Improved decoding for binary source coding with coded side information

Anne Savard and Claudio Weidmann
Equipes Traitement de l'Information et Systèmes (ETIS)
CNRS UMR 8051 / ENSEA / Université de Cergy-Pontoise
95014 Cergy-Pontoise, France
Email: (anne.savard, claudio.weidmann)@ensea.fr

***Abstract*—This paper presents a new iterative decoding algorithm for the *source coding with coded side information* problem. Side information (SI) is compressed to an index by a many-to-one (quantization) function. Instead of using the reconstruction corresponding to the quantization index as a single representative SI word to aid the main decoder, one can modify it by projecting an intermediate estimate of the source word onto the Voronoi cell associated to the SI index. The hope is that the projection brings the representative SI word closer to the source word, and thus accelerates iterative decoding. Simulations using LDPC syndrome coding in the main branch and trellis-coded quantization in the SI branch show that for a fixed number of decoder iterations, this method indeed increases the number of correctly decoded source words. In fact, the decoding threshold is shifted, which may be attributed to a partial compensation of the suboptimality of the quantizer.**

I. Coded side information

In a variety of applications, such as distributed video coding or networks with relays, the decoder may have access to coded side information (SI). In practice this SI, which is correlated with the message to be decoded, is compressed using a deterministic many-to-one function. Thus the compressor output may be mapped back to a *set* of possible sequences.

This paper considers the problem of source coding with coded SI, which can be seen as a typical example of such a situation. Two discrete sources X and Y, with finite alphabets $\mathcal{X}$ and $\mathcal{Y}$, respectively, and joint distribution $P_{X,Y}$, are separately encoded by encoders E_X and E_Y at rates R_X and R_Y, respectively. The decoder D_X tries to reconstruct X losslessly as $\hat{X}$, while Y serves only as SI (for decoding X) and is not reconstructed. This situation is depicted in Fig. 1.

The achievable rate region for this problem has been characterized by Ahlswede and Körner 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.

For the case when both X and Y are binary symmetric, Gu et al. [2] showed that encoding Y using binary quantization with Hamming distortion criterion is sufficient to achieve the rate region. In this case, one can give a closed-form expression of the achievable rate region. Let X a binary 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 quantized version of Y and can be represented, in an idealized setting, by a BSC-D, where D is the optimal distortion obtained for rate R_Y. This is depicted in Fig. 2.

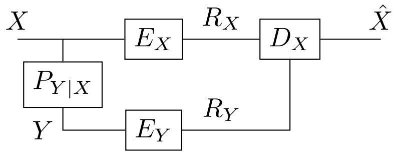

Fig. 1. Coded side information problem: general case

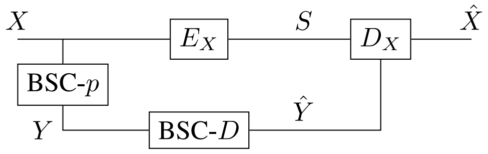

Fig. 2. Coded side information problem: binary case

The achievable rate region (1) becomes :

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

A straightforward, albeit slightly naive, "standard" implementation of the binary case may be outlined as follows. The encoder E_Y is modeled as ideal, see Fig. 2, yielding a concatenation of two BSCs that is equivalent to a BSC with error probability $\epsilon = (1-p)D + (1-D)p$. This situation corresponds to a classic binary Slepian-Wolf problem [3], where the correlation channel is a BSC-ϵ and X is encoded at rate $H(X|\hat{Y})$. A practical solution of this problem using Low-Density Parity Check (LDPC) codes has been proposed e.g. by Liveris et al. in [4]. An easy encoding and decoding procedure for all points of the rate region using linear codes has been described by Gehrig et al. [5].

The binary input sequence $X^n = (X_1, X_2, \ldots, X_n)$ of length n, is compressed by computing its syndrome $S^{n-k} = X^n H^T$, where $H \in GF(2)^{(n-k)\times n}$ is the parity check matrix of an LDPC code. The encoder E_Y produces a compressed version W of Y^n. This operation will be performed using a binary trellis-coded quantizer based on a convolutional code and the Viterbi algorithm. Such quantizers still offer an excellent performance-complexity trade off. The Viterbi algorithm is used to find the codeword $\hat{Y}^n$ closest in Hamming metric to the input sequence Y^n; the corresponding information sequence is output as index W.

The reconstruction $\hat{Y}^n(W)$ associated to the index W will be used as channel information at the LDPC decoder D_X. As in [4], the decoder must estimate the sequence X^n from the syndrome and the reconstructed SI $\hat{Y}^n$.

The following notations will be used:

- $\hat{y}_i(w)$, $i = 1, \dots, n$ are the current component values of $\hat{Y}^n(w)$
- s_j, $j = 1, \dots, n{-}k$ is the j-th component of realization of the syndrome $S^{n-k} = X^n H^T$
- LLR_i is the Log-Likelihood Ratio (LLR) corresponding to the channel message $\hat{y}_i(w)$
- $m_{j,i}^{c \to v}$ is the LLR message from check node c_j to variable node v_i
- $m_{i,j}^{v \to c}$ is the LLR message from variable node v_i to check node c_j
- $\mathcal{N}(v)$ is the set of check nodes connected to variable node v
- $\mathcal{N}(c)$ is the set of variable nodes connected to check node 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.

- 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 :

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

- Check pass :

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

- Decision :

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

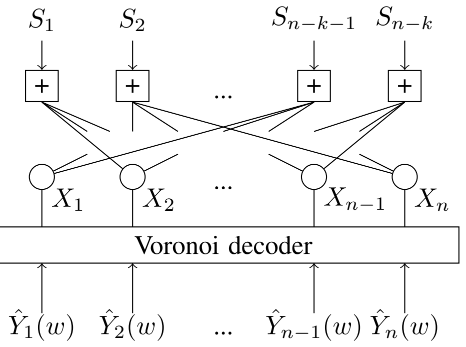

Fig. 3. Proposed decoder graph

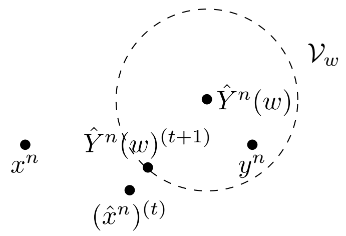

Fig. 4. Geometrical intuition

II. Proposed method

Our method relies on the following observation: the coded SI is conveyed by a single index w, but the quantizer maps many SI sequences to the same index. This set of sequences corresponds to the Voronoi cell $\mathcal{V}_w = \{y^n \in \{0,1\}^n | E_Y(y^n) = w\}$. As convolutional codes are linear, $\mathcal{V}_w = \mathcal{V}_0 \oplus \hat{Y}^n(w)$, with $\mathcal{V}_0$ the Voronoi cell associated with the origin (all-zero codeword). ($\oplus$ denotes componentwise modulo-2 addition.) Thus we will need to characterize the set $\mathcal{V}_0$ in order to perform a projection.

The projection of the sequence $(\hat{x}^n)^{(t)}$ onto the Voronoi cell $\mathcal{V}_w$ yields the sequence $\hat{Y}^n(w)^{(t+1)} \in \mathcal{V}_w$ closest to $(\hat{x}^n)^{(t)}$. Thanks to linearity, we can project the sequence $(\hat{x}^n)^{(t)} \oplus \hat{Y}^n(w)$ onto the Voronoi cell $\mathcal{V}_0$, yielding an output sequence v^* such that $\hat{Y}^n(w)^{(t+1)} = v^* \oplus \hat{Y}^n(w)$.

As in the standard setup, we start decoding X^n with T belief propagation decoding iterations. If the decoder fails to converge after T iterations, we modify the posterior probabilities of the channel information (i.e. the coded SI) by performing a projection onto the Voronoi cell associated to the received index w. 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.

A. Characterization of the Voronoi cell $\mathcal{V}_0$

In [6], it has been shown that the Voronoi cell $\mathcal{V}_0$ of a convolutional code can be characterized by the evolution of a particular finite state machine (FSM). We will refer to this as the Voronoi FSM.

To construct the Voronoi FSM for $\mathcal{V}_0$, we study the evolution of the Viterbi algorithm (in the trellis quantizer) in terms of metric differences of the sequences mapped onto the all-zero codeword. We use the rate-1/2 convolutional code (5,7) with polynomials $(1 + D^2,\ 1 + D + D^2)$ as an example. This code has 4 states and thus each trellis section is characterized by a 4-tuple of winning path metrics, named *metric state*. These 4-tuples can be computed using the Hamming distance between the trellis labels and the input sequence. Metric states can be translated by a constant 4-tuple such that their minimum element becomes 0. The Voronoi FSM for $\mathcal{V}_0$ is obtained by searching all transitions between metric states such that the minimum path metric is the one associated to the all-zero codeword. The number of such metric states is finite. A metric state (a, b, c, d) must satisfy the following condition to be included in the Voronoi FSM:

$$\min\{a + d_H(l_{E_1\to E_1}, s), b + d_H(l_{E_2\to E_1}, s)\} = a + d_H(l_{E_1\to E_1}, s) \quad (8)$$

where $l_{E_i\to E_j}$ is the trellis label from state E_i to state E_j and s is the corresponding segment of the input sequence. We suppose that the encoder starts from the all-zero state.

This procedure can be extended to any convolutional code. Condition (8) becomes :

$$\min_{\exists E_i\to E_1} \{M_i + d_H(l_{E_i\to E_1}, s)\} = M_1 + d_H(l_{E_1\to E_1}, s) \quad (9)$$

where M_i denotes the i^{th} component of the metric state and where $\exists E_i \to E_1$ means that there is a trellis transition between states E_i and E_1.

B. Decoding procedure

The decoder receives the syndrome s^{n-k} from encoder E_X and the index w from encoder E_Y. The LDPC decoder starts with the quantizer codeword $\hat{Y}^n(w)$. After T iterations, if it fails to converge, we compute APPs to obtain $(\hat{x}^n)^{(T)}$ and then search the closest sequence within the set $\mathcal{V}_w = \mathcal{V}_0 \oplus \hat{Y}^n(w)$:

$$\hat{Y}^n(w)^{(T+1)} = \hat{Y}^n(w) \oplus \arg\min_{v\in\mathcal{V}_0} d_H\left(v, (\hat{x}^n)^{(T)} \oplus \hat{Y}^n(w)\right) \quad (10)$$

This sequence is then used to modify the LLRs before carrying on with additional LDPC decoder iterations (the $m^{c\to v}$ and $m^{v\to c}$ messages are not reset). If after t additional decoding iterations the decoder still fails to converge, we restart the above procedure.

The next two sections discuss variants of the projection (10) and associated LLR update rules.

III. Viterbi Voronoi decoder

A low-complexity Voronoi decoder can be built using the Viterbi algorithm on the Voronoi FSM trellis to map the hard decision sequence $(\hat{x}^n)^{(T)}$ to $\hat{Y}^n(w)^{(T+1)}$ as in (10).

Based on our experiments, we propose the following LLR update rule:

$$LLR_i = \begin{cases} 0.9^j * LLR(\hat{Y}_i(w)) & \text{if } \hat{Y}_i(w) \neq \hat{Y}_i(w)^{(T+1)} \\ LLR(\hat{Y}_i(w)) & \text{else} \end{cases} \quad (11)$$

where j is the number of times we have already performed a projection.

This heuristic update rule is based on the intuition that the more projections (and decoder iterations) have been made, the closer $\hat{x}^n$ should be the source sequence. So for the first projections we do not reduce the LLRs too aggressively, since the LDPC decoder would have difficulties to correct them, while in later iterations this should be no longer a problem.

The factor 0.9 was determined experimentally to yield good results in our simulation setup; it may need to be changed for other setups.

Results obtained with a non-optimized rate $R_X = 0.1$ LDPC code of size $n = 1000$ and the rate $R_Y = 1/2$ convolutional code (5,7) are shown in Fig. 5. The sets of curves are indexed by the crossover probability p of the BSC relating X and Y. We can observe that the proposed method increases the rate of successful convergence. We perform $T = 30$ decoder iterations with the LLR associated with $\hat{Y}^n(w)$. In case of failure of the decoder, we perform up to 18 searches of $\hat{Y}^n(w)^{(T+lt+1)}$, and for each, we perform up to $t = 15$ decoder iterations. The results are given for 10000 samples.

This experimental setup was sufficient to show the feasibility of our method, but its overall performance is not overwhelming. To overcome this, one has to use optimized LDPC codes.

Since good low-rate LDPC codes are rather hard to come by, we decided to increase the rate of the encoder E_Y in order to be able to use rate-1/2 LDPC codes optimized for the BSC. Various optimized high-rate convolutional codes can be found in the literature, see for example [7], [8], [9].

For our second experiment, we chose an optimized LDPC code with variable node degrees

$$\lambda(x) = 0.24426x + 0.25907x^2 + 0.01054x^3 + 0.05510x^4 + 0.01455x^7 + 0.01275x^9 + 0.40373x^{11}$$

and concentrated check node degrees, found in [10]; parity check matrices were constructed with the PEG algorithm.

For the quantizer, we used a rate-5/6 convolutional code found in [7] with generator matrix

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 \\ 2 & 2 & 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 2 & 3 \end{bmatrix}. \quad (12)$$

This code uses 3 binary memory cells, meaning there are 8 trellis states. Increasing the encoder memory will improve the performance of the code in terms of distortion. At the same time, it will increase the number of trellis states and strongly expand the number of states in the Voronoi FSM, so we will only consider codes with 3 binary memory cells.

We can see in Fig. 6 that even with an optimized LDPC code and a high-rate convolutional code, the proposed method outperforms the standard one.

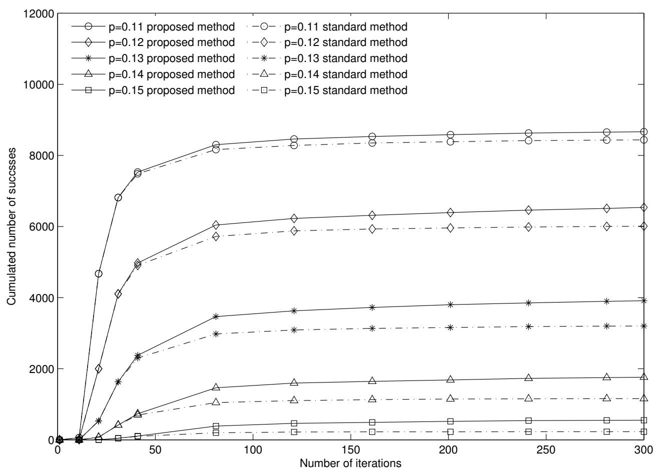

Fig. 5. Comparison between the standard and the proposed method using a rate-0.1 LDPC code, the rate-1/2 convolutional code (5,7) and the Viterbi algorithm to perform the projection: Cumulated number of decoding successes as a function of the number of iterations.

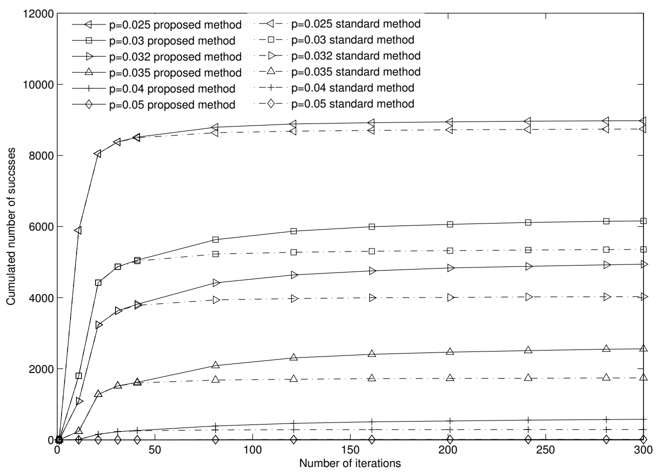

Fig. 6. Comparison between the standard and the proposed method using a rate-0.5 optimized LDPC code, a rate-5/6 convolutional code and the Viterbi algorithm to perform the projection: Cumulated number of decoding successes as a function of the number of iterations.

IV. BCJR VORONOI DECODER

The above hard-input, hard-output Viterbi Voronoi decoder has the conceptual advantage of delivering a sequence that is guaranteed to lie in $\mathcal{V}_w$ and thus to have been a possible quantizer input sequence. This is offset by the difficulty of finding an optimal LLR update rule. An obvious alternative is to use the soft-input, soft-output BCJR algorithm [11] on the Voronoi trellis. In the present setting, the BCJR takes the extrinsic $\tilde{z}$ as inputs

$$LLR_i^{\text{BCJR}} = \begin{cases} \tilde{z}_i & \text{if } \hat{Y}_i(w) = 0 \\ -\tilde{z}_i & \text{if } \hat{Y}_i(w) = 1 \end{cases} \tag{13}$$

where $\tilde{z}$ is a scaled version of the extrinsic

$$z_i = \sum_{c_j \in \mathcal{N}(v_i)} m_{j,i}^{c \to v}$$

computed by the BP decoder. This scaling is done for the same reason as for the Viterbi decoder: for the first projections, the extrinsic z isn't very reliable.

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

Since the BCJR gives us 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:

$$LLR_i = 2tanh^{-1}\left(tanh\left(\frac{extr_i}{2}\right)tanh\left(\frac{log\left(\frac{1-p}{p}\right)}{2}\right)\right) \tag{14}$$

For simulations with the BCJR, we used an optimized rate-1/2 LDPC code of length $n = 1200$ and first performed $T = 20$ decoding iterations with the LLR associated with $\hat{Y}^n(w)$. In case of failure of the decoder, we then perform up to 76 BCJR runs, after each of which we perform up to $t = 5$ LDPC decoding iterations. The results are given for 10000 samples.

As Fig. 7 shows, using a soft-input soft-output Voronoi decoder gives the best performances for the proposed method.

Since the decoding threshold is clearly seen to be shifted in Fig. 8, we computed the theoretical thresholds (Shannon limits) for comparison purposes. An ideal rate-5/6 binary code has theoretical distortion $D = 0.0246$, which yields $p_{th}^* = 0.0898$. If we consider the convolutional code (12), its actual distortion is $D = 0.0373$, yielding $p_{th} = 0.0786$. These thresholds are still far from the ones observed in our simulations, which may partly be due to the moderate LDPC block length. Nevertheless, we can see that the proposed method provides clearly better performance than the standard one. A possible explanation of this gain is that the Voronoi decoder is able to compensate part of the quantizer suboptimality.

For completeness, we also ran simulations with a soft-input Viterbi decoder; the obtained results are shown in Fig. 9 and compared to the standard and BCJR methods in Fig. 8. The gain over the standard variant is only slightly above that for hard-input Viterbi, indicating the importance of proper soft outputs.

Beyond its superior performance (at the price of complexity), the BCJR Voronoi decoder has another major interest: it can be used to perform numerical density evolution of the overall decoder, along the lines of the approach presented in [12]. This allows to match the optimization of the LDPC code to the structure of the Voronoi cell. This will also give a better insight on how the parameters that are currently chosen heuristically impact the decoding procedure.

V. CONCLUSION

We presented a new decoding principle for source coding with coded side information and compared various implementations of the proposed method with a standard setup.

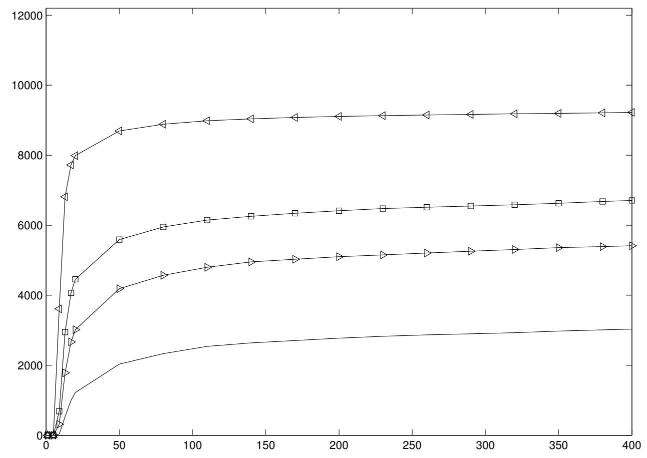

Fig. 7. Comparison between the standard and the proposed method using a rate-0.5 optimized LDPC code, a rate-5/6 convolutional code and the BCJR algorithm to perform the projection: Cumulated number of decoding successes as a function of the number of iterations.

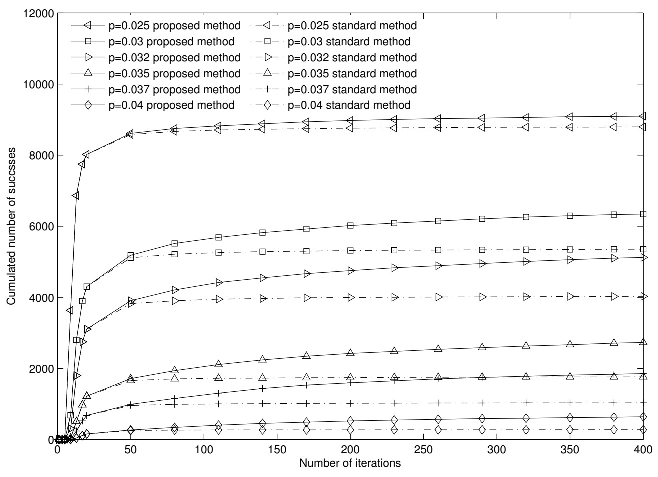

Fig. 9. Comparison between the standard and the proposed method using a rate-0.5 optimized LDPC code, a rate-5/6 convolutional code and the the soft-input Viterbi algorithm to perform the projection: Cumulated number of decoding successes as a function of the number of iterations.

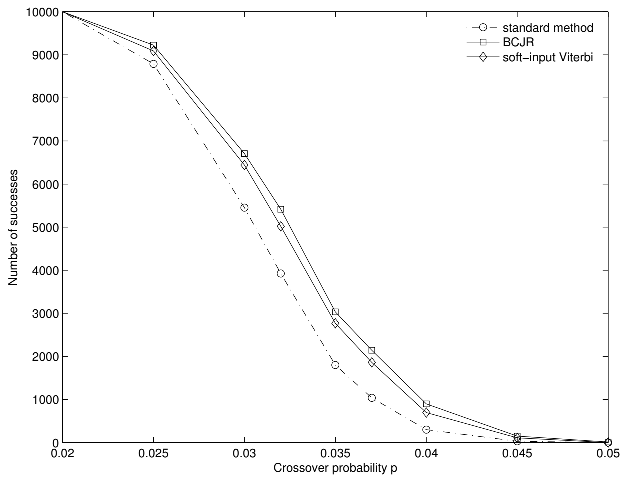

Fig. 8. Comparison between the standard and the proposed methods using a rate-0.5 optimized LDPC code, a rate-5/6 convolutional code, 400 decoding iterations: Number of decoding successes as a function of the crossover probability p.

The key idea is to exploit the knowledge of the preimage of the side information encoder output to accelerate the iterative main decoder. In the case of linear trellis-coded quantization, the preimage is congruent to the Voronoi cell $\mathcal{V}_o$, which can be described by a finite state machine. By projecting an intermediate source estimate onto this cell, one can improve the side information fed to the main decoder, which operates in standard Slepian-Wolf fashion. Using a Viterbi decoder with heuristic soft outputs as projector or a BCJR decoder yields gains in terms of decoding success vs. iteration number.

Currently, we are working on a more theoretical analysis of this decoding principle, beginning with a density evolution study of the presented example with trellis-coded quantization.

References

[1] R. Ahlswede and J. Körner, "Source coding with side information and a converse for degraded broadcast channels," *IEEE Trans. on Information 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," *ISIT 2007, Nice, France, June 24 - June 29*, pp. 1456–1460, 2007.

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

[4] 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.

[5] N. Gehrig and P. L. Dragotti, "Symetric and a-symetric Slepian-Wolf codes with systematic and non-systematic linear codes," *IEEE Communications letters*, vol. 9, pp. 2297–2301, 2005.

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

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

[8] S. Lin and D. J. Costello, *Error control coding*. Prentice-Hall, 2004.

[9] P. J. Lee, "High-rate convolutional code construction with the minimum required SNR criterion," NASA JPL, The Telecommun. and Data Acquisition Progr. Rept., Aug. 1985.

[10] T. Richardson and R. Urbanke, "Design of capacity-approaching irregular low-density parity check codes," *IEEE Trans. on Information Theory*, vol. 47, pp. 619–637, 2001.

[11] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. on Information Theory*, vol. IT-20, pp. 284–287, 1974.

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