may include information symbols according to the efficient encoding method introduced in the previous part. Indeed, there exist two methods to deal with this problem.

The first method is to rearrange all the symbols before transmission. The code is still encoded using the linear efficient encoding method but is punctured in a non-systematic form. The rate-compatible encoding process can be decomposed into two steps: encoding and puncturing. First, encoding is also performed

using Eq. 1 and Eq. 4. Then, in order to implement the rate compatibility, we rearrange the symbols and use another vector $\mathbf{c}_\pi = [\mathbf{c}_{\pi_0}, \mathbf{c}_{\pi_1}, \dots, \mathbf{c}_{\pi_L}]$ to denote which symbols are punctured for each code rate, i.e., the puncturing pattern. Here, we use the puncturing vector \mathbf{c}_π to denote the L punctured vectors for the L code rates besides the mother code rate. The code rate of the mother code is R_0 . If we punctured \mathbf{c}_{π_1} , we get the code rate R_1 , puncture $\mathbf{c}_{\pi_j}, 1 \leq j \leq 2$, to get R_2 and $\mathbf{c}_{\pi_j}, 1 \leq j \leq 3$ to get R_3 , and so on. In this way, we can get all the L code rates. Obviously, this increases the complexity. With this method, the rate-compatible codes are used in a non-systematic mode. It not only increases the complexity in the encoder and decoder, but also increases the transmission delay using the non-systematic mode.

The second method is to adopt preprocessing. We first group the punctured symbols as parity-check symbols. However, in this way, it may be difficult to obtain the efficient encoding structure and the encoding can only be executed using Eq. 3. Therefore, due to the density of \mathbf{H}_c^{-1} , it can not be encoded in linear time. This also increases the implementation complexity.

Due to the increased complexity of the two methods, a new systematic puncturing method for non-binary LDPC codes is presented in the next section.

V. SYSTEMATIC BITWISE PUNCTURING FOR NON-BINARY CODES

In this deliverable, we propose a new systematic puncturing scheme for non-binary LDPC codes, which can be encoded in linear time and punctured in a systematic way. In this scheme, we exploit the structure property of the non-binary $(2, d_c)$ LDPC codes proposed for the DaVinci project [14] as mother codes.

This method is different in two points from the puncturing methods via node grouping and sorting.

- 1) This method is to puncture only the parity-check symbols and all the information symbols are grouped into the set G_0 . That is, the puncturing is performed in a systematic way.
- 2) The parity-check symbols are punctured cluster by cluster rather than symbol by symbol. For example, two consecutive variable nodes are assembled into a cluster, which only involve one dead check node in the first recovery step. When we choose puncturing nodes, the two consecutive variable nodes in the cluster are punctured simultaneously. In this way, we change the node grouping and sorting method symbol by symbol to a cluster-by-cluster mode.

In order to grantee the low complexity of systematic puncturing, according to the observation in [6], we can puncture only the parity-check symbols via node grouping and sorting to obtain several code rates in the systematic form. However, in order to grantee that all the punctured parity-check symbols can be recovered from the unpunctured symbol in relatively less steps even one step, the number of nodes which can be punctured is limited. We show the shortage of this method with an example in Fig. 2. Assume we only allow one step nodes and a possible grouping process is shown in Fig. 2. If we group one variable node v_i associated to parity-check symbols into G_1 , v_j into G_0 , c_p into C_1 , and c_q into C_0 , we can not puncture variable nodes v_j for $v_j \in G_0$, and we can neither puncture variable nodes v_k because v_k connects to c_q and c_q has been connected to a punctured node v_i . In this way, at most only half of parity-check symbols can be punctured. This limits the punctured code rate range. For example, let assume the number of check nodes can be divided by 2 and the achievable code rate lies in the region $R = [R_0, 2R_0/(1+R_0)]$. If the mother code rate $R_0 = 1/2$, the achievable rate region $R = [1/2, 2/3]$ and it is limited.

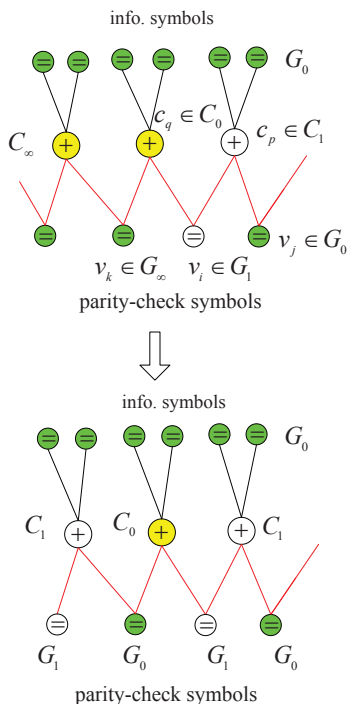


Figure 2. Shortage of the systematic puncturing method only performed on parity-check symbols for $(2, d_c)$ non-binary LDPC codes. The blank variable nodes represent punctured variable nodes associated with parity-check symbols, and the green nodes represent unpunctured variable nodes associated with parity-check nodes or information nodes.

In order to combat this problem and extend the achievable code rate range by systematic puncturing,

we first assemble the parity-check symbols in a cluster by cluster. Fig. 3 gives an example in which the punctured nodes are assembled pair by pair. In Fig. 3, the blank variable nodes are punctured nodes and the green ones are unpunctured. That is, there are two punctured nodes between two unpunctured nodes forming a punctured node cluster as shown in Fig. 4. It can be observed that in this subgraph, the two punctured symbols can be recovered in only one step, though there is a dead check nodes in the first recovery step. This satisfies the consideration in the method [15]. In this way, we can achieve good performance and minimize the performance loss due to puncturing. This will be verified in the next section using computer simulations.

We can explain the proposed method in several aspects. First, more punctured positions can be located. Indeed, at most $2/3$ parity-check symbols can be punctured with this method, which will enlarge the achievable code rate range. Second, the systematic puncturing method has the advantage in lower complexity. In this method, there are at most $2/3$ punctured parity-check symbols which can be recovered in the first step, which are evenly distributed in all the parity-check symbols. Thus, we do not need to record the positions of punctured symbols. In our experiments, if we puncture equal number of bits in each symbol and in equal positions, we do not need to record the punctured bit position. For example, we can puncturing all the first several bits in each symbols. This does not degrade the performance. Thus, the systematic method proposed in this deliverable exhibits lower complexity compared to the method in [15] in two aspects: efficient encoding and regular puncturing distribution. Third, this puncturing exhibits better performance when using the non-binary LDPC codes with short code lengths as designed in [14]. This will be shown in the next section.

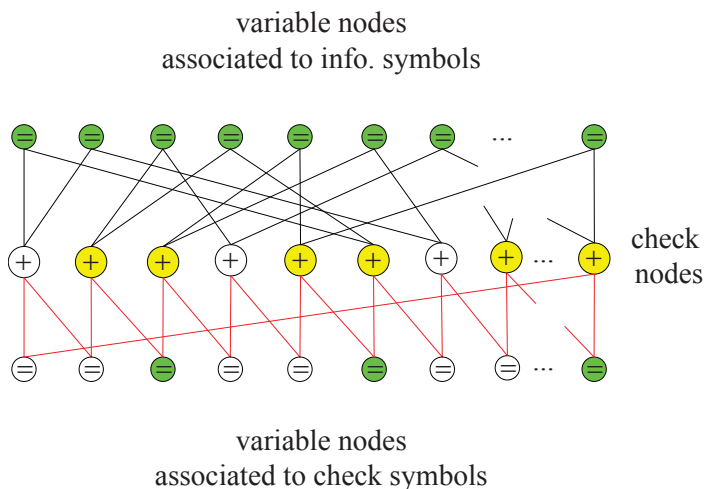


Figure 3. The systematic puncturing pattern for $(2, d_c)$ non-binary LDPC codes. The blank variable nodes represent punctured variable nodes associated with parity-check symbols, and the green nodes represent unpunctured variable nodes associated with parity-check nodes or information nodes. The blank check nodes represent the dead check nodes in the first recovery step while the yellow ones survived check nodes.

In conclusion, the proposed scheme has two advantages over the puncturing scheme via node grouping and sorting. First, this method has a systematic encoding method with a regular puncturing distribution and thus has a relatively lower complexity compared the two method in the previous section. On the other hand, the systematic puncturing method does not cause performance degradation and it even surpasses the puncturing method via node grouping and sorting in some cases. This will be verified via computer simulations in the next section.

However, this puncturing method also has a code rate range. Let assume the number of check nodes can be divided by 3. If the symbols are clustered pair by pair and we only allow 1-SR nodes, the achievable code rate lies in the region $R = [R_0, 3R_0/(1 + 2R_0)]$. If the check node number can not be divided by 3, the code rate range is a little different. In this deliverable, we choose the mother code rate is $1/2$ and the

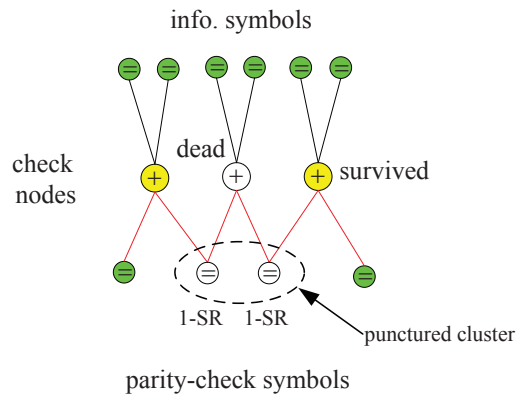


Figure 4. A punctured cluster for $(2, d_c)$ non-binary LDPC codes. The blank variable nodes represent punctured variable nodes associated with parity-check symbols, and the green nodes represent unpunctured variable nodes associated with parity-check nodes or information nodes. The blank check nodes represent the dead check nodes in the first recovery step while the yellow ones survived check nodes.

check node number of the underlying Tanner graph can be divided by 3. Thus, the code rate range can be achieved is $[1/2, 3/4]$. Strictly speaking, the code can be punctured bit by bit and the code rate can be achieved in a very fine grain. In this deliverable, we choose two code rate $2/3$ and $3/4$ as examples. Therefore, this method also has a code rate limitation as the puncturing method in [15].

Furthermore, if we want to extend the code rate further and can allow 2-SR nodes, we can extend the achievable code rate range to $R = [R_0, 5R_0/(1 + 4R_0)]$ with the assumption that the number of check nodes can be divided by 5. Otherwise, there exists a little difference in the range. If we also use the mother code rate of $1/2$, we can obtain a highest code rate of $5/6$. In this way, there are four punctured nodes between a pair of unpunctured nodes along the biggest cycle in the Tanner graph as shown in Fig. 5. In this case, 2-SR nodes are allowed to exist.

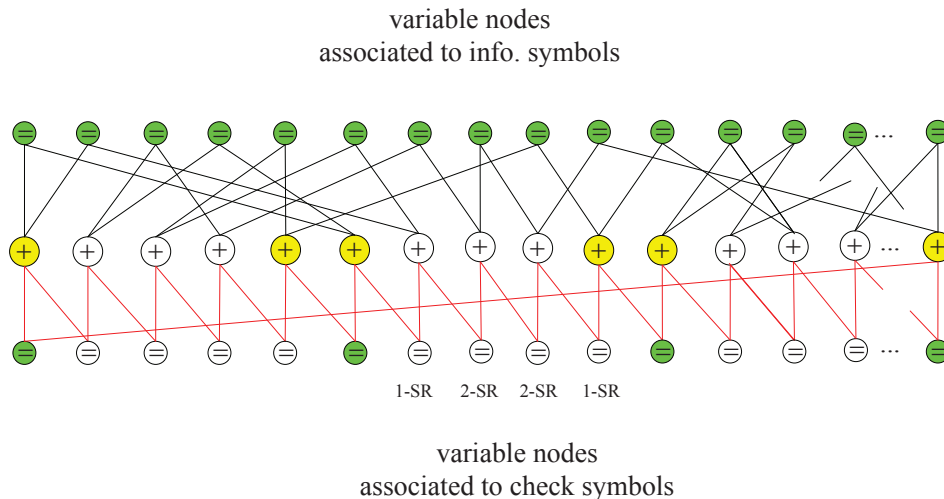


Figure 5. The systematic puncturing pattern from code rate $1/2$ to $5/6$ for $(2, d_c)$ non-binary LDPC codes. The blank variable nodes represent punctured variable nodes associated with parity-check symbols, and the green nodes represent unpunctured variable nodes associated with parity-check symbols or information symbols. The blank check nodes represent the dead check nodes in the first recovery step while the yellow ones survived check nodes.

VI. SIMULATION RESULTS FOR BITWISE PUNCTURING AND COMPARISONS

To verify the performance of the proposed systematic puncturing method in this deliverable, this section presents the simulation results to compare it with the puncturing method via node grouping and sorting for non-binary LDPC codes in [15]. We also compare it with some symbolwise puncturing and random puncturing methods for non-binary LDPC codes.

We present two examples. In the first simulation example, we choose the $(2, d_c)$ non-binary LDPC codes over $GF(64)$ of code rate $1/2$ as mother code [14]. The code length is 288 bits. Punctured codes with code rate $2/3$ using different puncturing methods are designed. We investigate five puncturing methods including symbolwise and bitwise method listed as follows:

- 1) Symbolwise random puncturing.
- 2) Symbolwise puncturing via node grouping and sorting.
- 3) Bitwise random puncturing.
- 4) Bitwise puncturing via node grouping and sorting [15].
- 5) Proposed systematic bitwise puncturing.

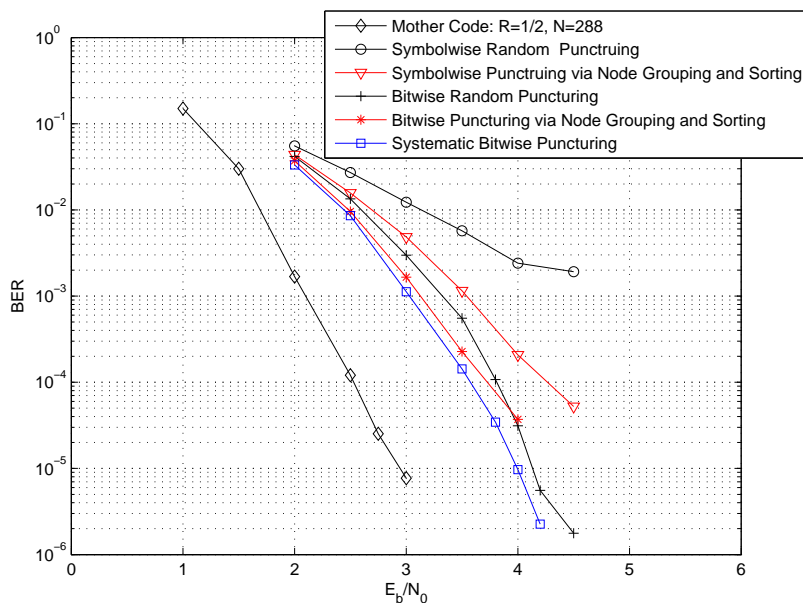


Figure 6. BER Performance comparison of different puncturing methods using the $(2, 4)$ non-binary LDPC codes over $GF(64)$ of code length 288 bits as mother code. The punctured code rate is $2/3$ and the code length is 216 bits.

In Fig. 6, it is observed that bitwise puncturing methods usually surpass the symbolwise methods if the puncturing is performed either randomly or via node grouping and sorting. Moreover, it is also observed that the proposed systematic method achieves the best performance among all these methods.

In the second simulation example, only bitwise puncturing methods are compared and two punctured code rates are included. We also use the $(2, d_c)$ non-binary LDPC codes over $GF(64)$ of code rate $1/2$ as mother code [14]. The code length is 576 bits and information bit number is 288. The two punctured code rates are $2/3$ and $3/4$. If more code rates lower than $3/4$ are needed, we just need to decrease the number of the punctured bits in some punctured symbols. This can be done in a straightforward way. In all these code rates, the information bit numbers are all 288, which is constant. It is quite suitable for type-II redundancy incremental HARQ.

The BER and FER performance are presented in Fig. 7 and Fig. 8. It can be observed that for the code rate of $3/4$, the systematic puncturing method achieved the best performance compared with the random bitwise puncturing and bitwise puncturing via node grouping and sorting [15]. For the code rate

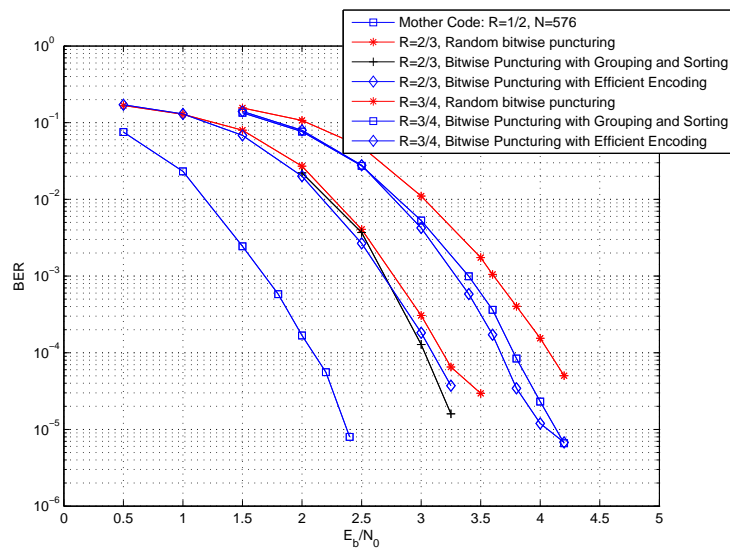


Figure 7. BER Performance of rate-compatible code design schemes based on the $(2, 4)$ non-binary LDPC code with code rate of $1/2$ and code length of 576 bits. The two punctured higher code rates are $2/3$ and $3/4$.

of $2/3$, the improvement is narrowed and the systematic puncturing has almost identical or a little better performance compared with bitwise puncturing via node grouping and sorting. The two methods both surpass the random bitwise puncturing.

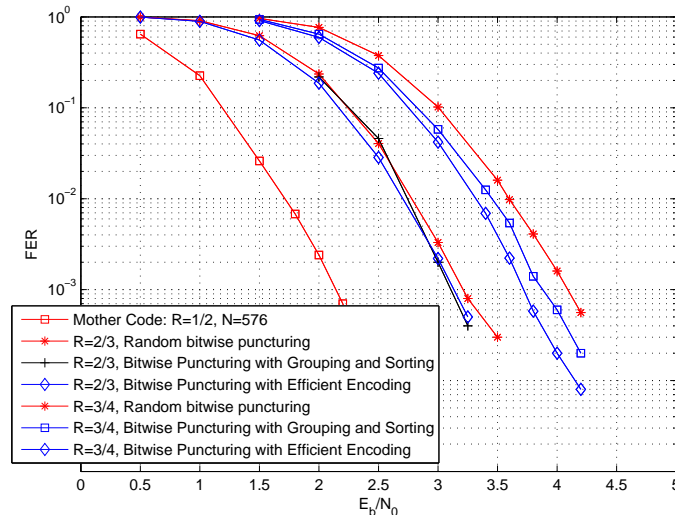


Figure 8. FER Performance of rate-compatible code design schemes based on the $(2, 4)$ non-binary LDPC code with code rate of $1/2$ and code length of 576 bits. The two punctured higher code rates are $2/3$ and $3/4$.

Therefore, we conclude that the systematic puncturing developed in the DAVINCI project and presented in this deliverable is a very promising scheme for the rate-compatible codecs with $(2, d_c)$ non-binary LDPC codes as mother codes.

However, this partial conclusion is based on the assumption that the communication system can deal easily with punctured bits or group of bits, which is not necessarily the case.

- In transmission modes which use only QPSK, the proposed approach does not suffer from any drawback and we advice the use of the proposed bitwise puncturing strategy (cf. section V).

- When a larger constellation is employed, like M -QAM with $M > 4$, the bitwise puncturing renders the demapping process cumbersome. This is also the case when bit allocation for efficient spectrum occupation is used, and several different orders of M -QAM span the same codeword. In this case, it is more desirable to stick with a rate-compatible scheme at the symbol level, which is more convenient to implement in the mapping/demapping process. Symbolwise puncturing could show noticeable performance degradation if not combined with a specific code design method. In the next section, we propose a new joint design of the Tanner graph of the code and the puncturing pattern which limits the performance loss due to the symbolwise puncturing.

VII. PROTOGRAPH BASED APPROACH: JOINT DESIGN OF CODE STRUCTURE AND SYMBOLWISE PUNCTURING

In this section, we propose a joint strategy of optimization for the puncturing pattern and the code design, in order to ensure good error correcting performance, while using a symbolwise puncturing. By doing so, we will adapt the code design proposed in the DAVINCI project and presented in deliverable D4.5 so that the puncturing strategy will be trivial to implement, which is a desirable property for the overall system. For more information about protograph based LDPC codes and the optimization of the design, please refer to deliverable D4.5.

The base matrix for the design of the punctured pattern is a rate $R = 0.5$ code with dimension $M_m = 6$ and $N_m = 12$, and is depicted on figure 9. This protograph is especially interesting for our needs since it has the following properties:

- The first two blocs have a structure similar to repeat-accumulate codes, which make it easy to encode, in linear time, with minimum extra hardware complexity (cf. deliverable D4.5),
- One can puncture either Bloc-3 or Bloc-4 with the property that the unpunctured symbols have *all* the 1-SR property. This ensures that a simple puncturing strategy will give good error correcting results for all rate in the range $[\frac{1}{2}, \frac{2}{3}]$.
- The base matrix has maximum girth $g = 6$. We have chosen the location of the nonzero entries in the matrix such that the number of 6-cycles is minimized. This allows to reach maximum girth Tanner graph after the lifting operation (cf. deliverable D4.5).

Protograph Type-1

0	1	1	0	0	0	0	0	1	0	1	0
1	0	1	0	0	0	1	0	0	0	0	1
1	1	0	0	0	0	0	1	0	1	0	0
0	0	0	0	1	1	0	1	0	0	0	1
0	0	0	1	0	1	0	0	1	1	0	0
0	0	0	1	1	0	1	0	0	0	1	0
⏟			⏟			⏟			⏟		
Bloc 1			Bloc 2			Bloc 3			Bloc 4		

Figure 9. Simplest Protograph for NB-LDPC rate-compatible code design.

As a result, the proposed protograph allows to have a trivial puncturing pattern. The symbols to be punctured are sorted from the last column of the lifted matrix in a decreasing order, as shown on figure 10. The specific code design ensures that this trivial puncturing pattern is optimal in terms of 1-SR symbol recovery, up to the rate $R = \frac{2}{3}$.

The only expected drawback of this structure is that when one to puncture more than 1/4th of the codeword to reach rates greater than $R = \frac{2}{3}$, the 1-SR property of remaining symbols is not ensured anymore and then performance degradation compared to the optimal puncturing patterns is expected. We although think that the most important modes of 4G transmissions will use rates mostly in the range $[\frac{1}{2}, \frac{2}{3}]$, and the extreme simplicity of the scheme presented in this section fully justifies the approach. In the next section, we compare the performance of this simple puncturing scheme with the optimal bitwise puncturing presented in section V.

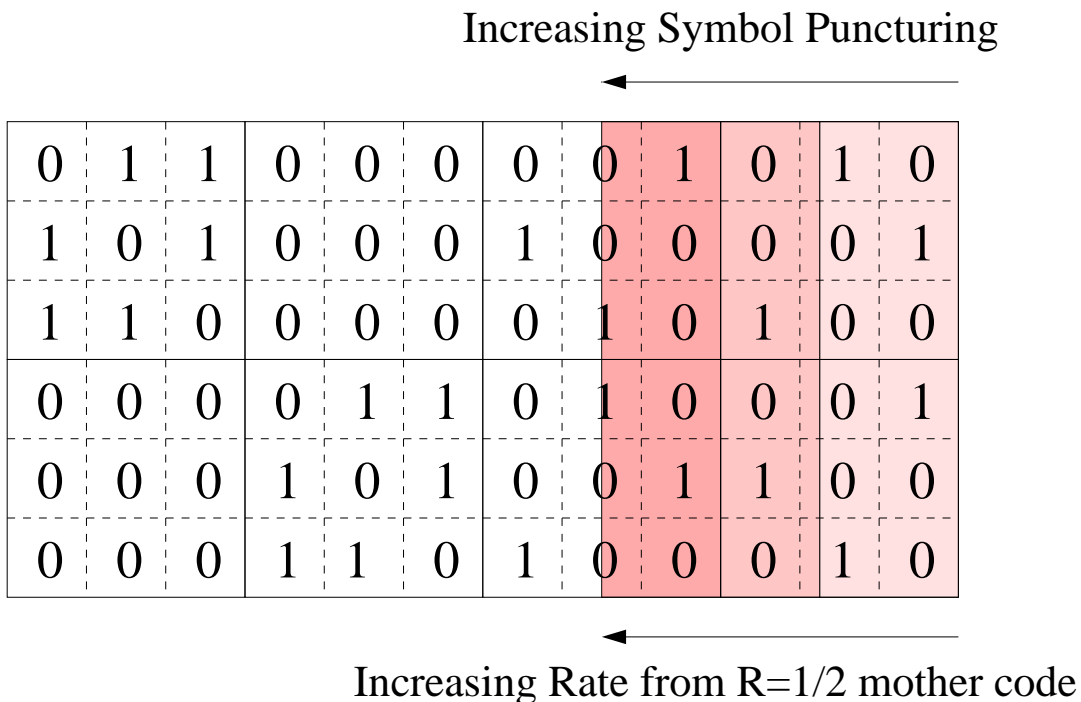


Figure 10. Puncturing pattern/scheme for the $R = 1/2$ proposed protograph.

VIII. PERFORMANCE RESULTS OF THE PROPOSED SYMBOLWISE PUNCTURING PATTERNS

We have made extensive simulations in order to compare the performance of the different schemes proposed in this deliverable, based on a mother matrix which has been designed with the hardware constraints necessary for practical implementation (cf. deliverable D4.5).

The simulations are presented in figure 11. For the rates $R = 0.545$, $R = 0.666$, $R = 0.75$ and $R = 0.833$, the optimal bitwise puncturing has been implemented and compared with the simplified symbolwise puncturing presented in the last section. For each case $R = 0.666$, $R = 0.75$ and $R = 0.833$, we have indicated the performance of a non-binary LDPC code which has been optimized for the specific rate, thereby denoted *stand alone optimized code*. The gap between the performance of the rate-compatible scheme and the performance of the stand alone optimized code therefore quantify the optimality of the puncturing mode and pattern. The mother code used for the rate-compatible scheme is a rate $R = \frac{1}{2}$ DAVINCI code in GF(64), and with codeword length $N_s = 480$ symbols ($N_b = 2880$ bits). This mother code has been optimized, as well as the stand alone codes, with the advanced DAVINCI techniques presented in deliverable D4.5.

The decoder parameters are the following. Since the DAVINCI demonstrator is not yet adapted to punctured coding schemes, we have made the simulations using the non-binary Belief Propagation decoder based on multi-dimensionnal FFT presented in details in [19]. The maximum number of decoding iterations is set to 100, and we use a stopping criterion based on the syndrome verification.

First, we verify that the bitwise puncturing patterns are in all cases very close to the performance of the stand alone optimized codes, which was expected from the partial results presented in section V. The maximum performance degradation observed for the bitwise puncturing scheme is for the highest rate $R = 0.833$, and the performance degradation is less than $0.1dB$ in the waterfall region. Note also that since the mother code has rate $R = 1/2$, the number of punctured bits to go from $R = 1/2$ to $R = 0.833$ is relatively large, and this slight performance degradation is more than acceptable.

As for the symbolwise puncturing scheme, we can observe a performance loss compared to the bitwise puncturing scheme and the stand alone optimized codes. Obviously, this was expected since the proposed

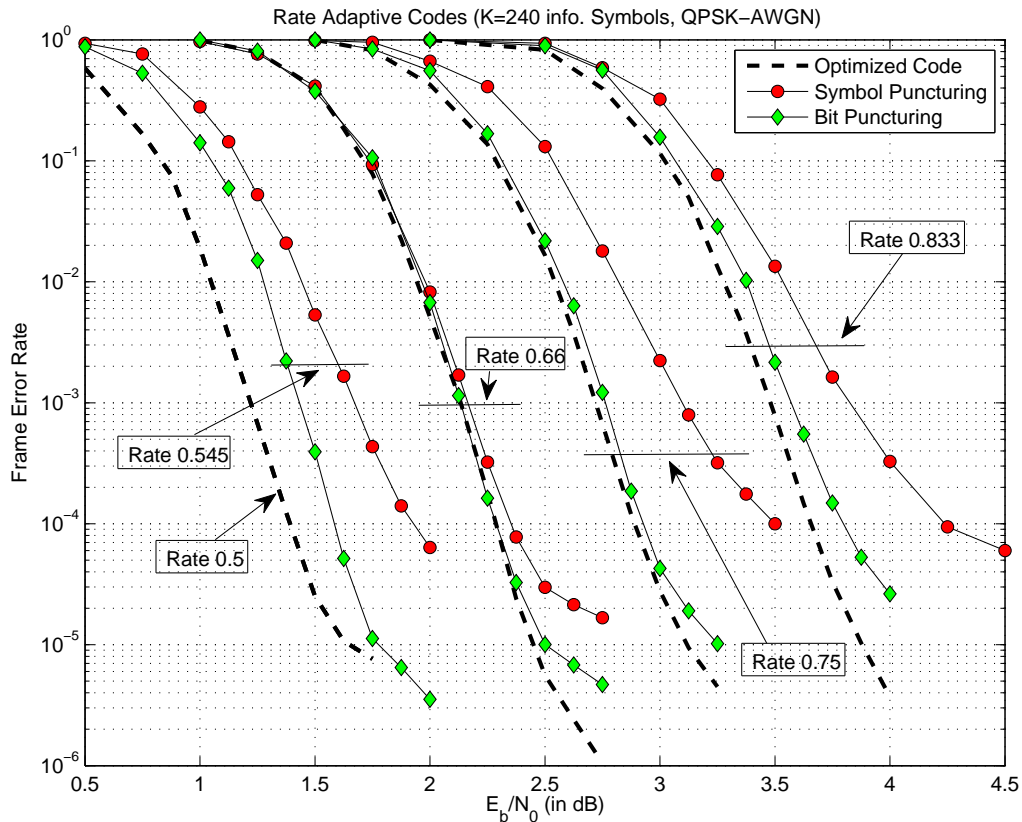


Figure 11. Frame error rate performance of non-binary rate compatible schemes proposed in the DAVINCI project.

symbolwise puncturing scheme from the protograph approach is quite simplistic. However, if we look at the performance degradation at a $\text{FER}=10^{-2}$, the gap is only of 0.2dB for the rates $R = 0.545$ and $R = 0.833$, and 0.3dB for the rate $R = 0.75$. In regard with the simplicity of implementing symbolwise puncturing compared to bitwise puncturing, this performance loss is rather small, and we expect that it will not have any impact on link-adaptation strategies.

Finally, we can note that the symbolwise puncturing scheme is optimal for the rate $R = 2/3$, which was predicted by the discussion in section VII. Actually, the joint design of the code and the puncturing pattern is specifically adapted to this rate. First, we can comment that this is a good sign that symbolwise puncturing is sufficient for getting good decoding performance. Secondly, we believe that there exist other protograph structures, which combined with a proper puncturing pattern would be closer to bitwise puncturing for other rates. We will continue until the end of the project to look for more general structures of rate compatible coding schemes.

IX. CONCLUSION

In this deliverable, we have proposed various puncturing schemes for $(2, d_c)$ non-binary LDPC codes over $GF(2^p)$ to achieve rate-compatible codes. First, we focused on optimal bitwise puncturing based on finite length criteria using the concept of *symbol recovery*. We have shown the superiority of the proposed systematic scheme in complexity and performance compared to published alternative solutions. However, the bitwise puncturing poses the problem of mapping and demapping when higher-order QAM are employed. To circumvent this problem and allow a simple demapping algorithm, we proposed a sub-optimal design of symbolwise puncturing, which is also based on *symbol recovery* criterion, which is jointly designed with the Tanner graph structure of the mother code. We have shown that although sub-optimal, the simple symbolwise puncturing scheme should be sufficient to have no impact on the global throughput obtained with the proposed H-ARQ techniques at the link adaptation level.

Several remaining issues could be investigated in the future, some of them within the DAVINCI project:

- 1) explore the possibility to use more than a single mother code, i.e. $k_m \in \{2, 3\}$ mother codes, and optimize the structure of the k_m mother codes to disjoint rate ranges.
- 2) consider shortening/pruning strategies as well as puncturing strategies. Shortening reduces the rate of the transmission while puncturing increases the rate. The issue behind using shortening is that it is not possible anymore to have constant information blocklength with rate adaptation, which is not desirable for many transmission schemes.
- 3) design other types of protographs which are closer to the optimal performance for rates different than $R = 2/3$ with symbolwise puncturing.

REFERENCES

- [1] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. 36, pp. 389-400, Apr. 1988.
- [2] J. Ha, J. Kim, and S. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2824-2836, Nov. 2004.
- [3] H. Pishro-Nik and F. Fekri, "Results on punctured LDPC codes," in *Proc. IEEE Information Theory Workshop*, San Antonio, TX, Oct. 2004.
- [4] H. Pishro-Nik and F. Fekri, "Results on punctured low-density parity-check codes and improved iterative decoding," *IEEE Trans. Inf. Theory*, Vol. 53, No. 2, pp. 599-614, February 2007.
- [5] I. Andriyanova and R. Urbanke, "Waterfall region performance of punctured LDPC codes over the BEC," in *Proc. ISIT'09*, Soul, Korea, June-July 2009, 2644-2648.
- [6] 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.
- [7] B. N. Vellambi and F. Fekri, "Finite-length rate-compatible LDPC codes: a novel puncturing scheme," *IEEE Trans. Commun.* vol. 57, no. 2, pp. 297-301, Feb. 2009.
- [8] G. Yue, X. Wang, and M. Madhian, "Design of rate-compatible irregular repeat accumulate codes," *IEEE Trans. Commun.* vol. 55, no. 6, pp. 1153-1153, June 2007.
- [9] M. C. Davey and D. MacKay, "Low-density parity-check codes over $GF(q)$," *IEEE Commun. Lett.*, vol. 2, pp. 165-167, June 1998.
- [10] X.-Y. Hu and E. Eleftheriou, "Binary representation of cycle Tanner-graph $GF(2^q)$ codes", in *Proc. IEEE International Conference on Communications (ICC'2004)*, Paris, France, June 2004.
- [11] X.-Y. Hu, E. Eleftheriou and D.M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386-398, January 2005.
- [12] A. Venkiah, D. Declercq and C. Poulliat, "Design of cages with a randomized progressive edge growth algorithm", *IEEE Commun. Lett.*, vol. 12, no. 4, pp. 301-303, April 2008.
- [13] C. Poulliat, M. Fossorier and D. Declercq, "Design of regular $(2, d_c)$ -LDPC codes over $GF(q)$ using their binary images", *IEEE Trans. Commun.*, vol. 56, no. 10, pp. 1626-1635, Oct. 2008.
- [14] DaVinci project, FP7 INFOS-ICT-216203, D2.2.1 v1.0, Link Level Evaluation, issue 1, <http://www.ict-davinci-codes.eu/>
- [15] D. Klinc, J. Ha, and S. W. McLaughlin, "On rate-adaptivity of nonbinary LDPC codes," in *Proc. Int. Conf. on Turbo Codes and Related Topics*, Sept. 2008.
- [16] D. Klinc, J. Ha, and S. W. McLaughlin, "Optimized puncturing and shortening distributions for nonbinary LDPC codes over the binary erasure channel," in *Proc. Allerton Conf. on Commun., Control, and Computing*, Sept. 2008.
- [17] L. Barnault and D. Declercq, "Fast Decoding Algorithm for LDPC over $GF(2^q)$," in *Proc. IEEE Information Theory Workshop*, Paris, France, 2003.
- [18] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over $GF(q)$," in *Proc. IEEE International Conference on Communications*, Paris, France, pp. 772-776, June 2004.
- [19] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over $GF(q)$," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633-643, 2007.
- [20] A. Goupil, M. Colas, G. Gelle and D. Declercq, "FFT-based BP decoding of general LDPC codes over Abelian groups," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 644-649, April 2007.
- [21] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low complexity, low memory EMS algorithm for non-binary LDPC codes," in *Proc. IEEE International Conference on Communications (ICC'07)*, pp. 671-676, Glasgow, UK, June 2007.
- [22] A. Voicila, D. Declercq, F. Verdier, M. Fossorier and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," accepted in *IEEE Trans. Commun.*, 2009.
- [23] T. Richardson and R. Urbanke, "Efficient Encoding of Low-Density Parity-Check Codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 638-656, Feb. 2001.
- [24] J. Kim, A. Ramamoorthy, and S. W. McLaughlin, "Design of efficiently encodable rate-compatible irregular LDPC codes," in *Proc. IEEE International Conference on Communications*, Istanbul, Turkey, June 2006, pp. 1131-1136.
- [25] J. Kim, A. Ramamoorthy, and S. W. McLaughlin, "Design of efficiently-encodable rate-compatible irregular LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 2, pp. 365-375, Feb. 2009.
- [26] T. Richardson and R. Urbanke, "Multi-edge type LDPC codes," <http://lthcwwww.epfl.ch/>
- [27] W. Chen, L. Yin and J. Lu, "Efficient encoding of cycle codes on graphs with large girths," in *Proc. 2008 IEEE International Conference on Communications, Circuits and Systems (ICCCAS'08)*, Xiamen, China, May 2008, pp. 11-15.
- [28] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*. New York: Macmillan Ltd. Press, 1976.
- [29] J. Huang and J. Zhu, "Linear time encoding of cycle $GF(2^p)$ codes through graph analysis," *IEEE Commun. Lett.*, vol. 10, pp. 369-371, May 2006.
- [30] J. Huang, S. Zhou, and P. Willett, "Nonbinary LDPC coding for multicarrier underwater acoustic communication," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 9, pp. 1684-1696, Dec. 2008
- [31] W. Chen, C. Poulliat, D. Declercq, et al, "Structured high-girth non-binary cycle codes," submitted to APCC'2009.
- [32] D. Duyck, J. Boutros and M. Moeneclaey, "Low-density graph codes for slow fading relay channels," arXiv:0903.1502 (March 2009).