Distributed coding using punctured quasi-arithmetic codes for memory and memoryless sources

Simon Malinowski
IRISA/Univ. of Rennes
Simon.Malinowski@irisa.fr

Xavi Artigas
Technical University of Catalunya
xavi@gps.tsc.upc.edu

Christine Guillemot
IRISA/INRIA
Christine.Guillemot@irisa.fr

Luis Torres
Technical University of Catalunya
luis@gps.tsc.upc.edu

***Abstract*— This paper considers the use of punctured quasi-arithmetic (QA) codes for the Slepian-Wolf problem. These entropy codes are defined by finite state machines for memoryless and first-order memory sources. Puncturing an entropy coded bit-stream leads to an ambiguity at the decoder side. The decoder makes use of a correlated version of the original message in order to remove this ambiguity. A complete distributed source coding (DSC) scheme based on QA encoding with side information at the decoder is presented, together with iterative structures based on QA codes. The proposed schemes are adapted to memoryless and first-order memory sources. Simulation results reveal that the proposed schemes are efficient in terms of decoding performance for short sequences compared to well-known DSC solutions using channel codes.**

I. Introduction

Distributed source coding (DSC) addresses the problem of compressing correlated sources X and Y by encoding them separately and jointly decoding them. DSC is mainly applied in sensor networks, where several sensors measure a given signal and provide these correlated measures to a base station, which decodes them jointly. Recently, DSC has also been applied to video compression by exploiting the temporal correlation between consecutive images in a video sequence [1] [2] [3]. This correlation between consecutive images is used at the decoder side. DSC theory is based on the Slepian-Wolf theorem established in [4]. This theorem states that, even if the encoders of X and Y do not communicate with each other, lossless compression of X and Y can be achieved if the rates R_X and R_Y satisfy $R_X + R_Y \leq H(X,Y)$, provided that X and Y are decoded jointly. Later, this result was extended in [5] in order to compute rate-distortion bounds. In this paper, we focus on the so-called *asymmetric* Slepian-Wolf problem, in which the second source Y is encoded at its entropy rate $H(Y)$ and the first one X at the rate $H(X|Y)$. At the decoder side, X is estimated using its compressed version and the side information Y. The first practical application of the asymmetric Slepian-Wolf problem has been proposed in [6]. This solution is called DIstributed Source Coding Using Syndromes and consists in transmitting a syndrome instead of a complete codeword.. More recent practical applications of the asymmetric Slepian Wolf problem mostly use capacity-approaching channel codes in order to compress the original message. Regular turbo codes are for instance used in [7][8][9], whereas low density parity check codes are used in [10]. Recently, irregular turbo codes have been applied to the Slepian-Wolf problem in [11].

Entropy codes have also been used recently for the Slepian-Wolf problem. The idea behind using source codes for that issue is to exploit their high compression capability together with their capability to exploit the source memory. In [12], the authors have designed Huffman codes for multilevel sources. Two approaches based on overlapped arithmetic and quasi-arithmetic (QA) codes have also been developed in parallel in [13] and in [14]. The overlapping mechanism introduced in the arithmetic encoding process induces some ambiguity in the encoded bit-stream. The decoder makes use of the correlated source in order to remove this ambiguity. This technique reveals better performance than the techniques based on channel codes for short sequences. Using entropy codes for DSC presents many interests:

- In contrast with entropy codes, channel codes are more efficient for long sequences (typically more than 10^4 symbols).
- Entropy codes are better suited to take into account the source probabilities and the memory of the source.

Recent research have been aimed at improving the performance of channel codes in the DSC context for short sequences, in [15] and [16] for instance.

In this paper, we propose an alternative DSC scheme based on QA codes, adapted to memoryless sources and sources with memory. It is shown in [17] that QA codes adapted to memoryless sources can be represented by finite state machines (FSM). Hence, an optimal BCJR algorithm [18] can be applied at the decoder side, using the state model proposed in [19]. In order to exploit the side information available at the decoder (correlated version of the original message), the branch metrics of the BCJR algorithm have to be modified. The FSM are then extended to account for the realization of the previous symbol, so that the proposed DSC scheme can be applied to sources with memory. Then, we have designed iterative structures involving QA codes in order to improve the decoding performance of punctured QA codes. These structures are detailed in this paper. The performance of punctured QA code for Slepian-Wolf coding has been assessed against the one obtained with turbo codes, for both memory and memoryless sources.

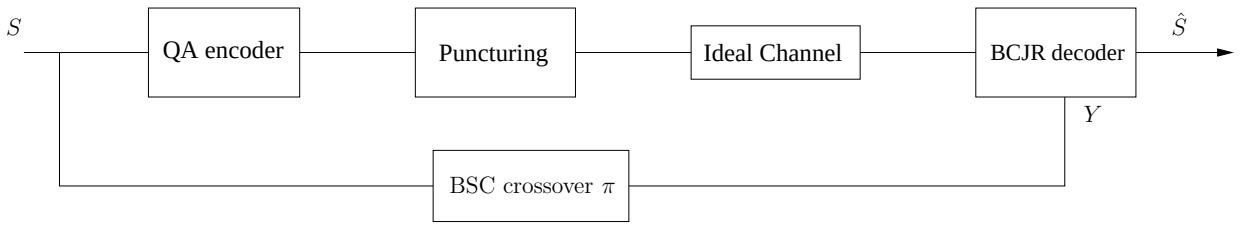

Fig. 1. The proposed DSC scheme

II. Description of the proposed DSC scheme

The proposed DSC scheme is represented in Fig.1. Let $\mathbf{S} = S_1, \ldots, S_{L(\mathbf{S})}$ be a source sequence of length $L(\mathbf{S})$ taking its value into the binary alphabet $\mathcal{A} = \{a, b\}$. The probability of the more probable symbol (MPS), $\mathbb{P}(a)$ will be denoted p in the following. Note that, for sake of clarity, only binary sources are considered in this article, but the whole method can be applied to non-binary sources. The symbol sequence $\mathbf{S}$ is encoded using a QA code, producing the bit-stream $\mathbf{X} = X_1, \ldots, X_{L(\mathbf{X})}$ of variable length $L(\mathbf{X})$. In order to reach a targeted overall rate, some bits of $\mathbf{X}$ are punctured. The resulting bit-stream is sent over an ideal channel. At the decoder side, a BCJR algorithm [18] is applied. This algorithm makes use of the received bits of $\mathbf{X}$, together with an additional side information $\mathbf{Y}$. This side information is obtained by passing $\mathbf{S}$ through a binary symmetrical channel of crossover probability π. In other words, $\forall i \in \{1, ..., L(\mathbf{S})\}$, $\mathbb{P}(\mathbf{Y}_i = \mathbf{S}_i) = 1 - \pi$. In the following, the notation $Y_m^{m'}$ will denote the sequence $Y_m, \ldots, Y_{m'}$. This scheme is explained in more detail in the rest of this section.

A. Quasi-arithmetic codes

In classical arithmetic coding, the initial interval $[0, 1]$ is recursively partitioned according to the source sequence and the source probabilities. The width of the final interval is equal to the probability of the source sequence that has to be transmitted. A bit sequence that distinguishes this final interval from all the others is then sent. The major drawback of arithmetic coding is that the encoding and decoding processes cannot be modeled with automata or finite state machines.

The authors of [20] have shown that by working on integer intervals rather than on real ones, the loss in terms of compression was very slight. This new kind of codes is called quasi-arithmetic codes. QA codes are hence defined by an integer N that represent the width of the initial interval. The interest of these codes is that they can be represented by automata. The states of this automata are tuples (l, u, f) as proposed in [19], where l and h respectively represent the lower and upper bound of the current interval I_c, and where f is an integer. The construction of the automata representing the encoder of a binary QA code is detailed in the following.

1) The initial state of the automata is $(0, N, 0)$: the initial interval being $[0, N[$ and f being initialized to 0.
2) I_c is partitioned into $I_c^a = [l^a, u^a[$ and $I_c^b = [l^b, u^b[$, according to the source probability $p = \mathbb{P}(a)$
3) I_c^a and $\mathcal{I}_c^b$ are rescaled (according to the rescaling rules described below) if and as many times as necessary
4) Two states are obtained: (l^a, u^a, f^a) and (l^b, u^b, f^b). These states are added to the automata if they were not
5) Step 2,3,4 are repeated for each new state of the automata

The rescaling rules of the third step below are the following. A sub-interval $[l, u[$ is rescaled if

1) $u < N/2$. Then, $l \rightarrow 2l$ and $u \rightarrow 2u$. 0 is emitted, followed by f bits 1. $f \rightarrow 0$.
2) $l > N/2$. Then, $l \rightarrow 2(l - N/2)$ and $u \rightarrow 2(u - N/2)$. 1 is emitted, followed by f bits 0. $f \rightarrow 0$.
3) $N/4 \leq l < N/2 \leq u < 3N/4$. Then $l \rightarrow 2 \times (l - N/4)$ and $u \rightarrow 2 \times (u - N/4)$. $f \rightarrow f + 1$.

Using this representation, the number of states may grow to infinity, due to the integer f that can be unbounded. In order to keep the number of states finite, authors of [17] have chosen to keep f below a threshold F_{max}. When f equals F_{max}, the source probabilities are modified in order to force the encoder to use the rescaling rule 1 or 2, so that bits are emitted and f is reset to 0.

Following the procedure described above, the encoder of a QA code can be represented by a FSM. Transitions on this FSM are labeled with symbols as input and bits as output. The decoder FSM can be obtained from the encoder one.

In [17], only memoryless sources are considered. In the following, we will also consider first-order memory sources. The construction of the encoding automaton representing a QA code for a first-order memory source is detailed hereafter. A first order memory source is uniquely defined by the probability of the MPS $\mathbb{P}(a) = p$ and a correlation coefficient ρ_m ($\rho_m \in [-1, 1]$). Then, we have

$$\begin{array}{ll} \mathbb{P}(a \,|\, a) & = 1 + (\rho_m - 1)(1 - p) \\ \mathbb{P}(b \,|\, b) & = 1 + (\rho_m - 1)p. \end{array} \quad (1)$$

The design of QA codes for first order memory sources is inspired from the description above. The initial state of the automata is set to $(0, N, 0)$ (as the step 1 for memoryless sources). Two new states are computed for symbols a and b by partitioning the interval $[0, N]$ according to p and $1 - p$ respectively (steps 2 and 3). However, in order to account for the memory of the source, the previous symbol has to be integrated in the states of the automata. Hence, the states (apart from the initial one) are composed by tuples (l, u, f, s) where s represents the previous symbol. For every state that is created, steps 2,3 and 4 are repeated. However, in order

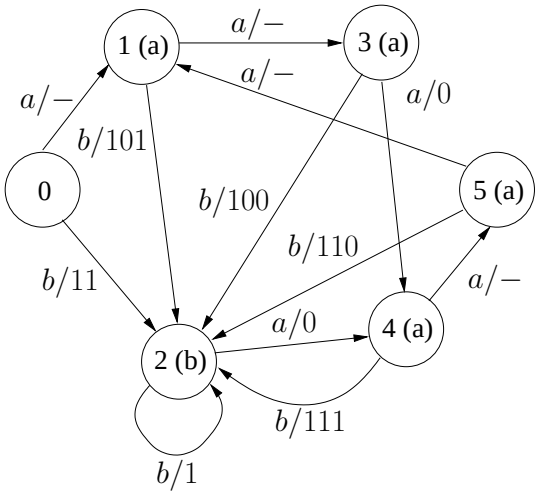

Fig. 2. Encoding automaton for a QA code with memory. This automaton is designed for $p = 0.8$ and $\rho_m = 0.3$

to partition the current interval, the probabilities $\mathbb{P}(a \,|\, a)$ and $\mathbb{P}(b \,|\, a)$ for each state such that $s = a$ and the probabilities $\mathbb{P}(a \,|\, b)$ and $\mathbb{P}(b \,|\, b)$ for the states such that $s = b$.

The encoding automaton for $p = 0.8$ and $\rho_m = 0.3$ is depicted in Fig.2. This automaton has been built according to the technique described above. The initial state of this automaton is state 0, corresponding to the tuple $(0, 8, 0)$ (this state does not contain the memory of the previous symbol). Note that the symbol in parenthesis in the labeling of a state corresponds to the memory of the previous symbol.

B. Puncturing

The encoding of $\mathbf{S}$ results in a bit-stream $\mathbf{X}$ of variable length $L(\mathbf{X})$. Let us denote by R_q the compression rate of the considered QA code. Then the average length of $\mathbf{X}$ is equal to $R_q \times L(\mathbf{S})$. Let a targeted compression rate be denoted by R_t, with $R_t < R_q$. In order to reach R_t, $\lceil (R_q - R_t) \times L(\mathbf{S}) \rceil$ bits in $\mathbf{X}$ have to be punctured. Different techniques exist in order to find the puncturing positions. In [21], the bits are inserted line by line into a square matrix, and punctured column by column. In our case, the best decoding results have been obtained by spreading the punctured bits along the bit-stream. The punctured bits are separated by $\lfloor (R_q - R_t) \times L(\mathbf{S}) / L(\mathbf{X}) - 1 \rfloor$ bit positions. Note that if $(R_q - R_t) \times L(\mathbf{S}) > L(\mathbf{X})/2$ (i.e., if more than half of the bits in $\mathbf{X}$ have to be punctured), the above technique is used to compute the non-punctured positions. The interest of this technique is that the decoder only needs to know the interval between two punctured bits to recover the punctured positions (it is assumed that the first punctured bit is in the first position).

C. Soft decoding with side information

In this section, the decoding algorithm of punctured quasi-arithmetic codes with side information is detailed. As the side information is an information about the symbols, an automaton sequential with respect to symbols is used (as the one of Figure 2 for example). Let $\mathcal{I} = \{e_0, \ldots, e_d\}$ be the set of states of the QA automaton. For every transition t in the automaton, let b_t denote the sequence of bits output by t and $\overline{b_t}$ the length of b_t. In the proposed DSC scheme, the BCJR decoder takes as input the received bit-stream $\mathbf{X}$ (that contains punctured bits) and the side information $\mathbf{Y}$. In order to optimally exploit the side information on the symbols, a symbol clock based state model is used. This model is adapted from the model proposed in [19] for QA codes. It is defined by the pair of random variables $\mathcal{V}_k = (N_k, M_k)$, where N_k is the state of the QA automaton at the symbol clock instant k (i.e., $N_k \in \mathcal{I}$), and M_k represents the possible bit clock values at the symbol clock instant k. The transition probabilities on this model are given by:

$$\begin{aligned} &\forall (e_i, e_j) \in \mathcal{I} \times \mathcal{I}, \forall (m, m') \in \mathbb{N} \times \mathbb{N} \\ &\mathbb{P}(N_k = e_i, M_k = m' \,|\, N_{k-1} = e_j, M_{k-1} = m) = \\ &\begin{cases} \mathbb{P}(N_k = e_i \,|\, N_{k-1} = e_j) & \text{if } m' - m = \overline{b_t} \\ 0 & \text{otherwise,} \end{cases} \end{aligned} \tag{2}$$

where $\mathbb{P}(N_k = e_i \,|\, N_{k-1} = e_j)$ are deduced from the source statistics. This model allows us to integrate the side information brought by $\mathbf{Y}$ in the decoding trellis. To take into account this additional information, the branch metric $\gamma_k(N_k, M_k \,|\, N_{k-1}, M_{k-1})$ of a transition, triggered by symbol s, in the trellis is modified as follows :

$$\begin{aligned} &\gamma_k(N_k = e_i, M_k = m' \,|\, N_{k-1} = e_j, M_{k-1} = m) = \\ &\mathbb{P}(N_k = e_i, M_k = m' | N_{k-1} = e_j, M_{k-1} = m) \\ &\times \mathbb{P}(X_m^{m'} = b_t) \times \mathbb{P}(S_k = s \,|\, Y_k), \end{aligned} \tag{3}$$

where $\mathbb{P}(X_m^{m'} = b_t)$ is computed from $\mathbf{X}$ (taking into account the punctured bits in $\mathbf{X}$) and $\mathbb{P}(S_k = s \,|\, Y_k)$ is computed from the correlation factor π between $\mathbf{S}$ and $\mathbf{Y}$.

The BCJR algorithm, is applied on the state model (N_k, M_k) defined above. For all state $v = (n, m), n \in \mathcal{I}, 1 \leq m \leq L(\mathbf{X})$ of the trellis, the following probability functions are computed:

$$\alpha_k(v) = \mathbb{P}(\mathcal{V}_k = v; \mathbf{Y}_1^k) \tag{4}$$

$$\beta_k(v) = \mathbb{P}(\mathbf{Y}_{k+1}^{L(\mathbf{S})} \,|\, \mathcal{V}_k = v), \tag{5}$$

for $1 \leq k \leq L(\mathbf{S})$. Let us define, for every symbol s of the alphabet, the set $\Omega(s)$ of state pairs ($v = (n, m), v' = (n', m')$) in the decoding trellis, such that the input symbol of the transition from v to v' is equal to s. Then, the symbol marginal probabilities on the trellis defined by the states $\mathcal{V}_k$ are obtained by:

$$\begin{aligned} &\forall k \in [1, L(\mathbf{S})], \forall s \in \mathcal{A}, \\ &\mathbb{P}(S_k = s \,|\, X_1^{L(\mathbf{X})}; Y_1^{L(\mathbf{S})}) \\ &\propto \sum_{(v,v') \in \Omega(s)} \alpha_k(v)\, \beta_{k+1}(v')\, \gamma_{k+1}(v' \,|\, v), \end{aligned} \tag{6}$$

where $\gamma_{k+1}(v' \,|\, v)$ is calculated from Eqn.3.

The BCJR algorithm applied on this state model allows the computation of posterior symbol marginals, and hence the minimization of the symbol error rate.

III. Iterative structures for distributed quasi-arithmetic coding

In this section, two iterative structures involving QA codes for distributed source coding are described. The first one is a parallel concatenation of two QA codes, and the second one is a serial concatenation of a QA code and a convolutional code (CC). The encoders and decoders of these structures are detailed hereafter. Some decoding performance of these structures are given in the next section.

A. Parallel QA-QA structure

The encoder and decoder of the parallel QA-QA structure are given in Fig. 3-a) and 3-b) respectively. This structure is similar to a parallel turbo-code where the convolutional codes are replaced by QA codes. The input symbol stream S and its interleaved version are encoded with two QA codes leading to the bit-streams X_1 and X_2 respectively. These bit-streams are punctured to reach the desired transmission rate. The puncturing is a regular puncturing scheme as detailed above. X_1 and X_2 are then transmitted over an ideal channel. The side information Y is assumed to be available at the decoder. Note that the interleaver in this structure is a random interleaver of length $L(\mathbf{S})$. Note also that the QA codes are adapted to the source probability p of the input message. Hence, if the source is a first order memory source with a given correlation factor ρ_m, the first QA code will be adapted to that source. However, due to the interleaver in the second branch of the encoder, the input of the second QA code is a memoryless binary source with probability p. The second QA code will hence be adapted to that stationary source.

At the decoder side, both QA decoders make use of the corresponding bit-stream (X_1 or X_2), the side information (interleaved or not) and the extrinsic probability coming from the other QA decoder. At the output of a QA decoder, a symbol a posteriori probability (APP) is computed as explained in the previous section. In order to compute the symbol extrinsic probability at the output a QA decoder, the a priori information on the symbols are removed from this symbol APP. Here, the stationary probability of the source (p), the side information and the extrinsic probability coming from the other BCJR decoder are removed from the symbol APP. This extrinsic probability is then provided to the other QA decoder. This information is used in a similar way as the side information Y in the decoding algorithm : it is multiplied in the branch metric of the transitions (cf. Eqn.(3)).

B. Serial QA-CC structure

The encoder and decoder of the serial QA-CC structure are given in Fig. 4-a) and . 4-b) respectively. The encoder is a serial concatenation of a QA encoder and a convolutional code. The input symbol stream $\mathbf{S}$ is fed into the QA encoder leading to a bit-stream X_3, which after being interleaved is encoded using a recursive systematic convolutional code. The resulting bit-stream is denoted X_4. The systematic bits in X_4 are punctured together with some parity bits in order to reach the desired transmission rate.

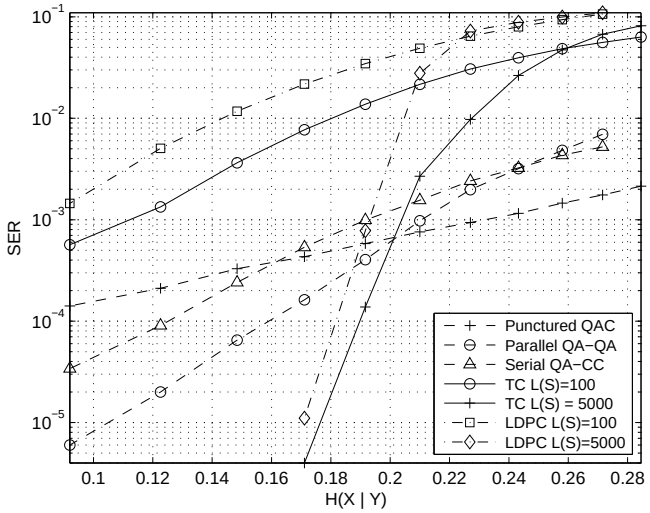

Fig. 5. Performance comparison of DSC schemes at an overall rate of 0.4 bps for a memoryless source with $p = 0.9$

At the decoder side, the CC decoder is based on a BCJR algorithm taking as input the bits of X_4, whereas the QA decoder takes as input the side information Y. As an additional information, both decoders exchange their extrinsic probabilities. The extrinsic probability at the output of the CC decoder is an information on the bits of X_3. This information is used by the QA decoder described in the previous section. The term $\mathbb{P}(X_m^{m'} = b_t)$ in (3) is replaced by $\mathbb{P}(Extr(X_m^{m'}) = b_t)$. The extrinsic probability out of the QA decoder is used by the CC decoder as an information on its input bits.

IV. Simulation results

The DSC schemes described above have been applied to both memoryless sources and first-order memory sources, assuming that the correlated side information Y is available at the decoder.

In the first experiment, we have considered memoryless sources. Two different sources have been used, the first one with $p = 0.9$ and the second with $p = 0.8$. The entropy of these two sources is 0.4690 and 0.7219 respectively. The proposed DSC schemes have been applied for sequences of length $L(\mathbf{S}) = 100$ symbols. The SER at the output of the decoder is averaged over 10^5 realizations. The average rate at the input of the ideal channel is set to 0.4 bits per input symbol (bps). The respective symbol error rates at the output of each decoder for these two different sources are plotted for different values of the conditional entropy $H(Y \mid X)$ in Fig. 5 and 6. In this figures, we have also depicted the decoding performance of punctured turbo codes for the same simulation parameters. The considered turbo code is a parallel concatenation of two (21,37) octal convolutional codes. The interleavers of the turbo codes are randomly chosen, and 15 iterations have been computed in order to evaluate the decoding performance of these codes. Note that, the source probabilities ($p = 0.9$ or $p = 0.8$) have been accordingly integrated to the turbo

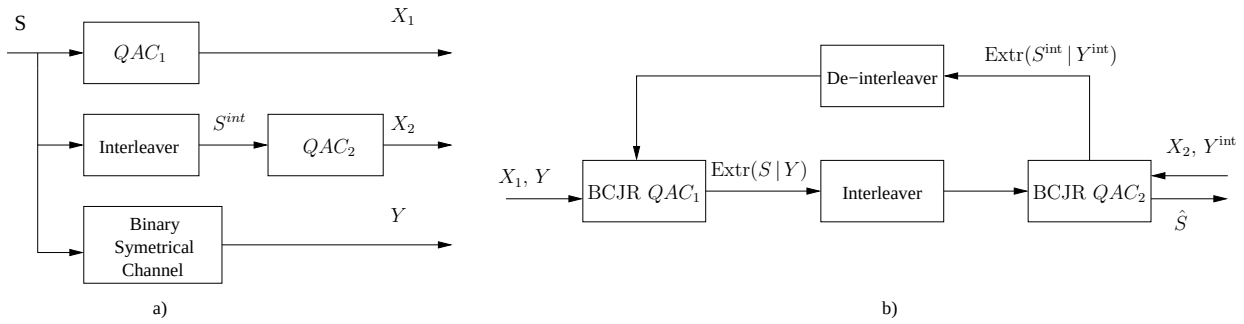

Fig. 3. Encoder a) and decoder b) of the parallel QA-QA structure

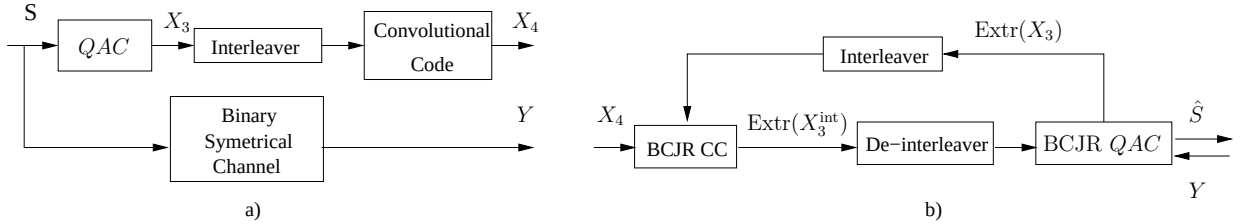

Fig. 4. Encoder a) and decoder b) of the serial QA-CC structure

decoder, as this additional information is also available when QA codes are used. The performance of LDPC codes of [22] is also depicted on Fig. 5, and the one of overlapped quasi-arithmetic codes of [13] are given in Fig. 6.

At the overall rate of 0.4 bps, punctured QA codes offer better performance than punctured turbo codes for the two considered input sources and for the same sequence lengths. This can be explained by the fact that turbo codes are less efficient for short sequences. The proposed iterative structures reduce the SER when the correlation between S and Y is relatively high. The same punctured turbo codes have also been used for sequences of $L(\mathbf{S}) = 5000$ symbols. In that case, as it can be seen in Fig. 5, the performance of these codes is significantly improved. Indeed, the performance of turbo codes are improved when the constituent interleaver of the code is longer. The same conclusions can be drawn with LDPC codes.

In Fig. 5, the Slepian-Wolf bound is equal to $H(X \mid Y) = 0.4$ bps. It means that for $p = 0.9$, the distance to the limit (for a bit-error rate of 10^{-5}) is equal to 0.29 bps. If a distortion of 10^{-3} is tolerated, the binary Winer-Ziv limit [5] is obtained by shifting to the right the Slepian-Wolf bound by about 0.011 bps (corresponding to the entropy of the tolerated distortion). In that case, the distance to the limit is about 0.19 bps. Note that when the input source is not uniform, the Slepian-Wolf bound is shifted to the right, as more information is available. This explains why the distance to the bound is a bit large here.

We have then considered first order memory sources. The sources are defined for $p = 0.9$ and a memory correlation factor $\rho_m = 0.9$. The entropy of this source is 0.1164. The QA code adapted to that source leads to an average compression factor of 0.125. The proposed DSC schemes have been applied for sequences of $L(\mathbf{S}) = 100$ symbols at an overall rate of 0.09 bps. The decoding performance in terms of SER of the proposed DSC schemes are depicted in Fig. 7. These results are compared with a turbo code for the same parameters. The convolutional codes of this turbo code are the same as in the first experiment (i.e., (21,37) octal convolutional code). The decoder of the turbo code consists in two BCJR algorithms which exchange their extrinsic information. The first BCJR takes into account the memory of the source, whereas the second decoder can only take into account the source probability (p) because this decoder takes as input an interleaved version of the original message (that contains memory). Hence, the memory of the source is broken due to the interleaver in the second branch of the turbo code. The

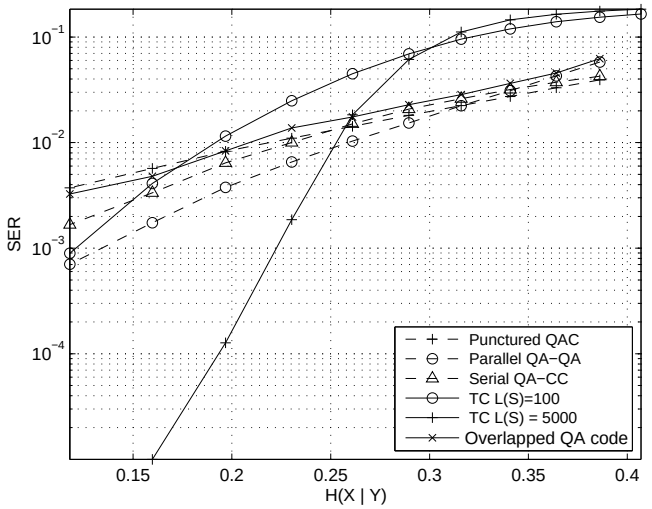

Fig. 6. Performance comparison of DSC schemes at an overall rate of 0.4 bps for a memoryless source with $p = 0.8$

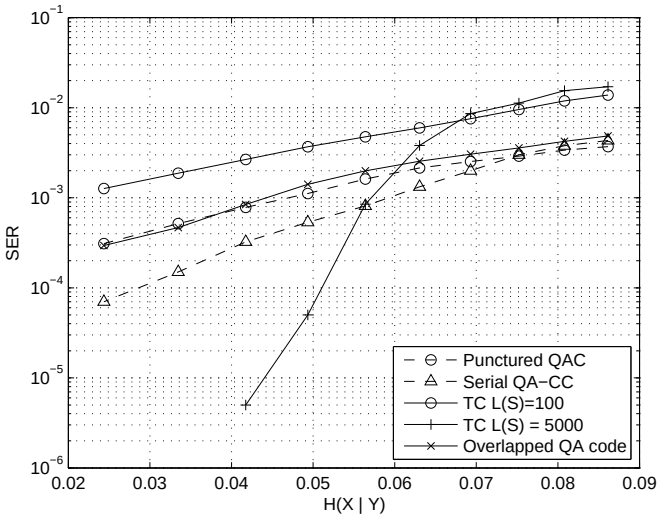

Fig. 7. Performance comparison of DSC schemes at an overall rate of 0.09 bps for a first order memory source $p = 0.9$ $\rho_m = 0.9$

performance of overlapped QA codes is also depicted on this graph for the same simulation parameters.

It can be seen in Fig. 7 that for short sequences, the punctured QA code performs better than the turbo code based DSC scheme and the serial QA-CC structure improves the performance. However, for longer sequences, turbo codes are more efficient in terms of SER. Note that the parallel QA-QA structure is not depicted here. For sources with memory, the memory is broken by the interleaver in the second branch of the structure. This implies that a memoryless QA code has to be used in the second branch, hence increasing the rate at the output of the QA code. This structure is not convenient for first-order memory sources.

V. Concluding remarks

We have proposed in this paper a DSC scheme based on punctured quasi-arithmetic codes. These codes can be defined with finite state machines for both memoryless and first-order memory sources, hence allowing to decode them using an optimal BCJR algorithm. The compression capability of these codes together with their capability to integrate extra information on the source probabilities are exploited by the proposed DSC scheme. Simulations reveal that this scheme is able to provide efficient decoding performance for short sequences compared to well-known DSC schemes based on channel codes, particularly when the source probability is very asymmetric and when the memory correlation factor of the source is relatively high. This reveals that QA codes are better suited than channel codes to deal with asymmetric sources.

References

[1] R. Puri and K. Ramchandran, "PRISM: A new robust video coding architecture based on distributed compression principles," in *Allerton Conf. on Comm. Control and Computing*, Oct. 2002.

[2] A. Aaron, R. Zhang, and B. Girod, "Wyner-ziv coding of motion video," in *Asilomar Conf. on Signals, Systems and Computers*, Nov. 2002.

[3] X. Artigas, J. Ascenso, M. Dalai, S. Klomp, D. Kubasov, and M. Ouaret, "The discover codec: Architecture, techniques and evaluation," in *Picture Coding Symposium*, Lisbonne, Portugal,, Nov. 2007.

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

[5] A. Wyner and J. Ziv, "The rate distorsion function for source coding with side information at the decoder," *IEEE Trans. Inform. Theory*, vol. 22, pp. 1–10, Jan. 1976.

[6] S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): design and construction," in *Proc. Data Compression Conf., DCC*, March 1999, pp. 158–167.

[7] J. Bajcsy and P. Mitran, "Coding for the slepian-wolf problem with turbo codes," in *Proc. IEEE Globecom*, 2001, pp. 1400–1404.

[8] A. Aaron and B. Girod, "Compression with side information using turbo codes," in *Proc. Data Compression Conf., DCC*, 2002, pp. 252–261.

[9] J. Garcia-Frias and Y. Zhao, "Compression of correlated binary sources using turbo codes," *IEEE Commun. Lett.*, vol. 5, pp. 417–419, Oct. 2001.

[10] A. Liveris, Z. Xiong, and C. Georghiades, "Compression of binary sources with side information at the decoder using ldpc codes," *IEEE Commun. Lett.*, vol. 6, pp. 440–442, Oct. 2002.

[11] D. V. Renterghem, X. Jaspar, B. Macq, and L. Vanderdorpe, "Distributed coding with optimized irregular turbo codes," in *Proc. Intl. Conf. Commun., ICC*, Glasgow, Scotland, June 2007, pp. 963–968.

[12] Q. Zhao and M. Effros, "Lossless and near lossless source coding for multiple access network," *IEEE Trans. Inform. Theory*, vol. 49, no. 1, pp. 112–128, Jan. 2003.

[13] X. Artigas, S. Malinowski, C. Guillemot, and L. Torres, "Overlapped quasi-arithmetic codes for distributed video coding," in *Proc. Intl. Conf. Image Processing, ICIP*, San Antonio, USA, 2007.

[14] M. Grangetto, E. Magli, and G. Olmo, "Distributed arithmetic coding," *IEEE Commun. Lett.*, vol. 11, pp. 883–885, Nov. 2007.

[15] M. Sartipi and F. Fekri, "Distributed source coding using short to moderate length rate-compatible LDPC codes : The entire slepian-wolf rate region," *IEEE Trans. Commun.*, vol. 56, no. 3, pp. 400–411, March 2008.

[16] J. Ha, J. Kim, D. Klinc, and S. McLaughlan, "Rate-compatible punctured low-density parity- check codes with short block lengths," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 728–738, Feb. 2006.

[17] S. Ben-Jamaa, C. Weidmann, and M. Kieffer, "Asymptotic error-correcting performance of joint source-channel schemes based on arithmetic coding," in *Proc. MMSP*, Victoria, Canada, October 2006, pp. 262–266.

[18] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, pp. 284–287, Mar. 1974.

[19] 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.

[20] 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.

[21] J. Hagenauer, J. Barros, and A. Schaefer, "Lossless turbo source coding with decremental redundancy," in *Proc. Intl. Conf. Source and Channel Coding, SCC*, Erlangen, Germany, January 2004.

[22] D. Varodayan, A. Aaron, and B. Girod, "Rate adaptative codes for distributed source coding," *IEEE Trans. Image Processing*, vol. 86, no. 11, pp. 3123–3130, Nov. 2006.