ON DISTRIBUTED ARITHMETIC CODES AND SYNDROME BASED TURBO CODES FOR SLEPIAN-WOLF CODING OF NON UNIFORM SOURCES

V. Toto-Zarasoa[1], *E. Magli*[2], *A. Roumy*[1] *and G. Olmo*[2]

[1]INRIA, Campus Universitaire de Beaulieu, 35042 Rennes-Cedex, France
[2] Politecnico di Torino, C. Duca degli Abruzzi 24, 10129 Torino, Italy

ABSTRACT

In this paper, we consider the use of source codes and channel codes for asymmetric distributed source coding of non uniform correlated sources. In particular, we use distributed arithmetic codes as source codes and syndrome based turbo codes as channel codes. We compare the advantages and drawbacks of the two systems for different source probabilities and different compression ratio. We show that prior knowledge of the source distribution improves the performance of both approaches in terms of their distances to the Slepian-Wolf bound. Turbo codes are better when the puncturing is low, while distributed arithmetic codes are less impacted by the change of compression rate.

1. INTRODUCTION

It is well known that the minimum achievable rate of lossless compression of correlated sources is their joint entropy. It has been shown by Slepian and Wolf [1] that two separate correlated sources can be compressed at that same rate, provided that joint decoding is performed and each source is compressed at most at their conditional entropy rate. That result affected the work is several domains, including wireless sensor networks [5] or video compression [6]. A Slepian-Wolf coder can be based either on a channel code or on a source code.

Channel codes can achieve the Slepian-Wolf bound as long as they achieve the capacity of the channel that models the correlation between the sources. Wyner has indeed shown the optimality of syndrome approach [4]. The principle of syndrome based coding is to partition the space of the source words into specific bins, or cosets, containing uncorrelated words having the same "syndrome". Knowing the syndrome, the decoder searches the closest codeword to the side information. The efficiency of a code can be measured as its ability to create cosets which would be as disjoint as possible.

Here, we consider the turbo-syndrome approach proposed in [7]. Turbo codes are indeed known to be capacity achieving channel codes [3]. The syndrome trellis of a turbo code is based on its parity-check polynomials, instead of its generator polynomials. Therefore, using the syndrome approach allows us to consider more general turbo codes for the Slepian-Wolf problem, not only systematic ones as for the traditional parity approach [8]. The use of LDPC codes [18, 19] has also been considered, and also relies on the coset principle.

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement n. 216715 (NEWCOM++)

Besides techniques based on channel coding, a few authors have also investigated the use of source codes for the Slepian-Wolf problem. This is motivated by the fact that existing source coders obviously exhibit nice compression features that should be retained, such as the ability to employ flexible and adaptive probability models, and low encoding complexity. In [10] the problem of designing a variable-length distributed source code is addressed; it is shown that the problem of designing a zero-error coder is NP-hard. In [11] a similar approach is followed; the authors consider the problem of designing Huffman and arithmetic distributed codes for multilevel sources with zero or almost-zero error probability. The idea is that, if the joint density of the source and the side information satisfies certain conditions, the same codeword (or the same interval for the arithmetic coding process) can be associated to multiple symbols. This approach leads to an encoder with a complex modeling stage (NP-hard for the optimal code, though suboptimal polynomial-time algorithms are provided in [11]), while the decoding process resembles a classical arithmetic decoder. In [12] an extension of arithmetic coding (AC), named distributed arithmetic coding (DAC), has been proposed for asymmetric Slepian-Wolf coding. The idea is to perform the binary AC process in such a way as to allow the intervals of symbols "0" and "1" to overlap to some extent. This introduces an ambiguity in the description of the source, which lowers the codeword bit-rate, and requires a correlated side information signal to be resolved. Moreover, in [13] DAC has been extended to the case of symmetric distributed coding of two sources at arbitrary rates within the Slepian-Wolf rate region. A rate-compatible extension of DAC has been presented in [14]. Similar concepts have been proposed in [15], in which the interval overlap is applied to quasi-arithmetic codes, and [16], in which sources with memory are considered.

In this paper, we consider non-uniform sources, which is the case encountered in real systems such as distributed video coding systems. The authors in [17] were the first to consider the use of turbo codes for coding non uniform sources. Their system is based on syndrome formers and inverse syndrome formers and generalizes the scheme they proposed in [9] for uniform sources. We first give the Slepian-Wolf bounds for non uniform sources. We then adapt the turbo syndrome trellis to deal with the non uniformity of the sources. Rate adaptation is naturally performed via the puncturing of the syndrome bits. The turbo syndrome approach is compared with a distributed arithmetic code.

2. THEORETICAL BOUNDS

Let X and Y be two binary correlated sources, of respective probabilities $\mathbb{P}(X=1)=p_X$ and $\mathbb{P}(Y=1)=p_Y$. The correlation between X and Y is modeled as a Binary Symmetric Channel (BSC) of cross-over probability p. Consider the separate compression of X and Y at respective rates R_X and R_Y; the Slepian-Wolf theorem [1] states that they can be recovered at the decoder without error as long as the two sources are jointly decoded, and:

$$\begin{cases} R_X \geq H(X|Y) \\ R_Y \geq H(Y|X) \\ R_X + R_Y \geq H(X,Y) \end{cases}$$

where H denotes the entropy function. In the case of asymmetric DSC, one source (say Y, the side "information") is compressed at its entropy rate $R_Y = H(Y)$, and the other (X)is compressed at the conditional entropy rate $R_X = H(X|Y)$.

When the source X is uniform, i.e. $p_X = 0.5$, the source Y is also uniform, and:

$$H(X) = H(Y) = 1$$
$$H(X|Y) = H(Y|X) = H(p)$$

But, when the sources are non uniform, i.e. $p_X \neq 0.5$, $p_Y = p_X(1-p) + (1-p_X)p$ and $H(X|Y)$ is different from $H(p)$. The minimum achievable rate for source X is in this case given by:

$$H(X|Y) = H(p_X) + H(p) - H(p_Y)$$

Fig. 1 shows that the capacity of the equivalent BSC between X and Y increases when $p_X \neq 0.5$. Therefore, knowing the source distribution, one gets closer to the Slepian-Wolf bound using the same code.

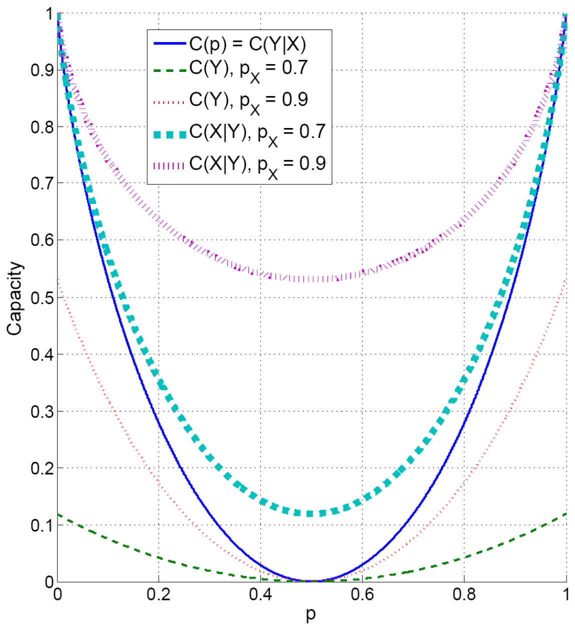

Figure 1: Capacity of the equivalent channel for $p_X = 0.7$ and $p_X = 0.9$.

Motivated by the capacity gain when the source distribution is known, we turn to adapting the existing source and channel Slepian-Wolf coding schemes to take into account the knowledge of the source distribution, and to see which type of codes more benefits from that intrinsic information.

3. SYNDROME BASED TURBO CODING OF NON-UNIFORM SOURCES

The turbo code we use in this system (Fig. 3) is composed of two identical convolutional codes, and the decoding is performed using the syndrome trellis first described in [7]. Instead of generating parity bits, we give syndromes to the decoder.

3.1 The syndrome trellis

Given the constituent convolutional code of generator matrix G, we build the syndrome trellis based on the parity-check matrix H, s.t. $H \cdot G = 0$. This construction is of low cost in the sense that there is no need to expand the parity check polynomials matrix into a matrix of an equivalent block code of large dimension. We consider here an example for an easier explanation. Let H be the parity-check matrix of the $(3,1,4)$ convolutional code with constraint length $L = 4$. Here $n = 3$, $n-k = 2$.

$$\mathbf{H} = \begin{pmatrix} 11\ 15\ 06 \\ 15\ 12\ 17 \end{pmatrix}_{(oct)} = \begin{pmatrix} 1001\ 1101\ 0110 \\ 1101\ 1010\ 1111 \end{pmatrix}_{(bin)} \quad (1)$$

In order to efficiently construct the syndrome trellis, we derive the following diagram from the binary form of $\mathbf{H}$ in equation (1).

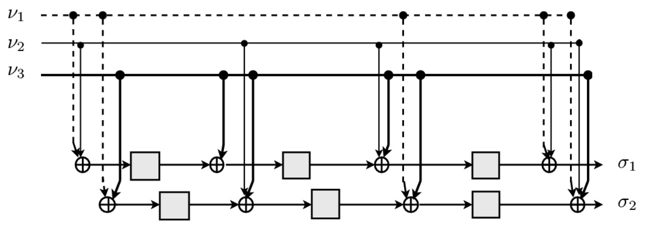

Figure 2: Block diagram for the computation of the syndrome for the rate 3 : 2 convolutional code. The boxes in gray represent the memories.

The states of the resulting trellis are determined by the values of the memories in that diagram. The transition between two states of the trellis is labeled by three input bits $v_1 v_2 v_3$ and two output bits $\sigma_1 \sigma_2$.

3.2 Decoding with the syndrome trellis

The decoder, knowing the syndrome s_X, looks for the sequence having that syndrome which is closest to $\mathbf{y}$ in terms of their Hamming distance. A modified BCJR is used. The recurrences for the calculation of the forward state metric, noted α in the literature, and the backward state metric, β, are the same; basically, the only change to bring to the original BCJR decoding [2] is the calculation of the branch metric, γ.

Let $(m_t)_{t=1\ldots\tau}$ the sequence of states of the trellis corresponding to a given block $\mathbf{x}$. Let v_1^n be the n input bits and σ_1^{n-k} the $(n-k)$ output bits labeling the transition between the states m_{t-1} and m_t, as on Fig. 2. Let y_1^n be the current side information bits and s_1^{n-k} current syndrome bits. Let p_j be the extrinsic probability $\mathbb{P}(\hat{x}_j = 1)$. By definition, $\gamma = \mathbb{P}(m_t, \mathbf{y}_t | m_{t-1})$ is given by:

$$\gamma = \delta_{\sigma_1^{n-k}=s_1^{n-k}} \cdot \prod_{j=1}^{n} \Big(p^{\delta_{v_j \neq y_j}} \cdot (1-p)^{\delta_{v_j = y_j}} \cdot p_j^{\delta_{v_j=1}} \cdot (1-p_j)^{\delta_{v_j=0}} \cdot p_X^{\delta_{v_j=1}} \cdot (1-p_X)^{\delta_{v_j=0}} \Big) \tag{2}$$

where δ is the Kronecker's delta ($\delta_{bool} = 1$ if $bool = true$ and 0 otherwise).

Because the transitions of the trellis that do not match the syndrome are not followed (since $\delta_{\sigma_1^{n-k}=s_1^{n-k}} = 0$), the search is actually performed in the coset with syndrome s_X. Note that to compress the source, the syndrome has to be punctured; that changes the term $\delta_{\sigma_1^{n-k}=s_1^{n-k}}$ of (2) into the following form:

$$\prod_{i=1}^{n-k} \left(\left(\frac{1}{2}\right)^{\delta_{Pct(s_i)}} \left(\delta_{\sigma_i = s_i}\right)^{\delta_{\overline{Pct(s_i)}}} \right)$$

where "$Pct(s_i)$" is the information that bits i of the syndrome is punctured. Puncturing the syndrome makes our code rate compatible for other compression ratio.

The first line in the product of equation (2) formalizes the information from the side information, the second line exploits the extrinsic probabilities and the last line exploits the source probabilities. That last term effectively favours the transitions which are labeled by inputs v_1^n which distribution are statistically close to the source distribution.

3.3 The syndrome trellis framework for coding of non uniform sources

The source X, having realizations $\mathbf{x}$ of length n, is mapped into its two syndromes $\mathbf{s}_{X1}$ and $\mathbf{s}_{X2}$, of length $(n-k)$, s.t. $\frac{2\cdot(n-k)}{n} = R_X, R_X \geq H(X|Y)$. Then the modified BCJR is used for each convolutional decoder to estimate $\hat{\mathbf{x}}$, passing iteratively updated soft extrinsic messages between them at each iteration. The intrinsic information about $\mathbf{y}$ and the source probabilities remain constant over the iterations. The turbo decoding stops when $\hat{\mathbf{x}}$ matches the two syndromes, or when a maximum number of iterations is reached.

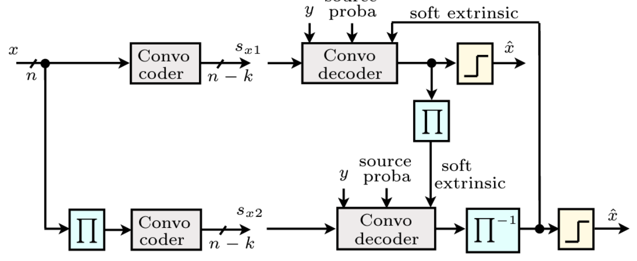

Figure 3: Turbo-syndrome scheme for asymmetric coding of correlated sources.

4. DISTRIBUTED ARITHMETIC CODING OF NON-UNIFORM SOURCES

4.1 DAC encoding

Let X be a binary memoryless source emitting symbols X_i with probability p_X. The classical binary arithmetic coding process for X uses the probabilities p_X and $1-p_X$ to partition the $[0,1)$ interval into sub-intervals associated to possible occurrences of the input symbols. At initialization the current interval is set to $I_0 = [0,1)$. For each input symbol, the current interval I_i is partitioned into two adjacent sub-intervals of lengths $(1-p_X)|I_i|$ and $p_X|I_i|$, where $|I_i|$ is the length of I_i. The sub-interval corresponding to the actual value of X_i is selected as the next current interval I_{i+1}, and this procedure is repeated for the next symbol. After all N symbols have been processed, the sequence is represented by the final interval I_N. The codeword C_X can consist in the binary representation of any number inside I_N.

DAC is based on the principle of inserting some ambiguity in the source description during the encoding process. This is obtained employing a set of intervals whose lengths are proportional to the modified probabilities $\widetilde{p}_X^0 \geq (1-p_X)$ and $\widetilde{p}_X^1 \geq p_X$. In order to fit the enlarged sub-intervals into the $[0,1)$ interval, they are allowed to partially overlap. The encoding procedure is exactly the same as for arithmetic coding, but employs the modified probabilities. The codeword C_X is shorter, i.e., the bit-rate is smaller, so much so as the interval overlap is large. The amount of overlap is a parameter that can be selected so as to achieve the desired rate, which should be no less than $H(X|Y)$.

In practice, the DAC encoding process has to be terminated properly in order to avoid clusters of errors at the end of the block. This can be done in several ways, e.g., encoding a known termination pattern or end-of-block symbol with a certain probability or, in the case of context-based AC, driving the AC encoder in a given context. For DAC, in this paper termination is obtained by encoding the last T symbols of the sequence without interval overlap. As T increases, the average error location tends to move towards the center of the block, yielding a correct decoder behavior. However, the termination has a cost in terms of bit-rate, as the last T symbols do not benefit from the Slepian-Wolf bit-rate saving.

4.2 DAC decoding

The DAC decoding process can be formulated as a symbol-driven sequential search along a proper decoding tree, where each node represents a state of the sequential arithmetic decoder. When the i-th input symbol X_i is decoded, if the codeword lies in overlapped region of the current interval then the decoder performs a branching. Two alternative paths are stored in the decoding memory, corresponding to the two alternative decoded symbols $X_i = 0$ and $X_i = 1$ that could be output at this step. For each new state, the associated branch metric is updated, and the corresponding interval is selected for next iteration. In particular, the correlated side information Y is employed to compute the *Maximum A Posteriori* branch metric $P(X|C_X, Y)$. In order to reduce complexity, after decoding a new input symbol, the decoder keeps only the M paths with the best partial metric, and prunes the others; this is done using the M-algorithm. More details on the DAC encoding and decoding procedures can be found in [13].

The DAC decoding algorithm is suboptimal, as the M-algorithm only keeps a limited number of likely decoding paths. If the correct path is dropped at some point during the decoding process, decoding will be unsuccessful. Thus, one would want to keep M very large in order to achieve the best performance, i.e., the lowest residual error rate. On the other hand, M heavily affects the decoder complexity, as it directly impacts on the size of the search space.

5. EXPERIMENTAL RESULTS

We carry out simulations based on the following conditions. We consider X with source distributions $p_X = \{0.5, 0.7\}$, of length $n = 1000$. For each source distribution, we consider the compression ratio 2 : 1 and 3 : 1 for source X; Y is compressed at its entropy-rate $H(Y)$, which depends on the correlation between the sources. Actually, given the compression ratio R_X, the source probability p_X and $H(X|Y)$, we compute the corresponding p s.t. $H(p_X) + H(p) - H(p_Y) = H(X|Y)$, where $p_Y = p_X(1-p) + (1-p_X)p$.

In order to plot comparative curves, we consider different values of $H(X|Y)$ and measure the bit error rate (BER) between $\hat{\mathbf{x}}$ and $\mathbf{x}$. For each value of $H(X|Y)$ at least 10^7 bits are simulated. We choose the parameters of the DAC and the turbo codes so as to obtain the same coding/decoding complexity.

For the system using turbo codes, we utilize constituent encoders defined by a constraint length $L = 4$ and by their octal parity-check matrix $\mathbf{H}$ of initial compression 3 : 2, see equation (1), yielding an initial redundancy ratio of 3 : 4 for the resulting turbo code. The different compression ratio 2 : 1 and 3 : 1 are obtained with a regular puncturing pattern of the syndrome. The interleaver is random and generated only once: the same interleaver is used for all the simulations. At most 20 decoder iterations are performed for each block of length n.

For the DAC, the desired compression ratio is achieved by properly selecting the overlap factor given the probability P_X, and taking into account the additional overhead generated by the termination. Throughout the simulations we employ $T = 20$. The rate achieved by the DAC over each data block is not always identical, but its deviation from the nominal value is very small. Over each simulation of 10^7 samples, the actual achieved rate is on average 0.2% smaller than the nominal rate. The value of M for the DAC decoder has been taken so that the complexity is roughly the same as the turbo code system. Both programs have been run on a Linux workstation, performing 200 rounds of encoding and decoding and measuring the running time. Taking $M = 1024$ yields similar times; this value has been used for all simulation results reported in this paper. While this procedure only yields a rough estimate of the complexity of the two systems, they are so different that an analytical complexity comparison is not viable.

Fig. 4 shows the BER achieved with the different methods, for different source probabilities and for different compression ratio.

For compression rate 2 : 1, the turbo code is consistently better, except when the conditional entropy is very small. This is because the DAC decoder is suboptimal, and this is particularly evident as the correlation decreases. For compression rate 3 : 1, the DAC is consistently better. The reason is that the heavier puncturing makes the syndrome based turbo code less efficient. The turbo code is still better at low correlation, as the DAC suffers from the decoder sub optimality.

From $p_X = 0.5$ to $p_X = 0.7$, both the turbo code and the DAC improve their distances to the Slepian-Wolf bounds. They clearly benefit from the prior knowledge of the source distribution. That gain should increase as the sources become

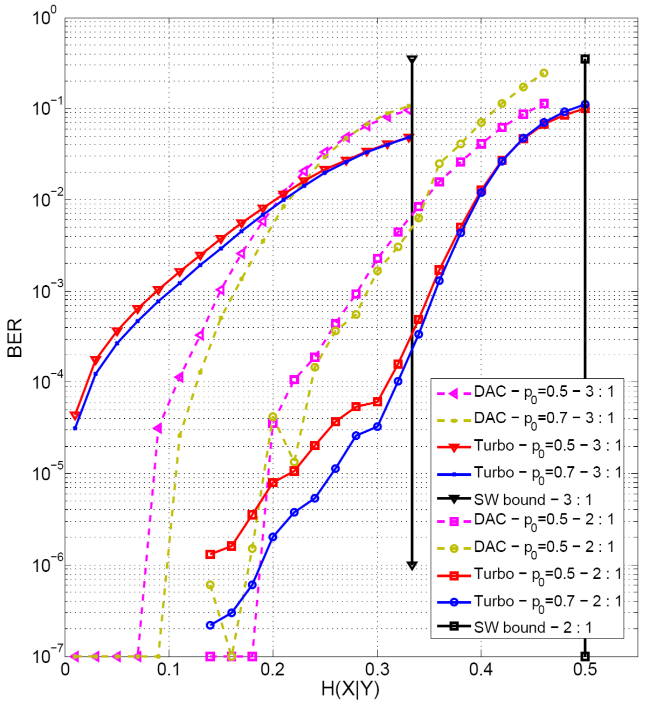

Figure 4: BER versus $H(X|Y)$ for DAC and turbo codes. $n = 1000$, $p_X = 0.5$ and $p_X = 0.7$.

less uniform, as expected from Fig. 1.

Finally, we have studied the DAC performance as the decoder complexity M varies. In particular, values of M ranging from 256 to 4096 have been taken. Simulations have been run for $p_X = 0.5$ and compression ratio 2:1. The results are shown in Fig. 5. As can be seen, between $M = 256$ and $M = 4096$ there is roughly an order of magnitude of BER difference. Interestingly, the performance gain does not tend to saturate in this range of M. This indicates that the sequential decoding process is rather suboptimal, and the performance can be improved even more by further increasing M, although the computation times become prohibitive.

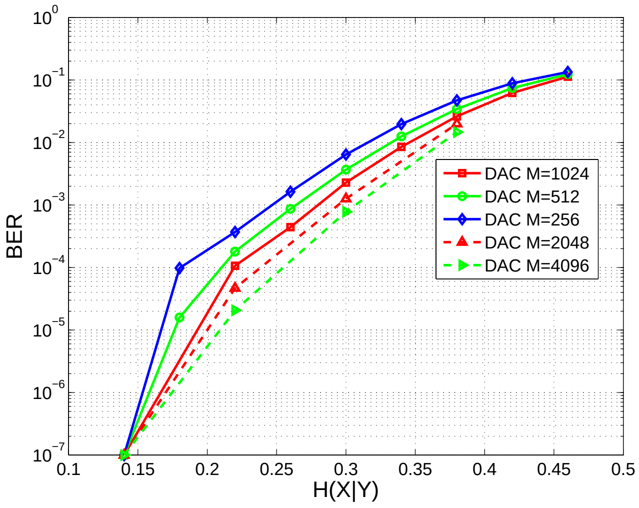

Figure 5: DAC performance as a function of M. $n = 1000$, $p_X = 0.5$.

6. CONCLUSION

We have discussed the advantages and the drawbacks of distributed arithmetic codes and turbo codes when encoding non uniform distributed correlated sources. Knowing the distribution of the source, both approaches improve their performances in terms of their distances to the Slepian-Wolf bound. The DAC performance gap with respect to the Slepian-Wolf bound is less impacted than for the turbo codes when the compression ratio varies. However, the turbo code is clearly better when the correlation between the sources is low and when the puncturing is low.

REFERENCES

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

[2] L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, 284–287, March 1974.

[3] C. Berrou, A. Glavieux, P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding:Turbo-codes," , *IEEE International Conference on Communications*, vol. 2, pp. 1064–1070, May 1993.

[4] A. Wyner, "Recent results in the Shannon theory," *IEEE Transactions on Information Theory*, vol. IT-20, pp. 2–10, Jan. 1974.

[5] M. Sartipi and F. Fekri, "Distributed Source Coding in Wireless Sensor Networks using LDPC coding: The entire Slepian-Wolf Rate Region," *IEEE Wireless Communications and Networking Conference*, vol. 4, pp. 1939–1944, March 2005.

[6] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov and M. Ouaret, "The Discover Codec: Architecture, Techniques And Evaluation," *Proceedings of Picture Coding Symposium (PCS)*, Nov. 2007.

[7] A. Roumy, K. Lajnef, and C. Guillemot, "Rate-adaptive turbo-syndrome scheme for Slepian-Wolf coding," in *41st Asilomar Conference on Signals, Systems and Computers*, Nov. 2007.

[8] J. Li and H. Alqamzi, "An Optimal Distributed and Adaptive Source Coding Strategy Using Rate-Compatible Punctured Convolutional Codes," in *Proceeding of IEEE International Conference on Acoustic, Speech, and Signal Processing (ICASSP)*, March 2005.

[9] Z. Tu, J. Li, and R. Blum, "An efficient turbo-binning approach for the Slepian-Wolf source coding problem," in *Eurasip Jour. on Applied Signal Processing - Special Issue on Turbo Processing*, 2003.

[10] P. Koulgi, E. Tuncel, S.L. Regunathan, and K. Rose, "On zero-error source coding with decoder side information," *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 99–111, Jan. 2003.

[11] Q. Zhao and M. Effros, "Lossless and near-lossless source coding for multiple access networks," *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 112–128, Jan. 2003.

[12] M. Grangetto, E. Magli, and G. Olmo, "Distributed arithmetic coding," *IEEE Communications Letters*, vol. 11, no. 11, pp. 883–885, Nov. 2007.

[13] M. Grangetto, E. Magli, and G. Olmo, "Distributed arithmetic coding for the Slepian-Wolf problem," *IEEE Transactions on Signal Processing (to appear 2009)*, preprint available at URL `www1.tlc.polito.it/sas-ipl/Magli/index.php`.

[14] M. Grangetto, E. Magli, R. Tron, and G. Olmo, "Rate-compatible distributed arithmetic coding," *IEEE Communications Letters*, vol. 12, no. 8, pp. 575–577, Aug. 2008.

[15] X. Artigas, S. Malinowski, C. Guillemot, and L. Torres, "Overlapped quasi-arithmetic codes for distributed video coding," in *Proceedings of IEEE ICIP*, 2007, pp. 9–12.

[16] X. Artigas, S. Malinowski, C. Guillemot, and L. Torres, "Overlapped arithmetic codes with memory," in *Proceedings of EUSIPCO*, 2008.

[17] J. Li, Z. Tu, and R. Blum, "Slepian-Wolf Coding for Nonuniform Sources Using Turbo Codes," in *Proceedings of the Conference on Data Compression*, 2004.

[18] A. D. Liveris, Z. Xiong, and C. N. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," in *IEEE Communications Letters*, vol. 6, no. 10, pp. 440442, October 2002.

[19] T. Tian, J. Garcia-Frias, and W. Zhong, "Compression of correlated sources using LDPC codes," in *Proceedings of the IEEE Data Compression Conference*, p. 450, 2003.