Optimized codes for the binary coded side-information problem

Anne Savard, Claudio Weidmann

Optimized Codes for the Binary Coded Side-Information Problem

Anne Savard and Claudio Weidmann
ETIS/ CNRS UMR 8051 - ENSEA - Université de Cergy-Pontoise
95014 Cergy-Pontoise, France
Email: {anne.savard, claudio.weidmann}@ensea.fr

***Abstract*—This paper analyzes a practical scheme for the binary coded side-information problem based on LDPC codes and trellis quantization. A recently proposed improved decoder is shown to be amenable to numerical density evolution and thus to LDPC code optimization. First results display significant gains compared to off-the-shelf codes, which could be further improved by refined modeling of the system.**

I. Coded side information

Source coding with coded side-information is a classic problem of information theory in which two encoders E_X and E_Y separately encode two discrete correlated sources X and Y, with joint distribution $P_{X,Y}$, at rates R_X and R_Y, respectively. The decoder D_X tries to reconstruct X losslessly as $\hat{X}$, using a compressed version of Y as side-information (SI), while Y will not be reconstructed. The situation is depicted in Fig. 1. This problem is paradigmatic for distributed source coding in e.g. sensor networks, where a helper node (relay) has access to a noisy version of the source.

Ahlswede and Körner characterized the achievable rate region in [1]. A rate pair (R_X, R_Y) is achievable if

$$\begin{cases} R_X \geq H(X|U) \\ R_Y \geq I(Y;U) \end{cases} \tag{1}$$

for an auxiliary random variable $U \in \mathcal{U}$, with $|\mathcal{U}| \leq |\mathcal{Y}| + 2$, such that $X - Y - U$ form a Markov chain.

When both X and Y are binary symmetric, Gu et al. [2] showed that encoding Y using binary quantization with Hamming distortion criterion achieves the aforementioned rate region, for which a closed-form expression can be obtained. Let X be a Bernoulli-1/2 source. The correlation between X and Y is modeled by a Binary Symmetric Channel (BSC) with error probability p (BSC-p). The encoder E_Y produces a compressed version of Y and can be modeled, in an idealized setting, by a BSC-D, where D is the optimal distortion obtained at rate R_Y. This is depicted in Fig. 2.

The achievable rate region (1) becomes :

$$\begin{cases} R_X \geq H(p + D - 2pD) \\ R_Y \geq 1 - H(D) \end{cases} \tag{2}$$

When considering this idealized setup, one notices that it is equivalent to a standard binary Slepian-Wolf problem [3] with a BSC-ϵ correlation channel, where $\epsilon = (1-p)D + (1-D)p$ is the error probability of the BSC equivalent to the concatenation of the two BSCs in Fig. 2 and X is encoded at rate $H(X|\hat{Y})$. An easy encoding and decoding procedure for all points of the Slepian-Wolf rate region using linear codes was described by Gehrig and Dragotti in [4]. A practical solution using Low-Density Parity Check (LDPC) codes was proposed by Liveris et al. in [5].

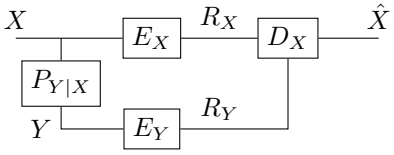

Fig. 1. Coded side information problem: general case

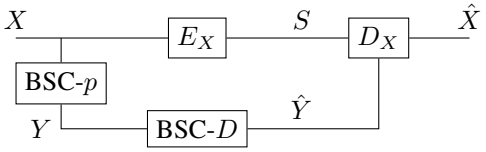

Fig. 2. Coded side information problem: binary case

In our previous work [6], we proposed the following simple scheme for the coded side-information problem based on the above LDPC approach and a trellis quantizer. The binary input sequence $X = (X_1, X_2, \ldots, X_n)$ of length n is compressed by computing its syndrome $S = XH^T$, where $H \in GF(2)^{(n-k)\times n}$ is the parity check matrix of an LDPC code. The encoder E_Y produces a quantized version W of Y using a binary trellis-coded quantizer based on a convolutional code. The codeword $\hat{Y}$ that is closest in Hamming metric to the input sequence Y is found using the Viterbi algorithm and the corresponding information sequence is output as index W. The reconstruction $\hat{Y}(W)$ associated to the index W is used as channel information at the LDPC decoder D_X. This decoder must then estimate the sequence X from the syndrome S and the reconstructed SI $\hat{Y}(W)$, as in [5].

The following notations will be used throughout this paper:

- $\hat{y}_i(w)$, $i = 1, \ldots, n$: current component values of $\hat{Y}(w)$
- s_j, $j = 1, \ldots, n-k$: components of the realization of syndrome $S = XH^T$
- LLR_i: Log-Likelihood Ratio (LLR) corresponding to channel value $\hat{y}_i(w)$

- $m_{j,i}^{c\to v}$: LLR message from check node (CN) c_j to variable node (VN) v_i
- $m_{i,j}^{v\to c}$: LLR message from VN v_i to CN c_j
- $\mathcal{N}(v)$: set of CNs connected to VN v
- $\mathcal{N}(c)$: set of VNs connected to CN c

The LDPC decoder performs MAP decoding by computing the *a posteriori probabilities* (APP) for each variable node and deciding for the value $\hat{x}_i$ that maximizes this quantity. The steps of the Belief Propagation decoder (BP) are:

- Initialization :

$$LLR_i = \log\left(\frac{P(X_i = 0|\hat{y}_i(w))}{P(X_i = 1|\hat{y}_i(w))}\right) \tag{3}$$

$$= (1 - 2\hat{y}_i(w)) \log\frac{1-\epsilon}{\epsilon} \tag{4}$$

- Data pass :

$$m_{i,j}^{v\to c} = LLR_i + \sum_{c_{j'} \in \mathcal{N}(v_i)\setminus c_j} m_{j',i}^{c\to v} \tag{5}$$

- Check pass :

$$m_{j,i}^{c\to v} = 2tanh^{-1}\left((1-2s_j)\prod_{v_{i'}\in\mathcal{N}(c_j)\setminus v_i} \tanh\left(\frac{m_{i',j}^{v\to c}}{2}\right)\right) \tag{6}$$

- Decision :

$$\hat{x}_i = \begin{cases} 0, \text{ if } LLR_i + \sum_{c_j\in\mathcal{N}(v_i)} m_{j,i}^{c\to v} \geq 0 \\ 1, \text{ else} \end{cases} \tag{7}$$

The next section briefly recalls the improved decoder for this scheme that we proposed in [6]. The remaining sections then present numerical density evolution of this decoder and first code optimization results.

Our main contributions in this paper are:

- adapting numerical density evolution along the same lines as [7] to the improved decoder presented in [6].
- using this density evolution in a differential evolution algorithm in order to optimize the degree distribution of irregular LDPC codes.

II. Improved Decoder [6]

A. Principle

The method relies on the following observation: the coded SI is given by a single index w, but many SI sequences are mapped to the same index by the quantizer. The sequences mapped onto the index w form the Voronoi cell $\mathcal{V}_w = \{y \in \{0,1\}^n | E_Y(y) = w\}$. Instead of using only $\hat{Y}(w)$ to decode X, our method exploits the knowledge of $\mathcal{V}_w$ to select another representative sequence. The hope is that this new sequence is closer to X than $\hat{Y}(w)$ and thus accelerates iterative decoding. The new sequence will be obtained by projecting an approximate solution $\hat{x}^{(t)}$ onto $\mathcal{V}_w$. Thus, we need a characterization of the Voronoi cells. Since convolutional codes are linear, we only need to characterize the Voronoi cell $\mathcal{V}_0$ associated to the all-zero codeword; other cells are obtained as $\mathcal{V}_w = \mathcal{V}_0 \oplus \hat{Y}(w)$, where $\oplus$ denotes component-wise modulo-2 addition.

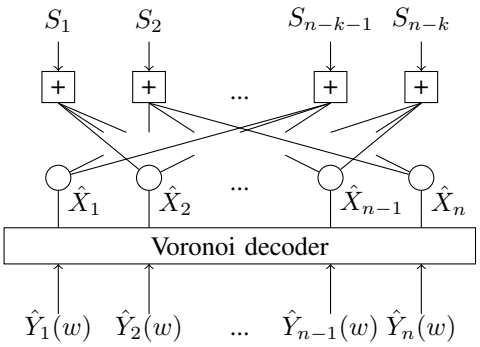

Fig. 3. Proposed decoder graph

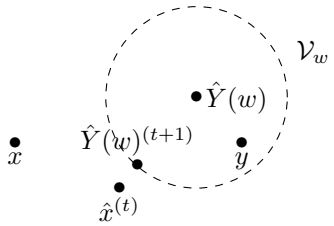

Fig. 4. Geometrical intuition

The projection of the sequence $\hat{x}^{(t)}$ onto the Voronoi cell $\mathcal{V}_w$ yields the sequence $\hat{Y}(w)^{(t+1)}$ closest to $\hat{x}^{(t)}$. Thanks to linearity, we can project the sequence $\hat{x}^{(t)} \oplus \hat{Y}(w)$ onto $\mathcal{V}_0$, yielding an output sequence v^* such that $\hat{Y}(w)^{(t+1)} = v^* \oplus \hat{Y}(w)$. Calderbank et al. showed in [8] that $\mathcal{V}_0$ can be characterized by the evolution of a particular finite state machine (FSM). Thus the above *Voronoi projection* step can be implemented using the hard- or soft-input Viterbi and BCJR algorithms. In [6], we found that a BCJR Voronoi decoder gives the best performances.

B. Decoding procedure

Decoding of X starts with T BP LDPC decoder iterations as in the standard setup. If the decoder fails to converge after T iterations, the SI is modified by performing a Voronoi projection. The resulting sequence is used to modify the input LLRs before carrying on with additional BP iterations (the $m^{c\to v}$ and $m^{v\to c}$ messages are not reset). If the decoder still fails to converge after t additional decoding iterations, the above procedure is restarted.

Fig. 3 depicts the decoder graph, where the "Voronoi decoder" is responsible for modifying the channel LLR values fed to the LDPC decoder shown above it. The geometrical intuition behind this decoding principle is depicted in Fig. 4.

C. BCJR Voronoi decoder

The inputs to the BCJR Voronoi decoder are scaled versions $\tilde{z}_i$ of the extrinsic $z_i = \sum_{c_j\in\mathcal{N}(v_i)} m_{j,i}^{c\to v}$ computed by the BP

LDPC decoder,

$$LLR_i^{(\text{BCJR})} = \begin{cases} (1-0.99^j)z_i, & \text{if } \hat{Y}_i(w)=0, \\ -(1-0.99^j)z_i, & \text{if } \hat{Y}_i(w)=1, \end{cases} \quad (8)$$

where $j-1$ is the number of times a Voronoi projection has already been performed. This heuristic scaling takes into account the reliability of the extrinsic z and is based on the intuition that the more projections (and decoder iterations) have been made, the closer $\hat{x}^n$ should be to the source sequence. So for the first projections, the sequence projected onto the Voronoi cell is very close to the all-zero codeword, thus we don't change the LLRs too aggressively, since the LDPC decoder would have difficulties to correct them, while in later iterations this shouldn't be a problem anymore. The factor 0.99 was chosen based on the good performances obtained in our simulation setup, it may be changed for other setups.

The BCJR then computes the bit-wise APP values, i.e. the marginal posterior likelihood of bit Y_i given the quantizer index w and a soft estimate of the source. From these we obtain an extrinsic soft-output LLR sequence $extr$.

Since the BCJR yields an APP for Y and since $X = Y \oplus Z$, with Z a binary Bernoulli-p source, we can use the tanh-rule to compute the new "channel" LLR fed to the LDPC decoder for the next t iterations:

$$LLR_i = 2tanh^{-1}\left(\tanh\left(\frac{extr_i}{2}\right)\tanh\left(\frac{\log\left(\frac{1-p}{p}\right)}{2}\right)\right) \quad (9)$$

The BCJR Voronoi decoder has the major advantage that it can be used to perform a numerical density evolution of the overall decoder, following the approach in [7]. Using this density evolution, a differential evolution algorithm can then optimize the LDPC degree distribution for a given quantizer, improving even more the performances obtained in [6].

In the next two sections, we present standard density evolution and propose a modified version suited to our Voronoi decoder, as well as differential evolution. Using these two tools, we optimize the degree distributions of a rate-1/2 LDPC code for the binary coded side-information problem.

III. Density evolution and decoding threshold

For any irregular LDPC code, one defines two polynomials to describe the distribution of the edges as follows:

$$\lambda(x) = \sum_{i=1}^{V_{max}} \lambda_i x^{i-1} \text{ and } \rho(x) = \sum_{i=1}^{C_{max}} \rho_i x^{i-1}$$

where V_{max} and C_{max} are the maximal variable node and check node degrees, respectively. Given the Tanner graph of the LDPC code, λ_i represents the fraction of edges that are connected to a variable node of degree i (i.e. connected to i check nodes). Similarly, ρ_i is the fraction of edges that are connected to a check node of degree i.

Density evolution, which has been proposed by Richardson and Urbanke in [9], is a tool to determine the asymptotic ensemble average behavior of LDPC codes given channel model and noise power. This algorithm tracks the average probability density function (pdf) of the messages from variable nodes to check nodes for each iteration of the BP algorithm.

We can define the error probability at iteration l of the BP algorithm as:

$$P_e^{(l)} = \int_{-\infty}^{0} f_v^{(l+1)}(x)dx$$

where $f_v^{(l+1)}(x)$ is the pdf of the correct message from a variable node to a check node during the l-th iteration of BP algorithm. Density evolution computes $f_v^{(l+1)}(x)$ from $f_v^{(l)}(x)$. Using this tool, we can determine the decoding thresholds of a LDPC code and also optimize its degree distributions using e.g. differential evolution as explained in the next section.

The decoding threshold is defined as the maximum noise power (here: crossover probability p) such that the error probability vanishes as l grows large, $\lim_{l\to\infty} P_e^{(l)} = 0$.

Density evolution for the improved decoder follows along the lines of [7], where the authors used a numerical density evolution step using a BCJR algorithm over a binary inter-symbol interference channels.

The following notations will be used:

- $f_v^{(l)}$: pdf of message from a VN to a CN at lth iteration.
- $f_c^{(l)}$: pdf of message from a CN to a VN at lth iteration.
- $f_o^{(l)}$: pdf of a priori LLR at the lth iteration.
- $f_e^{(l)}$: pdf of extrinsic given to the BCJR at lth iteration.

Since the path from X to $\hat{Y}$ is modeled by a BSC of error probability ϵ, the initialization of f_o is given by

$$f_o^{(1)} = \epsilon\delta\left(x+\log\left(\frac{1-\epsilon}{\epsilon}\right)\right)+(1-\epsilon)\delta\left(x-\log\left(\frac{1-\epsilon}{\epsilon}\right)\right),$$

where $\delta(x)$ is the Dirac distribution.

The average density $f_v^{(l+1)}$ is given by

$$f_v^{(l+1)} = f_o^{(l+1)} \otimes \left[\sum_{i=1}^{V_{max}} \lambda_i \left(\bigotimes_{k=1}^{i-1} f_c^{(l)}\right)\right] \quad (10)$$

where $\otimes$ stands for the convolution operation and $\bigotimes_{k=1}^{i-1}$ for the convolution of $i-1$ pdfs.

In the same way, the average $f_c^{(l+1)}$ is given by

$$f_c^{(l+1)} = \sum_{i=1}^{C_{max}} \rho_i \Gamma^{-1}\left[\bigotimes_{k=1}^{i-1} \Gamma\left(f_v^{(l+1)}\right)\right], \quad (11)$$

where Γ is the density transformation operator induced by $\gamma : x \to (\text{sgn}(x), -\log\tanh|x/2|)$.

Similarly, the average density of the extrinsic given to the BCJR is obtained by

$$f_e^{(l+1)} = \sum_{i=1}^{V_{max}} \tilde{\lambda}_i \left(\bigotimes_{k=1}^{i} f_c^{(l)}\right) \quad (12)$$

where $\tilde{\lambda}_i = \lambda_i / \left(i\int_0^1 \lambda(x)dx\right)$ is the fraction of variable nodes of degree i. (We still need to take into account the reliability scaling before performing the BCJR step.)

The Voronoi decoder transforms $f_e^{(l+1)}$ into $f_o^{(l+1)}$ as follows:

$$f_o^{(l+1)} = \begin{cases} \epsilon_{trellis}(f_e^{(l)}, p), & \text{if } l = T + kt, \quad k \in \mathbb{N}, \\ f_o^{(l)}, & \text{else,} \end{cases} \tag{13}$$

where $\epsilon_{trellis}$ is a symbolic notation for trellis evolution. Since there isn't a closed-form expression for this evolution, it will be computed numerically using Monte-Carlo techniques.

The density evolution associated to the proposed decoder is described in Algorithm 1. The operation in line 2, corresponding to standard density evolution, is described in Algorithm 2 and the operation in line 5, corresponding to the Monte Carlo simulation, is detailed in Algorithm 3.

Algorithm 1 Density evolution with BCJR Voronoi decoder

1: Initialization:
$f_o^{(1)} = \epsilon\delta\big(x + \log\left(\frac{1-\epsilon}{\epsilon}\right)\big) + (1-\epsilon)\delta\big(x - \log\left(\frac{1-\epsilon}{\epsilon}\right)\big)$
$P_e^{(1)} = 1$;
2: Density evolution: Phase 1 $iter \leftarrow T$
3: **for** $i \rightarrow proj$ **do**
4: $f_e^{(l+1)} = \sum_{i=1}^{V_{max}} \tilde{\lambda}_i \left(\bigotimes_{k=1}^{i} f_c^{(l)}\right)$
5: Trellis evolution: $f_o^{(l+1)} = \epsilon_{trellis}(f_e^{(l)}, p)$
6: Density evolution: Phase 2 $iter \leftarrow t$
7: **end for**

Algorithm 2 Density evolution

1: **for** $l \rightarrow iter$ **do**
2: $f_v^{(l+1)} = f_o^{(l+1)} \otimes \left[\sum_{i=1}^{V_{max}} \lambda_i \left(\bigotimes_{k=1}^{i-1} f_c^{(l)}\right)\right]$
3: $f_c^{(l+1)} = \sum_{i=1}^{C_{max}} \rho_i \Gamma^{-1}\left[\bigotimes_{k=1}^{i-1} \Gamma\left(f_v^{(l+1)}\right)\right]$
4: $P_e^{(l)} = \int_{-\infty}^{0} f_v^{(l+1)}(x)dx$
5: **if** $P_e^{(l)} \geq P_e^{(l-1)}$ **then**
6: **break**
7: **end if**
8: **end for**

Algorithm 3 Trellis evolution

1: Draw an i.i.d. vector V of distribution $f_o^{(l+1)}$
2: $V \leftarrow V \times \tilde{X} \times \tilde{Y}$ where $\tilde{X} \in \{-1, 1\}^n$ and $\tilde{Y} = 1 - 2\hat{Y}(w)$
3: Take into account the reliability of the estimate
4: $V_1 \leftarrow$ BCJR-based Voronoi decoder
5: $V_1 \leftarrow V_1\tilde{Y}$
6: $LLR_i = 2\tanh^{-1}\left(\tanh\left(\frac{V_{1_i}}{2}\right)\tanh\left(\frac{\log\left(\frac{1-p}{p}\right)}{2}\right)\right)$
7: $\widetilde{LLR} \leftarrow LLR \times \tilde{X}$
8: $f_o^{(l+1)} \leftarrow$ Density of $\widetilde{LLR}$

Since the projection step has to be performed for an i.i.d. binary sequence to obtain proper averages (see [7] for more details), we first draw a sequence X and compute the corresponding quantization index. Then we use the BCJR Voronoi projection method to compute a realization of an extrinsic vector. The block length n has to be rather large in order to avoid trellis boundary effects.

Using this numerical density evolution, we obtained the following decoding thresholds for the scheme in [6], which uses an off-the-shelf LDPC code optimized for the BSC from [10]: $p^* = 0.0337$ for the standard method and $p^* = 0.0368$ for the proposed decoder. These values match well with the performances observed on Fig. 8 in [6].

IV. Differential evolution for degree optimization

Differential evolution is an iterative method for global nonlinear optimization problems [11]. It has been successfully applied to the optimization of irregular LDPC degree distributions in many works, e.g. [10] for binary output-symmetric channels.

Each iteration l of this algorithm generates NP possible degree distributions, denoted by $\{\pi_i^{(l)}\}_{i=1}^{NP}$. Here π stands for a vector grouping a non-redundant set of coefficients of $\lambda(x)$ and $\rho(x)$ satisfying the design rate and other constraints, e.g. concentrated check degrees. The first generation, $\{\pi_i^{(0)}\}_{i=1}^{NP}$, is initialized uniformly at random.

At each round, we choose within the NP candidates the one that gives the smallest error probability after performing a fixed number of density evolution steps; this distribution is denoted by $\pi_{best}^{(l)}$. For each of the NP distributions, we generate a mutant distribution by randomly selecting four distinct distributions indexed $i1, i2, i3, i4$ and computing

$$\tilde{\pi}_i^{(l+1)} = \pi_{best}^{(l)} + F(\pi_{i1}^{(l)} + \pi_{i2}^{(l)} - \pi_{i3}^{(l)} - \pi_{i4}^{(l)}),$$

where $F > 0$ is a non-negative differential mixing parameter. The NP distributions for the next round are taken to be either the distribution from the previous round or the mutant candidate $\tilde{\pi}_i$, if the latter yields a smaller error probability during the density evolution phase.

The computationally most expensive step is the BCJR. We observed that faster convergence is obtained by allowing for more density evolution iterations (including more BCJR evaluations), rather than increasing the generation size NP (which linearly increases the number of BCJR evaluations).

For the following simulation results, we used a quantizer built with a rate-5/6 convolutional code from [12] with generator matrix

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 \\ 2 & 2 & 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 2 & 3 \end{bmatrix}. \tag{14}$$

Using differential evolution, we found a rate-1/2 LDPC code ensemble with variable degree distribution

$$\begin{aligned} \lambda(x) = {} & 0.094167x^2 + 0.7275x^3 + 0.0125x^5 + 0.045x^6 \\ & + 0.00417x^{10} + 0.0317x^{14} + 0.0233x^{15} \\ & + 0.000833x^{16} + 0.04583x^{19} + 0.015x^{20} \end{aligned}$$

and concentrated check-node degrees. Actual parity check matrices were obtained using the PEG algorithm. The decoding threshold for this ensemble is $p^* = 0.06$.

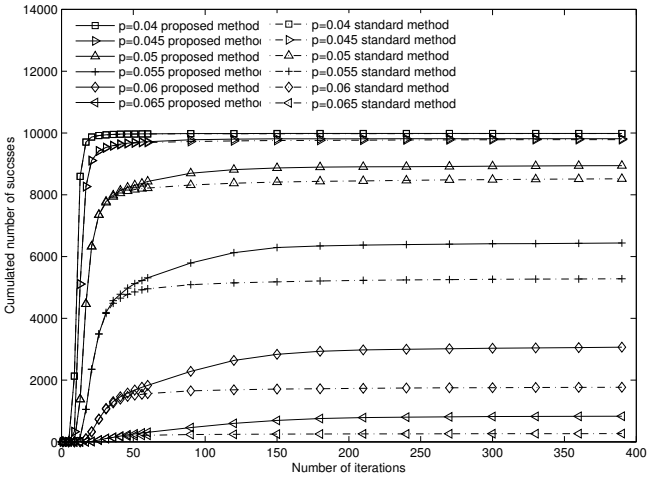

Fig. 5. Comparison between standard and proposed decoder using an optimized rate-1/2 LDPC code, a rate-5/6 trellis quantizer and the BCJR Voronoi decoder: Cumulated number of decoding successes vs. number of iterations.

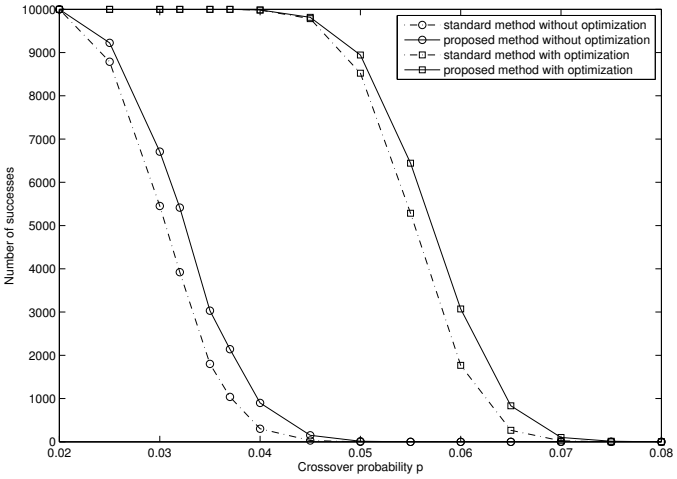

Fig. 6. Comparison between standard and proposed decoder using rate-1/2 LDPC codes, a rate-5/6 trellis quantizer, and 400 decoding iterations: Number of decoding successes vs. source crossover probability p. Curves on the left are for an off-the-shelf code (optimized for the BSC), curves on the right for a code optimized with the proposed method.

For our simulations, we set $n = 1200$ and first perform $T = 20$ LDPC decoding iterations with the LLR associated with $\hat{Y}(w)$. In case of decoding failure, we then perform up to 76 BCJR runs, after each of which we perform up to $t = 5$ LDPC decoding iterations. The results for 10000 samples are given in Fig. 5 and Fig. 6. We can again notice that our method gives clearly better results than the standard one, even for optimized codes. As in [6], we notice that the decoding threshold has been shifted. A possible explanation of this gain is that the Voronoi decoder is able to compensate part of the quantizer suboptimality.

For comparison purposes, we computed the theoretical thresholds (Shannon limits) for this setup. An ideal rate-5/6 quantizer has theoretical distortion $D^* = 0.0246$, which yields $p^*_{th} = 0.0898$. The actual distortion of our considered convolutional quantizer is $D = 0.0373$, yielding $p_{th} = 0.0786$. For now, the best rate-1/2 distributions that we found have a threshold of $p = 0.065$, which is still far from the theoretical limits. An explanation for this gap may lie in the heuristic scaling rule (8) used before the projection. Thus it is likely that density evolution won't be able to close this gap. To overcome this problem, we are investigating a refined model of the processing chain $X - Y - \hat{Y}(W)$, which should allow to use the Voronoi FSM to directly estimate X, without having to estimate the reliability of the projection step.

V. Conclusion

This work introduced an analysis and optimization framework for the practical scheme for source coding with coded side-information presented in [6]. The scheme consists of low-complexity encoders (LDPC syndrome and trellis quantizer) and an improved decoder that exploits the knowledge of the quantizer Voronoi cell $\mathcal{V}_0$ to accelerate the main decoder, which operates in Slepian-Wolf fashion. The key decoding step involves a BCJR projection onto $\mathcal{V}_0$, which cannot be modeled in closed form. However, we found that numerical density evolution using Monte Carlo simulations is able to provide precise estimates of the decoding thresholds. In a second step, we used this density evolution in a differential evolution algorithm to optimize irregular LDPC codes. The results show impressive gains compared to codes optimized for the BSC, but fall short of theoretical limits, because of heuristic approximations employed in the decoder.

References

[1] R. Ahlswede and J. Körner, "Source coding with side information and a converse for degraded broadcast channels," *IEEE Trans. Inform. Theory*, vol. 21, pp. 629–637, 1975.

[2] W. Gu, R. Koetter, M. Effros, and T. Ho, "On source coding with coded side information for a binary source with binary side information," in *Proc. ISIT*, Nice, France, June 24–29 2007, pp. 1456–1460.

[3] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inform. Theory*, vol. 19, no. 4, pp. 471–480, July 1973.

[4] N. Gehrig and P. L. Dragotti, "Symmetric and asymmetric Slepian-Wolf codes with systematic and non-systematic linear codes," *IEEE Commun. Lett.*, vol. 9, pp. 2297–2301, 2005.

[5] A. D. Liveris, Z. Xiong, and C. N. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," *IEEE Communications Letters*, vol. 6, pp. 440–442, 2002.

[6] A. Savard and C. Weidmann, "Improved decoding for binary source coding with coded side information," in *Proc. IEEE Information Theory Workshop (ITW)*, Seville, Spain, Sep. 2013.

[7] A. Kavcic, X. Ma, and M. Mitzenmacher, "Binary intersymbol interference channel: Gallager codes, density evolution, and code performance bounds," *IEEE Trans. Inform. Theory*, vol. 49, pp. 1636–1652, 2003.

[8] A. R. Calderbank, P. C. Fishburn, and A. Rabinovich, "Covering properties of convolutional codes and associated lattices," *IEEE Trans. Inform. Theory*, vol. 41, pp. 732–746, 1995.

[9] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Transactions Inform. Theory*, vol. 47, pp. 599–618, 2001.

[10] ——, "Design of capacity-approaching irregular low-density parity check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, 2001.

[11] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, pp. 341–359, 1997.

[12] H.-H. Tang and M.-C. Lin, "On $(n, n-1)$ convolutional codes with low trellis complexity," *IEEE Trans. Commun.*, vol. 50, pp. 37–47, 2002.