# Untainted Puncturing for Irregular Low-Density Parity-Check Codes

David Elkouss, Jesus Martinez-Mateo, and Vicente Martin

*Abstract*—Puncturing is a well-known coding technique widely used for constructing rate-compatible codes. In this paper, we consider the problem of puncturing low-density parity-check codes and propose a new algorithm for intentional puncturing. The algorithm is based on the puncturing of *untainted* symbols, i.e. nodes with no punctured symbols within their neighboring set. It is shown that the algorithm proposed here performs better than previous proposals for a range of coding rates and short proportions of punctured symbols.

*Index Terms*—Low-density parity-check codes, intentional puncturing, short-length codes.

## I. INTRODUCTION

LOW-DENSITY parity-check (LDPC) codes are considered *fixed-rate* channel codes since they incorporate a fixed amount of redundant information [1]. However, there exist some well-known techniques for adapting the coding rate of a linear code, one of them is *puncturing*. A linear code is punctured by deleting a set of symbols in a codeword.

When puncturing a code, we must differentiate between random and intentional puncturing. In the former, punctured symbols are randomly chosen, whereas in the latter, the code is analyzed to select the set of symbols to puncture. The asymptotic performance of random and intentional punctured LDPC codes was analyzed in [2], [3], and puncturing thresholds were also identified in [4]. Some other methods delved into the code structure to identify good puncturing patterns [5], [6], or examined its graph construction [7], [8] to facilitate its puncturing [9]–[12].

The objectives pursued when switching from random puncturing can lead to different solutions. In particular, algorithms that focus in covering a wide range of coding rates do not offer the best performance for small puncturing proportions and vice-versa. Several algorithms find puncturing patterns that allow to cover a wide range of rates [5], [6], [13]–[15]. However, when working with short length codes, but also in other scenarios, the ensemble of punctured symbols determines the decoding performance. In this work, we describe an algorithm that we call *untainted*. Its main focus is optimizing the decoding of moderately punctured codes.

The rest of this paper is organized as follows. In Section II, we introduce the notation, some puncturing properties and the untainted algorithm. In Section III, we present simulation results over several channels, and compare them with the results in previous studies [5], [6]. Conclusions are presented in Section IV.

## II. UNTAINTED PUNCTURING

### A. Notation and Definitions

Error correcting codes can be represented by bipartite graphs linking symbol nodes with check nodes. Let[1] $\mathcal{N}(z)$ denote the neighborhood of a node $z$, that is, the set of nodes adjacent to $z$. The degree of a node is defined as the cardinality of its neighborhood. This concept can be extended to include all the nodes reachable from $z$ by traversing a maximum of $k$ edges, we call this set of nodes $\mathcal{N}^k(z)$ the neighborhood of depth $k$ of $z$.

Let $\mathcal{P}$ stand for the set of punctured symbols, $v \in \mathcal{P}$ belongs to the set of 1-step extended recoverable symbols ($v \in \mathcal{R}_1$) if $\exists c \in \mathcal{N}(v)$ such that $\forall w \in \mathcal{N}(c) \backslash \{v\}, w \notin \mathcal{P}$. For $n > 1$ we recursively define the sets of $k$-step extended recoverable symbols $R_k$. We say that a punctured symbol $v \notin \mathcal{R}_1 \cup ... \cup \mathcal{R}_{k-1}$ belongs to the set of $k$-step extended recoverable ($v \in \mathcal{R}_k$) symbols if $\exists c \in \mathcal{N}(v)$ and $\exists w \in \mathcal{N}(c) \backslash \{v\}$ such that $w \in \mathcal{R}_{k-1}$ and $\forall w' \in \mathcal{N}(c) \backslash \{v, w\}, w' \in \mathcal{P} \Rightarrow w' \in \mathcal{R}_1 \cup ... \cup \mathcal{R}_{k-1}$.

The graph subjacent to $\mathcal{N}^{2k}(v)$, $v \in \mathcal{R}_k$, is assumed to be tree-like. Let $u$ be a node in $\mathcal{N}^{2k}(v)$. We denote by $\mathcal{N}_\downarrow{}^t(u)$ the neighborhood of $u$ of depth $t$ restricted to the descendants of $u$ in this tree. Note that $\mathcal{N}_\downarrow(u) = \mathcal{N}_\downarrow{}^1(u)$. We can prune it by eliminating the connection of any symbol $w \in \mathcal{R}_l$ with a check $c \in \mathcal{N}_\downarrow(w)$ if $\max_{w' \in \mathcal{N}_\downarrow(c)} \{m | w' \in \mathcal{R}_m\} > l - 1$. We call this graph $\mathcal{T}_v$ the *extended recovery tree* of $v$.

We consider in this letter the sum-product decoding algorithm. The algorithm exchanges messages representing probabilities or the log-likelihood ratio (LLR) of probabilities. The messages are iteratively exchanged from symbol to check nodes and from check to symbol nodes. If the decoding graph is tree-like then, for uniform sources and output symmetric channels, sum-product decoding is equivalent to maximum a posteriori decoding and the decoder minimizes the decoding error [16].

Let us analyze the effect of puncturing on the messages exchanged in the sum-product algorithm. For every $v \in \mathcal{P}$ the decoding algorithm has no information on the value that $v$ takes. In consequence, for binary input channels it can take both values (one and zero) with probability one half. On the LLR version of the sum-product algorithm the outgoing messages from symbol $v$ on iteration one are equal to zero.

A check $c \in \mathcal{N}(v)$ in the neighborhood of a punctured symbol $v$ is called a survived check node if $\exists w \in \mathcal{N}_\downarrow(c) | w \in \mathcal{R}_{k-1}$, and a dead check node otherwise. The message that a dead check node $c$ sends to $v$ is a zero LLR. A punctured symbol is recovered when it receives a message from a

survived check node. It follows from the previous definitions that a symbol in $\mathcal{R}_k$ is recovered after $k$ decoding iterations.

We define the extended recovery error probability of a punctured symbol $P_e(v)$ as the decoding error probability of $v \in \mathcal{R}_k$ after $k$ iterations on $\mathcal{T}_v$. Assuming that a codeword is sent through a binary input memoryless output symmetric channel the probability of error is independent of the codeword sent [16]. Then $P_e(v)$ is the probability that $v$ takes the value one conditional to sending the all zero codeword.

These concepts, are extended in the sense that they generalize the definitions introduced by Ha *et al.* in [5]. The difference between both sets of definitions is that every punctured symbol is connected to only one survived check node in the (non-extended) recovery tree in [5] and, in consequence, the (non-extended) recovery error probability is given by the probability that the message sent by the survived node is wrong.

### B. Properties of the extended recovery error probability

Ha considered the exact recovery error probability over the (non-extended) recovery tree for the binary erasure channel (BEC), the additive white Gaussian noise (AWGN) channel and the binary symmetric channel (BSC). This error probability is a monotone increasing function on the number of nodes in the recovery tree. The algorithm in [5] was developed to exploit this non-intuitive property.

The single survived check node assumption captures the tree structure when a high proportion of symbols are punctured. However, for a low proportion of punctured symbols there can be more than one survived check node. We now show that having more than one survived check node is a desirable property. More precisely, adding a survived check node in an extended recovery tree can not increase $P_e(v)$. We first prove a stronger claim on the BEC, i.e. adding a survived check node decreases $P_e(v)$. Then, we prove the property for general symmetric channels. The idea behind the general proof is that we can reduce the number of survived check nodes by adding noise to the symbol nodes under a survived check node. Then, given that the sum-product algorithm on tree like graphs with uniform priors is equivalent to a maximum a posteriori estimation, the decoding on the noisier tree can not reduce the decoding error probability.

*Theorem 1:* Let $l, k \in \mathbb{N}$ and $0 \le l < k$. Now consider the subgraph of a check node $z$ of depth $2k - 2l - 1$ such that $\max_{w \in \mathcal{N}(z)} \{m | w \in \mathcal{R}_m\} = k - l$. Let $\mathcal{T}_{v_1}, \mathcal{T}_{v_2}$ be the extended recovery trees associated with punctured symbols $v_1, v_2 \in \mathcal{R}_k$. Let both trees be identical except for some $x \in \mathcal{N}^{2l}(v_2)$ that is linked with $z$. Then the recovery error probability of $v_1$ and $v_2$ sent through a BEC($\alpha$), with $0 < \alpha < 1$, verify:

$$P_e(v_1) > P_e(v_2). \tag{1}$$

*Proof:* The initial erasure probability of a symbol $v$ is:

$$\epsilon_v^{(0)} = \begin{cases} 1 & \text{if } v \in \mathcal{P} \\ \alpha & \text{otherwise} \end{cases}$$

further, $\epsilon$ can be recursively defined for any symbol and check in the tree from its children erasure probabilities:

$$\epsilon_v = \epsilon_v^{(0)} \prod_{c \in \mathcal{N}_\downarrow(v)} \epsilon_c \tag{2}$$

$$\epsilon_c = 1 - \prod_{v \in \mathcal{N}_\downarrow(c)} (1 - \epsilon_v). \tag{3}$$

Now, taking into account that there are no punctured symbols within the leave nodes by the definition of the extended recovery tree, it holds that $\epsilon_v^{(0)} < 1$ for the leaf symbols of the tree spanning from check $z$. It follows by induction that: 1) $\forall v, c \in \mathcal{N}_\downarrow^{2k-2l-1}(z), \epsilon_v, \epsilon_c < 1$; which implies that $\epsilon_z < 1$, and 2) if we attach a check node $z$ with $\epsilon_z < 1$ to a symbol $x$ then $\epsilon_{v_1} > \epsilon_{v_2}$. Finally, the recovery error probability of a symbol node $v$ is $P_e(v) = \epsilon_v/2$, which completes the proof. ∎

*Theorem 2:* The recovery error probability of $v_1$ and $v_2$ over the trees $\mathcal{T}_{v_1}, \mathcal{T}_{v_2}$ defined exactly as in Th. 1 and sent through any binary input symmetric output memoryless channel $C$, verify:

$$P_e(v_1) \ge P_e(v_2). \tag{4}$$

*Proof:* For a precise characterization of $P_e$ in the general setting, we need to track a message density instead of a scalar. The initial density function of a non-recovered punctured node takes the form of the Dirac delta function $D(y) = \delta(y)$, since a punctured node transmits a zero LLR with probability one. Let the remaining nodes in $\mathcal{T}_{v_1}, \mathcal{T}_{v_2}$ have initial densities given by $P_0(y)$, the initial LLR density associated with channel $C$.

Now consider a second scenario for $\mathcal{T}_{v_2}$. We associate every leaf node in $\mathcal{N}_\downarrow^{2k-2l-1}(z)$ with samples from $D(y)$, which is equivalent to puncturing these nodes. If we puncture the leave nodes, their parent check nodes do not become survived check nodes and the symbols $w \in \mathcal{N}_\downarrow^{2k-2l-3}(z) | w \in \mathcal{P}$ are not recovered. It follows by induction that $z$ remains a dead check node, i.e. associating the leaves with $D(y)$ is equivalent to eliminating the edge joining $z$ to $x$. In consequence, the density of messages reaching the root node $v_2$ in the second scenario is identical to the density of messages reaching $v_1$ in the first scenario.

The (binary output) degenerate channel $D$, with initial density $D(y)$, transforms any input into a one or a zero with equal probability, i.e. $p_D(1|x) = p_D(0|x) = 0.5$. Let $Q$ represent the concatenated channel of $C$ with $D$:

$$\begin{aligned} p_Q(y'|x) &= \sum_{y \in \mathcal{Y}} p_D(y'|y) p_C(y|x) \\ &= 0.5 \sum_{y \in \mathcal{Y}} p_C(y|x) = p_D(y'|x). \end{aligned} \tag{5}$$

In other words, $D$ can be regarded as the concatenation of $C$ with itself, and the samples from $D(y)$ are stochastically degraded samples of $P_0(y)$ [17].

Following the argument in [18, Th. 5], Eq. (5) implies that $P_e(v_1) \ge P_e(v_2)$. The assertion follows from the fact that the estimate of both $v_1$ and $v_2$ are maximum likelihood estimates. ∎

## C. Untainted Puncturing Algorithm Description

We introduce the concept of *untainted*, to propose a simple method that chooses symbols such that all the check nodes of a selected symbol are survived nodes. This restriction guarantees that more than one survived check node is associated with every punctured symbol.

*Definition 1:* A symbol node $v$ is said to be *untainted* if there are no punctured symbols within $\mathcal{N}^2(v)$.

Let $\mathcal{X}_\infty$ be the set of untainted symbol nodes. Initially, when there are no punctured symbols, $\mathcal{X}_\infty$ consists of every symbol node.

> {Initialize} $\mathcal{X}_\infty = \{1, ..., n\}$, $p = 1$.
> **while** $\mathcal{X}_\infty \neq \emptyset$ **do**
>     {*Step 1.– Look for candidates*}
>     Make the set of candidates $\Omega$, which is a subset of $\mathcal{X}_\infty$, such that $u \in \Omega$ if $|\mathcal{N}^2(u)| \leq |\mathcal{N}^2(v)|$ for any $v \in \mathcal{X}_\infty$.
>     {*Step 2.– Select for puncturing*}
>     Pick a symbol node $v^{(p)}$ from $\Omega$ (pick one randomly if there exist more that one symbols in $\Omega$).
>     {*Step 3.– Update the set of untainted symbols*}
>     $\mathcal{X}_\infty = \mathcal{X}_\infty \backslash \mathcal{N}^2(v)$
>     $p = p + 1$
> **end while**

The algorithm obtains a set of puncturable symbol nodes consisting in the symbols selected in the second step. It concludes when $\mathcal{X}_\infty = \emptyset$, i.e. there is no untainted symbols. The range of values for $p$, the number of punctured symbols, can be found empirically by simulations (see Table I).

Note that for codes with an almost regular check node degree distribution, the searching criterion in Step 1 can be simplified: instead of looking for a symbol with the smallest neighboring set of depth 2, the algorithm can look for symbols with the lowest degree.

## III. SIMULATION RESULTS

The untainted algorithm ensures that the extended recovery tree of a punctured symbol has more than one survived check node. In this section, we construct codes to show that the untainted algorithm yields a better performance in terms of the frame error rate (FER). We have constructed $10^4$ bit-long irregular LDPC codes of different coding rates for the BEC, the BSC and the AWGN channels. The polynomials for the BSC with rates $(0.5, 0.6, 0.7, 0.8)$ have been drawn from [19]. The remaining generating polynomials, as well as all the matrices used can be checked in [20].

Figs. 1, 2 and 3 compare the performance of intentional punctured codes using the untainted algorithm with those in [5] and in [6] for different proportions of punctured symbols $\pi$. All codes are decoded with 200 iterations using the sum-product algorithm [16] over the graph of the mother (non-punctured) code.

As in Ha *et al.* [5] we used three random seeds for the simulations of each intentional puncturing algorithm, the results in the figures show the intermediate performer.

The untainted punctured LDPC codes outperform the codes punctured following the algorithms in [5] and in [6] for all the coding rates, channels and puncturing proportions considered.
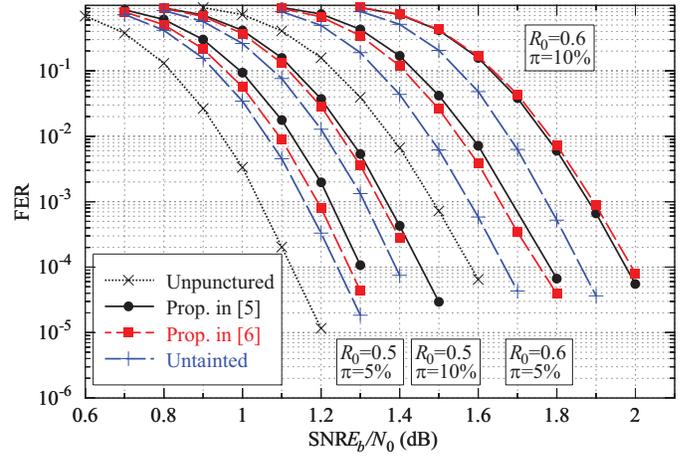


Fig. 1. FER over the AWGN as a function of the signal-to-noise ratio (SNR). Two LDPC codes with coding rates $R_0 = 0.5$ and $R_0 = 0.6$, and two different proportions of punctured symbols, $\pi = 5\%$ and $\pi = 10\%$ were used.
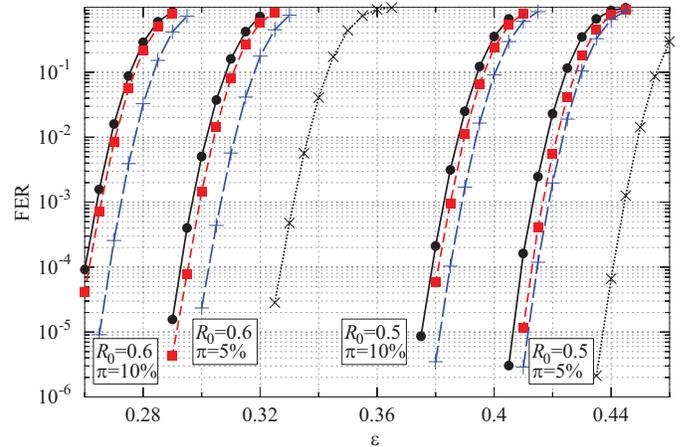


Fig. 2. FER over the BEC with crossover probability $\varepsilon$ for different intentional puncturing strategies. Two LDPC codes with coding rates $R_0 = 0.5$ and $R_0 = 0.6$, and two different proportions of punctured symbols, $\pi = 5\%$ and $\pi = 10\%$ were used. See inset in Fig. 1 for symbols and line-styles.

For a FER of $10^{-3}$ the strongest improvements appear in the BSC for a mother code of rate $R_0 = 0.3$ punctured a 10%, in the BEC for a mother code of rate $R_0 = 0.6$ punctured a 5% and in the AWGN for a mother code of rate $R_0 = 0.6$ punctured a 10% respectively.

Table I shows $p_{\min}$ and $p_{\max}$, the minimum and maximum values of $p$, respectively, after $5 \cdot 10^3$ algorithm runs. Sizes are computed for both the untainted algorithm and the algorithm in [5]. The values are computed for the same codes used in Fig. 3 and two additional codes of rates 0.7 and 0.8. The table shows that the untainted algorithm can puncture a smaller number of symbols compared to [5]. This behavior is consistent with the additional number of survived check nodes required by the untainted algorithm.

## IV. CONCLUSIONS

We proved that having more than one survived check node in the extended recovery tree is a desirable property. The untainted algorithm is a method that chooses symbols such that all the check nodes of a selected symbol are survived
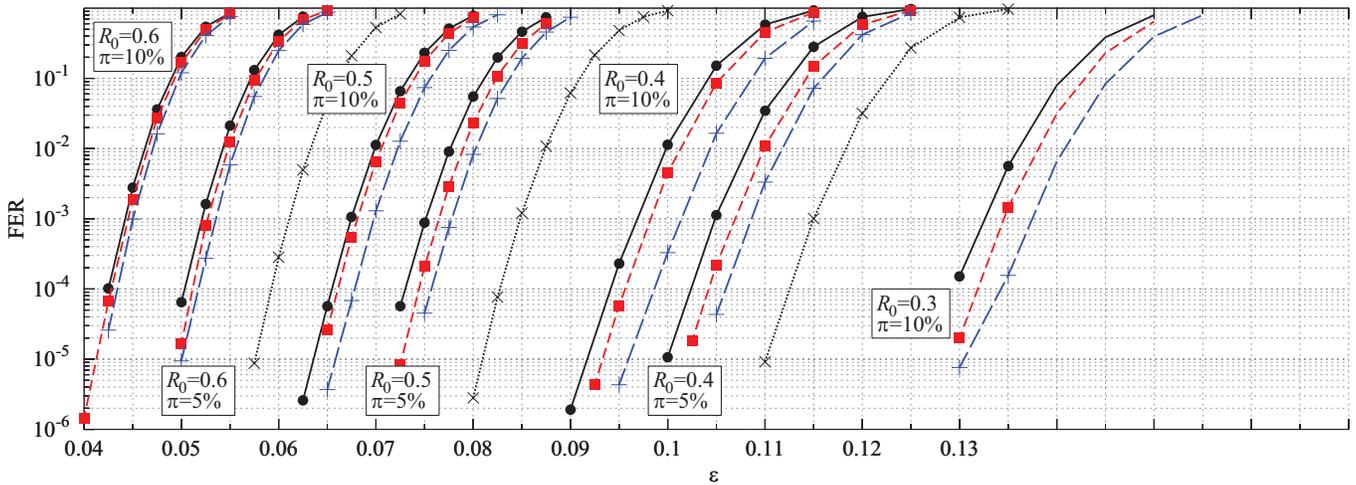
Fig. 3.    FER over the BSC with crossover a probability $\varepsilon$ for several LDPC codes with coding rates $R_0 = 0.3$, $R_0 = 0.4$, $R_0 = 0.5$ and $R_0 = 0.6$, and two different proportions of punctured symbols, $\pi = 5\%$ and $\pi = 10\%$.

TABLE I
NUMBER OF PUNCTURED SYMBOLS

|  |  | Coding rate (mother code) | | | | | |
|---|---|---|---|---|---|---|---|
|  |  | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 |
| Ref. [5] | $p_{min}$ | 4628 | 4031 | 3439 | 2866 | 2258 | 1581 |
|  | $p_{max}$ | 4753 | 4139 | 3552 | 2983 | 2353 | 1657 |
| Untainted | $p_{min}$ | 2603 | 2286 | 1909 | 1587 | 1212 | 851 |
|  | $p_{max}$ | 2686 | 2374 | 1987 | 1655 | 1273 | 901 |

check nodes. Furthermore, the algorithm can be implemented with a low computational cost.

Simulation results show that the performance of the untainted algorithm in terms of the FER is better than the best intentional puncturing algorithms in the literature for a range of coding rates, channels and puncturing proportions. The drawback of using this algorithm is a reduction in the maximum achievable rate since the proportion of puncturable symbols is limited by the untainted criterion.

## REFERENCES

[1] N. Bonello, S. Chen, and L. Hanzo, "Low-density parity-check codes and their rateless relatives," *IEEE Commun. Surv. Tutor.*, vol. 13, no. 1, pp. 3–26, 2011.
[2] J. Ha, J. Kim, and S. W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 11, pp. 2824–2836, Nov. 2004.
[3] C.-H. Hsu and A. Anastasopoulos, "Capacity achieving LDPC codes through puncturing," *IEEE Trans. Inf. Theory*, vol. 54, no. 10, pp. 4698–4706, Oct. 2008.
[4] H. Pishro-Nik and F. Fekri, "Results on punctured low-density parity-check codes and improved iterative decoding techniques," *IEEE Trans. Inf. Theory*, vol. 53, no. 2, pp. 599–614, Feb. 2007.

[5] 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.
[6] 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.
[7] X.-Y. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.
[8] S. Bandi, V. Tralli, A. Conti, and M. Nonato, "On girth conditioning for low-density parity-check codes," *IEEE Trans. Commun.*, vol. 59, no. 2, pp. 357–362, Feb. 2011.
[9] M. Yazdani and A. Banihashemi, "On construction of rate-compatible low-density parity-check codes," *IEEE Commun. Lett.*, vol. 8, no. 3, pp. 159–161, Mar. 2004.
[10] T. Tian and C. R. Jones, "Construction of rate-compatible LDPC codes utilizing information shortening and parity puncturing," *EURASIP J. Wireless Commun. Netw.*, vol. 2005, no. 5, pp. 789–795, Oct. 2005.
[11] J. Kim, A. Ramamoorthy, and S. Mclaughlin, "The design of efficiently-encodable rate-compatible LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 2, pp. 365–375, Feb. 2009.
[12] C. Shi and A. Ramamoorthy, "Design and analysis of E2RC codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 889–898, Aug. 2009.
[13] J. Ha, D. Klinc, J. Kwon, and S. W. McLaughlin, "Layered BP decoding for rate-compatible punctured LDPC codes," *IEEE Commun. Lett.*, vol. 11, no. 5, pp. 440–442, May 2007.
[14] H. Y. Park, J. W. Kang, K. S. Kim, and K. C. Whang, "Efficient puncturing method for rate-compatible low-density parity-check codes," *IEEE Trans. Wireless Commun.*, vol. 6, no. 11, pp. 3914–3919, Nov. 2007.
[15] M. El-Khamy, J. Hou, and N. Bhushan, "Design of rate-compatible structured LDPC codes for hybrid ARQ applications," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 965–973, Aug. 2009.
[16] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.
[17] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, 1991.
[18] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.
[19] J. Martinez-Mateo, D. Elkouss, and V. Martin, "Blind reconciliation," *Quantum Inf. and Computation*, vol. 12, no. 9 & 10, pp. 791–812, 2012.
[20] Generating polynomials and matrices used in this paper. Available: http://gcc.ls.fi.upm.es/en/codes.html

[2]http://www.cesvima.upm.es