16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

OVERLAPPED ARITHMETIC CODES WITH MEMORY

Xavi Artigas[1], Simon Malinowski[2], Christine Guillemot[2] and Luis Torres[1]

[1]Department of Signal Theory and Communications,
Technical University of Catalonia
Campus Nord, D-5, Jordi Girona 1-3
08034 Barcelona, SPAIN
{xavi, luis}@gps.tsc.upc.edu
http://gps-tsc.upc.es/GTAV/

[2]IRISA
Campus Universitaire de Beaulieu
35042 Rennes Cedex, FRANCE

{simon.malinowski, christine.guillemot}@irisa.fr
http://www.irisa.fr/temics/

ABSTRACT

This paper describes a family of codes based on arithmetic coding with overlapping intervals. These codes are not uniquely decodable but the presence of correlated side information at the decoder can be exploited to achieve a vanishing decoding error probability, making them well suited for the Slepian-Wolf problem. Since these are source codes, they are also particularly well suited for sources exhibiting non-uniform symbol probabilities or memory; and the provided experimental results support this assertion. The construction of the codes is described, along with the corresponding soft decoding algorithm with side information. Finally, simulation results are given which compare very favourably against turbo codes in the presence of source memory.

1. INTRODUCTION

Distributed source coding (DSC) refers to the problem of compressing correlated signals captured by different sensors which do not communicate between themselves. DSC finds its foundation in the seminal Slepian-Wolf [1] and Wyner-Ziv [2] theorems establishing lossless rate bounds and rate-distortion bounds respectively. Most Slepian-Wolf and Wyner-Ziv coding systems are based on channel coding principles, using e.g., coset codes [3] or turbo codes [4]. The statistical dependence between two sources is modelled as a virtual correlation channel analogous to binary symmetric channels or additive white Gaussian noise (AWGN) channels. The use of entropy source codes tailored to the conditional distribution between the two sources has also been considered for DSC in [5].

This paper considers the design of Slepian-Wolf codes based on arithmetic and quasi-arithmetic codes. The basic idea is to allow the intervals corresponding to each source symbol to overlap. This procedure can achieve arbitrary compression controlled by the amount of allowed overlap, but, as already stated by Langdon in 1984 [6], it leads to a code which is not uniquely decodable. The side information available only at the decoder is then used to disambiguate the overlaps and achieve a vanishing decoding error probability.

It is shown that overlapped arithmetic (OA) and overlapped quasi-arithmetic (OQA) codes can be modelled with an encoding state machine (SM). The method to obtain these state machines is adapted from the ones proposed in [7] and [8]. The representation of OA and OQA codes by SM allows defining a state model for soft decoding of these codes. Finally, the side information available at the decoder is integrated into this state model to help the decoder remove the ambiguity introduced by the overlapping.

Multiple reasons motivate the research of source codes for the Slepian-Wolf problem as an alternative to channel codes. For example, the proposed DSC scheme can be easily extended to non-binary sources by just considering non-binary source codes. Moreover, for non uniform sources or sources with memory, the additional information about the distribution of the source can be directly exploited by the decoder, as shown later. Finally, the considered scheme does not require the source to be uniform to perform optimally, as turbo codes do, for example.

The idea of overlapped arithmetic coding was first introduced by the authors in [9], and also independently by Grangetto *et al.* in [10] and [11]. This work expands on [9] by taking source memory into account and by presenting more exhaustive results that show that, in the presence of source memory, the proposed codes perform better than turbo codes by an ample margin.

Section 2 describes overlapped codes, their construction and the problem raised by their ambiguous decoding (this section follows closely that of [9] in order to make this a self contained paper). Section 3 then introduces the used soft decoding algorithm. Section 4 presents simulation results for a broad spectrum of theoretical sources and finally Section 5 draws some conclusions and sketches the future research derived from this work.

2. OVERLAPPED ARITHMETIC CODES

Let $\boldsymbol{X} = X_1, \ldots, X_K$ be a symbol string generated by a binary memoryless source. The alphabet of the source is $\mathcal{A} = \{a, b\}$ and the probability of symbol a is denoted P_a. The addition of memory will be described in the next subsection. Only binary sources are considered in this work but the results can be generalized to non-binary alphabets.

The work presented was developed within DISCOVER, a European Project (http://www.discoverdvc.org), funded under the European Commission IST FP6 programme. This work has also been partially supported by TEC2005–07751–C02–02 grant of the Spanish Government.

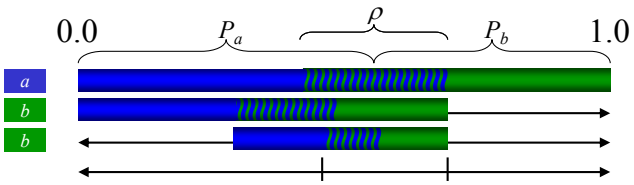

Figure 1 - Example of overlapped arithmetic encoding. P_s is the probability of symbol s and ρ is the overlapping factor. The sequence abb has been encoded into the shown final interval.

In classical binary arithmetic coding, the interval $[0,1)$ is recursively partitioned according to the source probabilities. At each symbol clock instant n, the current interval $\mathcal{I}_n = [L_n, H_n)$ is partitioned into two subintervals $\mathcal{I}^a_{n+1}$ and $\mathcal{I}^b_{n+1}$ whose widths are proportional to P_a and $P_b = 1 - P_a$ respectively:

$$\begin{aligned} \mathcal{I}^a_{n+1} &= \big[L_n, P_a(H_n - L_n)\big) \\ \mathcal{I}^b_{n+1} &= \big[P_a(H_n - L_n), H_n\big) \end{aligned} \tag{1}$$

One of these subintervals is selected according to the value of X_{n+1} and becomes the current interval. Once the last symbol is encoded, the encoder outputs enough bits to distinguish the final interval from all other possible intervals of $[0,1)$.

To reduce the coding delay, bits can be output during the encoding process as soon as the current interval is entirely in $[0, 0.5)$ (a bit 0 is output) or entirely in $[0.5, 1)$ (a bit 1 is output). The current interval is rescaled each time a bit is output. The arithmetic principle has been represented in [12] as a stochastic model used for soft decoding of these codes. The number of states is finite if the source probabilities are known but it exponentially increases with the sequence length.

It has been shown in [13] that the number of states of an arithmetic code can be reduced if a slight loss in compression performance is accepted. The initial interval is set to $[0, N)$ where N is an integer. The partitioning of the current interval is done as in arithmetic coding but the current intervals are always integer intervals. This principle is called quasi-arithmetic (QA) coding. The main interest of QA coding is that the number of states of the encoder is finite and does not increase with the sequence length.

In the case of overlapped encoding, the current interval is first partitioned according to the source probabilities, but the subintervals are then modified so that they overlap. The parameter ρ controls the width of overlapping between the subintervals. It can be seen in Figure 1 that the overlapping allows wider subintervals, and this leads to higher compression. The subintervals of $\mathcal{I}_n = [L_n, H_n)$ are defined as follows:

$$\begin{aligned} \mathcal{I}^a_{n+1} &= \big[L_n, P_a(H_n - L_n) + \rho N/2\big) \\ \mathcal{I}^b_{n+1} &= \big[P_a(H_n - L_n) - \rho N/2, H_n\big) \end{aligned} \tag{2}$$

Note that this partitioning of the current interval can be applied to both arithmetic (by taking $N = 1$) and quasi-arithmetic coding (by taking $N > 1$ and by rounding the bounds of the subintervals to the nearest integers).

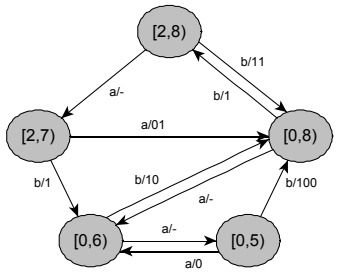

Figure 2 - SM for an example binary OQA encoder. N=8, P_a=0.7, P_b=0.3 and ρ=0.1. $\{a, b\}$ are input symbols, $\{0, 1\}$ are output bits. A dash indicates a transition that does not output any bit.

Only quasi-arithmetic coding will be considered in the following because *i*) it has been shown in [13] that the loss of compression efficiency introduced by integer arithmetic coding is very low (1% increase of the bitrate over the entropy in the worst case, with N=256) and *ii*) quasi-arithmetic coding allows the usage of well-known decoding algorithms, as will be shown in the text section. Also, note that QA coding is implemented with a SM, so its encoding complexity is negligible (comparable to turbo encoding). No floating point operations need to be performed at all.

Iteratively applying the overlapped method for partitioning the current interval, a finite state encode is obtained, which depends on N, P_a and the overlapping factor ρ. An example of OQA encoder is given in Figure 2. This encoder is not uniquely decodable. For instance, when at state [0, 8), symbol strings baa and abb are encoded with the same bit string 101.

The overlap principle results in enlarging the current interval, and hence in reducing the number of output bits. Examples of the compression performance of an $N = 256$ OQA encoder are given in Figure 3 for different values of ρ and P_a. For ρ = 0 (no overlapping) the compression rate is almost equal to the entropy (1% above the entropy, for P_a=0.6), as one would expect from conventional arithmetic codes. Larger values of ρ further reduce the rate below the source entropy, down to any desired rate. The analytical expression of the obtained rate is extremely cumbersome and is not presented here.

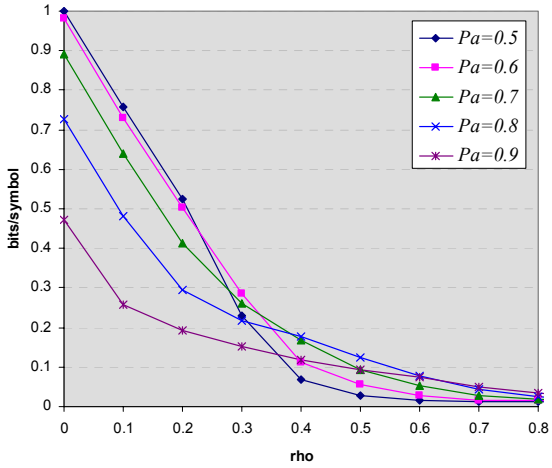

Figure 3 - Compression rates obtained with an $N = 256$ overlapped quasi-arithmetic code at different values of P_a and ρ after simulation of 1000 strings of 1000 symbols each.

2.1 Addition of source memory

If the source has an order-m memory, P_a will have different values depending on the preceding m symbols. For example, the higher bitplanes of an image or video sequence typically contain long strings of 0's or 1's, meaning that symbol 0 has a much greater probability of occurrence if the previous symbol was also a 0 and conversely for 1. Exploiting this source property has an important impact on the coding performance because the bound for the compression is then no longer the entropy of the source, but the entropy rate (i.e., the entropy of a symbol given all the previous symbols), which can be significantly lower depending on the amount of memory of the source.

To exploit the source memory, the state labels in the encoding automaton can be expanded to include the context (i.e., the m previous symbols), as in [0,8,*a*), which means that the current interval for this state is [0,8) and that the previous symbol was an a. Therefore, states [0,8,*a*) and [0,8,*b*) are considered separately even when they represent the same current interval, because they have a different P_a (called conditional probabilities, and denoted $P_{a|a}$ and $P_{a|b}$ respectively) and therefore produce different subdivisions.

Once the automaton has been created taking memory into consideration (in the state labels), the decoding process does not need to deal with the memory anymore, besides using the appropriate conditional P_a for the context of each state. In the following, only order-1 symmetric memory has been considered, therefore, the conditional probabilities are uniquely identified by the memory correlation factor ρ_m:

$$\begin{aligned} P_{a|a} &= 1 + (\rho_m - 1)(1 - P_a) \\ P_{a|b} &= -(\rho_m - 1)P_a \end{aligned} \tag{3}$$

ρ_m ranges from 0 (memoryless source) to 1 (the source only produces *a*'s or *b*'s).

3. SOFT DECODING WITH SIDE INFORMATION

Overlapped arithmetic and quasi-arithmetic encoders have been defined in the previous section. Both kinds of encoders can be defined with a state machine. The state machine generates a variable number of bits depending on the current state and the next source symbol that has to be encoded. A decoding SM can be computed by just inverting the inputs and outputs of the encoding SM.

Let $\mathcal{T} = \{\alpha_0, \ldots, \alpha_{d-1}\}$ be the set of states of the decoding SM. Two transitions exit each state corresponding to the two symbols that can be encoded, as seen in Figure 2. For every transition t in the encoding SM, let b_t be the bit string output by this transition and s_t the symbol that triggered it. The length of b_t is denoted $\overline{b_t}$.

The overlapping mechanism introduces ambiguity in the encoded bit string, and this means that, even if the received bit string is error-free, more than one symbol sequence will be retrieved by the decoding state machine. In the following, it will be assumed that side information $\boldsymbol{Y} = Y_1 \ldots, Y_K$ correlated to $\boldsymbol{X}$ is available at the decoder, and it will be modelled as if it had passed through a binary symmetric channel (BSC) with crossover probability $1 - \mu$. In other words, for $1 \leqslant i \leqslant K$, $\mathbb{P}(X_i = Y_i) = \mu$.

In the memoryless case, the Slepian-Wolf bound for the lowest achievable (still decodable) rate for $\boldsymbol{X}$ can be calculated as:

$$R_{\boldsymbol{X}}^{SW} = H(\boldsymbol{X}|\boldsymbol{Y}) = H(\boldsymbol{X}) - H(\boldsymbol{Y}) + H(\boldsymbol{Y}|\boldsymbol{X}) \tag{4}$$

Where $H(\boldsymbol{Y}|\boldsymbol{X})$ is the binary entropy of μ and $H(\boldsymbol{Y})$ is the binary entropy of $P_a\mu + (1 - P_a)(1 - \mu)$. For sources with memory the following equation is used. Note that $H(\boldsymbol{Y}|\boldsymbol{X})$ remains the same because the channel has been assumed memoryless.

$$\begin{aligned} R_{\boldsymbol{X}}^{SW} &= H(X_n|X_{n-1}, Y_n) \\ &= H(X_n|X_{n-1}) - H(Y_n|X_{n-1}) + H(Y_n|X_n) \end{aligned} \tag{5}$$

In order to optimally exploit the side information $\boldsymbol{Y}$, a symbol-clock-based state model is defined [7][12]. The state model for soft decoding of overlapped codes is defined by the pairs (N_k, M_k) where N_k is the state of the decoding SM at the bit clock instant k (i.e., $N_k \in \mathcal{T}$) and M_k represents the possible values of the symbol clock at the bit clock instant k (i.e. $1 \leqslant M_k \leqslant K$).

The probability of transition t is then defined as:

$$\mathbb{P}\big(N_k = \alpha_i, M_k = m \big| N_l = \alpha_j, M_l = m-1\big) = \begin{cases} \mathbb{P}\big(N_k = \alpha_i \big| N_l = \alpha_j\big) & \text{if } k - l = \overline{b_t} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Where the probabilities $\mathbb{P}(N_k = \alpha_i | N_l = \alpha_j)$ are deduced from the source statistics and the decoding SM. The BCJR [14] algorithm can then be applied on this state model in order to estimate $\boldsymbol{X}$. The symbol-clock-based model defined above allows integrating the side information brought by $\boldsymbol{Y}$ in the decoding trellis. To take into account this additional information, the branch metric of transition t, denoted $\gamma_t(N_k, M_k \,|\, N_l, M_l)$, used for the BCJR algorithm is defined in (7), where $\boldsymbol{U}_l^k$ is the received bit string from bit l to bit k and $\mathbb{P}(Y_m|X_m = s_t)$ is deduced from the correlation between $\boldsymbol{X}$ and $\boldsymbol{Y}$:

$$\mathbb{P}\big(Y_m \big| X_m = s_t\big) = \begin{cases} \mu & \text{if } Y_m = s_t \\ 1 - \mu & \text{otherwise} \end{cases} \tag{8}$$

The BCJR algorithm is then applied on this state model to compute the *a posteriori* marginal probabilities of each symbol $\mathbb{P}(X_m|\boldsymbol{Y})$.

4. EXPERIMENTAL RESULTS

In order to present the performance of this new class of codes, four different sources have been simulated to try to cover the broadest possible spectrum. The four sources are symmetric and asymmetric, with and without memory.

$$\gamma\big(\alpha_i, m \,\big|\, \alpha_j, m-1\big) = \begin{cases} \mathbb{P}\big(N_k = \alpha_i, M_k = m \big| N_l = \alpha_j, M_l = m-1\big) \times \mathbb{P}\big(Y_m \big| X_m = s_t\big) & \text{if } U_l^k = b_t \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

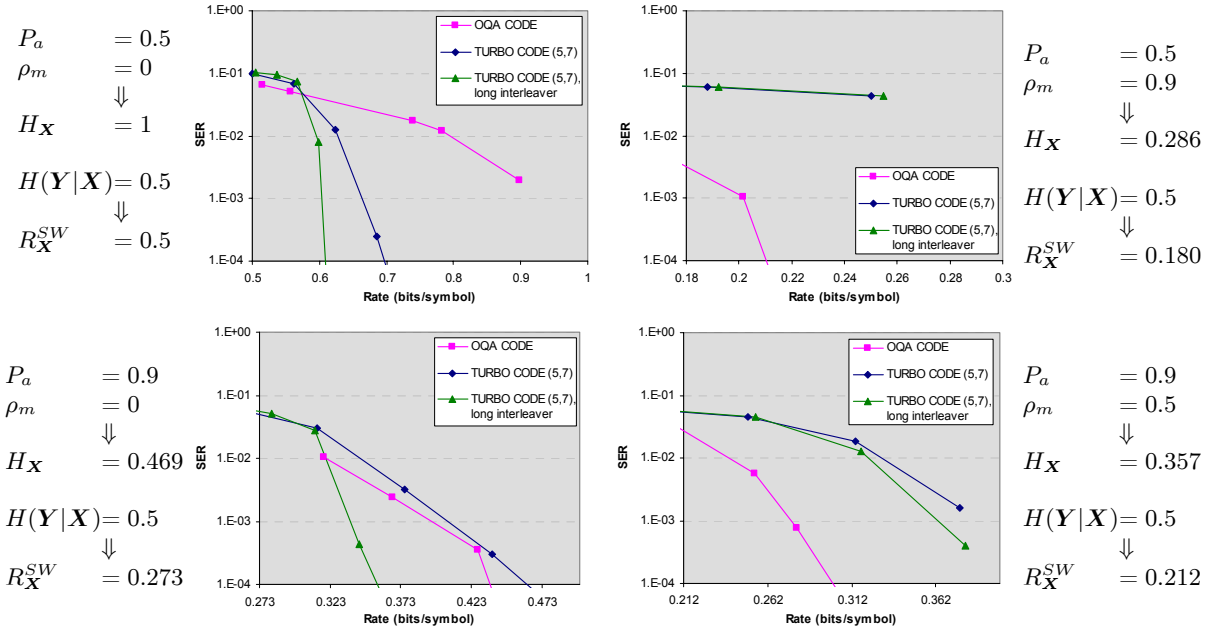

Figure 4 – Decoding performance for a fixed $H(\boldsymbol{Y}|\boldsymbol{X}) = 0.5$. 100 sequences of 10^3 bits have been simulated. For comparison, a turbo code with a long interleaver of 10^4 bits is also included. The leftmost point of the rate axis corresponds to $R_{\boldsymbol{X}}^{SW}$ calculated using (5).

Once the source is fixed, the decoding error probability (or Symbol Error Rate, SER), depends mainly on two variables: the bit rate and the side information correlation μ. Instead of the complete simulation of this two-dimensional search space, which would be cumbersome to plot and understand, two experiments have been conducted: the first one leaves μ fixed and changes the rate, whereas the second one leaves the rate fixed and changes μ.

4.1 Fixed side information quality

In this experiment a fixed conditional entropy $H(\boldsymbol{Y}|\boldsymbol{X}) = 0.5$ has been selected (corresponding to a crossover probability $\mu = 0.889$) because the obtained results are very illustrative. The Slepian-Wolf bound has been calculated using (5) and is used as the lower limit for the rate axis, since rates below this bound are not decodable according to the Slepian-Wolf theorem [1]. The upper limit for the rate axis is roughly the entropy rate of the source, since it is pointless to use distributed source coding above this rate, as conventional entropy coding can achieve this rate without incurring in decoding errors. The block length is 1000 symbols; turbo codes use random interleavers and have taken the source probabilities into account in the branch metrics, including memory in the first constituent decoder. The second constituent decoder cannot make use of the source memory because it is destroyed by the interleaver.

Figure 4 shows the obtained Symbol Error rate (SER) for the four different sources (left column is memoryless, right column has memory, upper row has symmetric symbols, lower row has asymmetric symbols). It can be seen that, for memoryless symmetric sources turbo codes produce much lower SER than OQA codes at any given rate. This is not surprising, since this is the kind of source channel codes are tailored to work with. However, for sources with memory or asymmetric, OQA codes perform better. In the presence of memory, the SER difference for a given rate can be as much as two orders of magnitude.

Since turbo codes are known to work better for long block lengths, another curve has been added for a turbo code with a long interleaver (or block length) of 10^4 symbols. It can be observed that indeed the turbo code with the long interleaver always works better than the same code with a shorter block length, and it also outperforms the OQA codes in both memoryless cases. However, for sources with memory, the length of the interleaver does not make a substantial difference and both turbo codes achieve worse SER than the OQA code at any given rate.

4.2 Fixed rate

This experiment aims at understanding the dependence of the obtained SER with the quality of the side information $H(\boldsymbol{Y}|\boldsymbol{X})$. A fixed rate has been selected, equal to the 75% of the entropy rate of the source (so that it is not achievable by conventional entropy coding) and $H(\boldsymbol{Y}|\boldsymbol{X})$ is moved throughout its whole range, from 0 (totally correlated side information) to 1 (totally uncorrelated). Using (5), the point at which the code should be decodable given the rate and the source probabilities is also calculated and used as the upper limit for the plot. This point is again the Slepian-Wolf bound: values of $H(\boldsymbol{Y}|\boldsymbol{X})$ to its left should be decodable, and values to its right should not.

Figure 5 shows the obtained results with the same sources and conditions as Figure 4. It can be observed that similar behaviour as in the previous experiment is present: OQA codes perform favourably in the presence of memory, but not as much as in the previous experiment, indicating that the dependence of the decoding performance with the rate and the side information is not immediately evident.

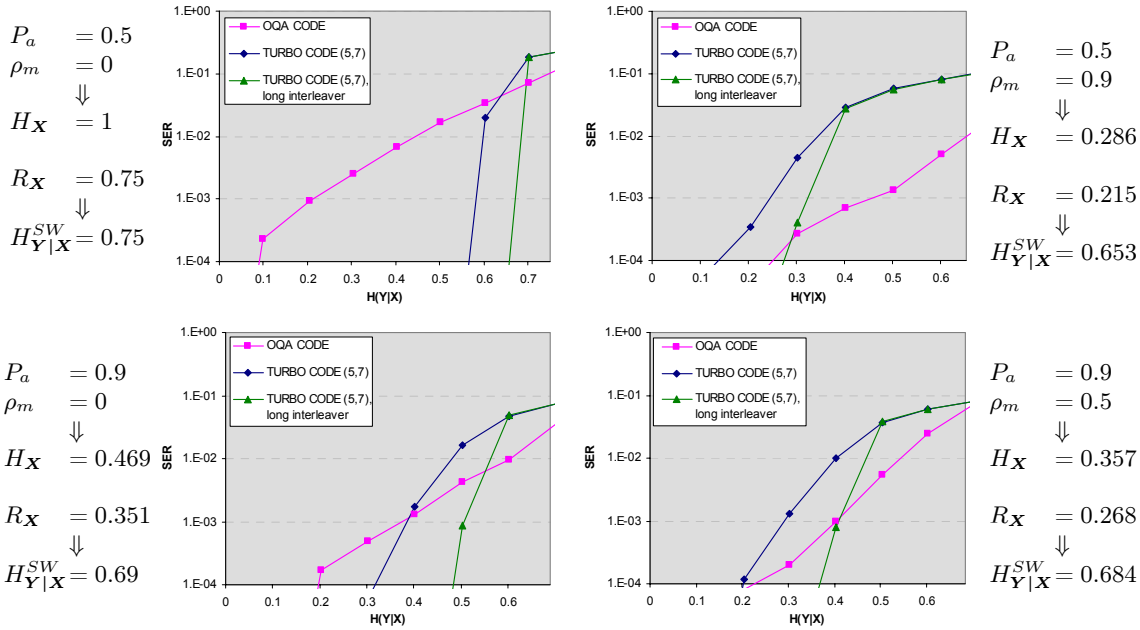

Figure 5 – Decoding performance for a fixed rate R_X equal to 0.75 times the entropy rate of the source. The same conditions as in Figure 4 have been used. $H^{SW}_{Y|X}$ stands for the conditional entropy (and therefore μ) required to reach the Slepian-Wolf bound, and it corresponds to the rightmost point of the conditional entropy axis.

5. CONCLUSIONS

This paper has described a Slepian-Wolf coding approach based on overlapped quasi-arithmetic codes, considering, particularly, the case of sources with memory. The first reported results show that, for sources with memory, OQA codes can outperform turbo codes, especially for block lengths up to 10^3 and 10^4 symbols. A possible reason is that source codes are better tailored to exploit source statistics than channel codes. Further work will be devoted to the study of other overlapping mechanisms as well as of iterative structures in order to try to improve the performance of these new codes.

REFERENCES

[1] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources", *IEEE Trans. Inform. Theory*, vol. 19 pp. 471-480, July 1973

[2] Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder", *IEEE Trans. Inform. Theory*, vol. 22, pp. 1-10, January 1976

[3] R. Puri and K. Ramchandran. "PRISM: A new robust video coding architecture based on distributed compression principles". *Proc. of 40th Allerton Conf. on Comm., Control, and Computing*, Allerton, IL, Oct. 2002

[4] B. Girod, A. Aaron, S. Rane and D. Rebollo-Monedero, "Distributed video coding", *Proc. of the IEEE*, vol. 93, no. 1, January 2005

[5] Qian Zhao, Effros, M., "Optimal code design for lossless and near lossless source coding in multiple access networks", *Proc. of Data Compression Conference* (DCC 2001). Page(s):263 – 272, 27-29 March 2001, Snowbird, Utah, USA

[6] G. G. Langdon, Jr., "An introduction to arithmetic coding", *IBM Journal of Research and Development*, vol. 28, no. 2, pp. 135-149, March 1984

[7] T. Guionnet and C. Guillemot, "Soft decoding and synchronisation of arithmetic codes: application to image transmission over noisy channels", *IEEE Trans. on Image Processing*, vol. 12, pp. 1599–1609, 2003.

[8] S. Ben-Jamaa, C. Weidmann, M. Kieffer, "Asymptotic Error-Correcting Performance of Joint Source-Channel Schemes based on Arithmetic Coding", *IEEE International Workshop on Multimedia and Signal Processing* (MMSP 2006), Victoria, Canada, October 3-6, 2006

[9] X. Artigas, S. Malinowski, C. Guillemot, L. Torres, "Overlapped quasi-arithmetic codes for distributed video coding", IEEE International Conference on Image Processing (ICIP), San Antonio, USA, September 16-19, 2007

[10] M. Grangetto, E. Magli, G. Olmo, "Distributed arithmetic coding", IEEE Communication letters, vol. 11, no. 11, pp. 883-885, November 2007.

[11] M. Grangetto, E. Magli, G. Olmo, "Symmetric Distributed Arithmetic Coding of Correlated Sources", IEEE International Workshop on Multimedia Signal Processing (MMSP), Crete, Greece, Oct. 2007

[12] T. Guionnet and C. Guillemot, "Soft and joint source-channel decoding of quasi-arithmetic codes", *EURASIP Journal on applied signal processing*, vol. 3, pp. 394-411, Mar. 2004

[13] P. Howard and J. Vitter, "Practical implementations of arithmetic coding", *Image and Text Compression*, J.A. Storer, ed. Kluwer Academic Publishers, Norwell, MA, pp. 85-112, 1992

[14] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimising Symbol Error Rate", *IEEE Trans. on Inform. Theory*, vol. IT-20, pp. 284-287, March 1974.