# Toward constructive Slepian-Wolf coding schemes

Christine Guillemot, Aline Roumy

# Toward constructive Slepian-Wolf coding schemes

C. Guillemot, A. Roumy

2009

## 1 Introduction

This chapter deals with practical solutions for the Slepian Wolf (SW) coding problem, which refers to the problem of lossless compression of correlated sources with coders which do not communicate. Here, we will consider the case of two binary correlated sources $X$ and $Y$, characterized by their joint distribution. If the two coders communicate, it is well known from Shannon's theory that the minimum lossless rate for $X$ and $Y$ is given by the joint entropy $H(X,Y)$. Slepian and Wolf established in 1973 [30] that this lossless compression rate bound can be approached with a vanishing error probability for infinitely long sequences, even if the two sources are coded separately, provided that they are decoded jointly and that their correlation is known to both the encoder and the decoder. Hence, the challenge is to construct a set of encoders which do not communicate and a joint decoder which can achieve the theoretical limit.

This chapter gives an overview of constructive solutions both for the asymmetric and the non asymmetric SW coding problems. Asymmetric SW coding refers to the case where one source, for example $Y$, is transmitted at its entropy rate and used as side information to decode the second source $X$. Non asymmetric SW coding refers to the case where both sources are compressed at a rate lower than their respective entropy rates. Sections 2 and 3 recall the principles and then describe practical schemes for asymmetric and symmetric coding respectively. Practical solutions for which the compression rate is *a priori* fixed according to the correlation between the two sources are first described. In this case, the correlation between the two sources needs to be known - or estimated - at the transmitter. Rate-adaptive schemes in which the SW code is incremental are then presented. This chapter ends with Section 4 covering various advanced SW coding topics such as the design of schemes based on source codes and the generalization to the case of non-binary sources, and to the case of $M$ sources.

# 2 Asymmetric SW coding

The SW region for two discrete sources is an unbounded polygon with two corner points (see points A and B in Figure 1 (b)). At these points, one source (say $Y$ for point A) is compressed at its entropy rate and can therefore be reconstructed at the decoder independently of the information received from the other source $X$. The source $Y$ is called the *side information* (SI) (available at the decoder only). $X$ is compressed at a smaller rate than its entropy. More precisely, $X$ is compressed at the conditional entropy $H(X|Y)$ and can therefore only be reconstructed if $Y$ is available at the decoder. The sources $X$ and $Y$ play different roles in this scheme, and therefore the scheme is usually referred to as *asymmetric SW coding*.

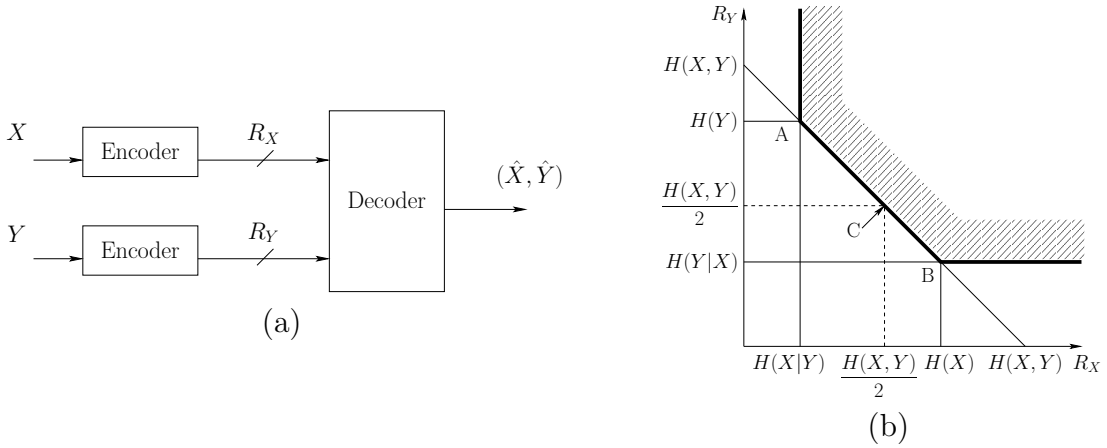



Figure 1: Distributed source coding of statistically dependent i.i.d. discrete random sequences $X$ and $Y$. Set-up (left); achievable rate region (right).

## 2.1 Principle of asymmetric SW coding

### 2.1.1 The syndrome approach

Because of the random code generation in the SW theorem, the SW proof is non-constructive. However, some details in the proof do give insights on how to construct SW bound achieving codes. More precisely, the proof relies on binning – that is a partition of the space of the source sequences into subsets (or bins) such that the vectors in the bins are as far apart (or as much "jointly non-typical") as possible (see Figure 2). In 1974 [40], Wyner suggested to construct the bins as cosets of a binary linear code and showed the optimality of this construction. More precisely, if the linear block code achieves the capacity of the binary symmetric channel (BSC) that models the correlation between the two sources, then this capacity achieving channel code can be turned into a SW achieving source code.

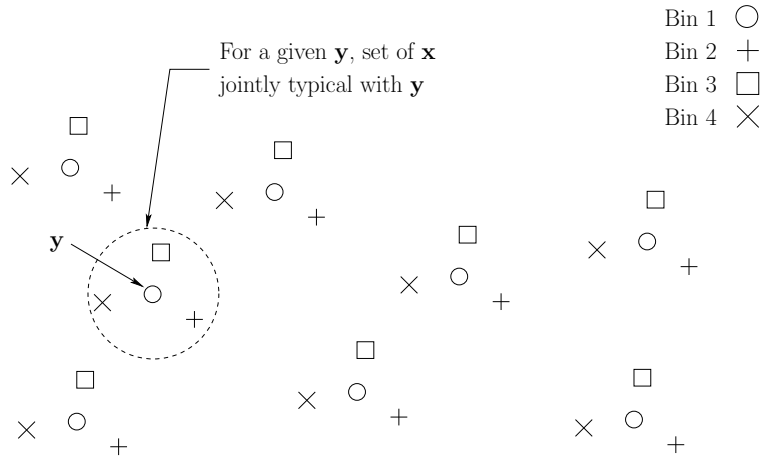



Figure 2: The partition of the sequences $\mathbf{x}$ into four subsets (called bins) such that the sequences in each bin are as far apart as possible. The set of sequences $\mathbf{x}$ jointly typical with a given $\mathbf{y}$ are as "different" as possible: here they are elements of different bins.

In order to present Wyner's algebraic binning scheme, we first review some notations on binary linear codes. A binary $(n, k)$ code $\mathcal{C}$ is defined by an $(n-k) \times n$ parity-check matrix $H$, and contains all the $n$-length vectors[1] $\mathbf{x}$ s. t. $\mathbf{x}H^T = \mathbf{0}$:

$$\mathcal{C} = \left\{\mathbf{x} : \mathbf{x}H^T = \mathbf{0}\right\}.$$

Notice that the code $\mathcal{C}$ is equivalently defined by a $k \times n$ generator matrix $G$ such that $\mathcal{C} = \{\mathbf{x} : \mathbf{x} = \mathbf{u}G\}$, where $\mathbf{u}$ is a $k$-bit vector.

If $H$ has full row rank, then the rate of the channel code is $k/n$. Moreover, for a good code, all the vectors of the code (called words or codewords) have maximum Hamming distance. The code partitions the space containing $2^n$ sequences into $2^{n-k}$ *cosets* of $2^k$ words with maximum Hamming distance. Each coset is indexed by an $(n-k)$-length *syndrome*, defined as $\mathbf{s} = \mathbf{x}H^T$. In other words, all sequences in a coset share the same syndrome: $\mathcal{C}_\mathbf{s} = \left\{\mathbf{x} : \mathbf{x}H^T = \mathbf{s}\right\}$. Moreover, as a consequence of the linearity of the code, a coset results from the translation of the code by any representative of the coset:

$$\forall \mathbf{v} \in \mathcal{C}_\mathbf{s}, \mathcal{C}_\mathbf{s} = \mathcal{C} \oplus \mathbf{v}. \tag{1}$$

A geometric approach to binary coding visualizes binary sequences of length $n$ as vertices of an $n$-dimensional cube. In Figure 3, the code $\mathcal{C}$ (the set of codewords or the set of vectors of syndrome 0 that is represented with $\bullet$) is a subspace of $\{0,1\}^3$, whereas the coset with syndrome 1, denoted $\mathcal{C}_1$ and represented with $\circ$, is an affine subspace parallel to $\mathcal{C}$. Notice that properties of a subspace and the linearity property (1) are satisfied (to verify this, all operations have to be performed over the finite field of order 2).

[1] All vectors are line vectors, and $^T$ denotes transposition. Moreover, $\oplus$ denotes the addition over the finite field of order 2 and $+$ the addition over the real field.

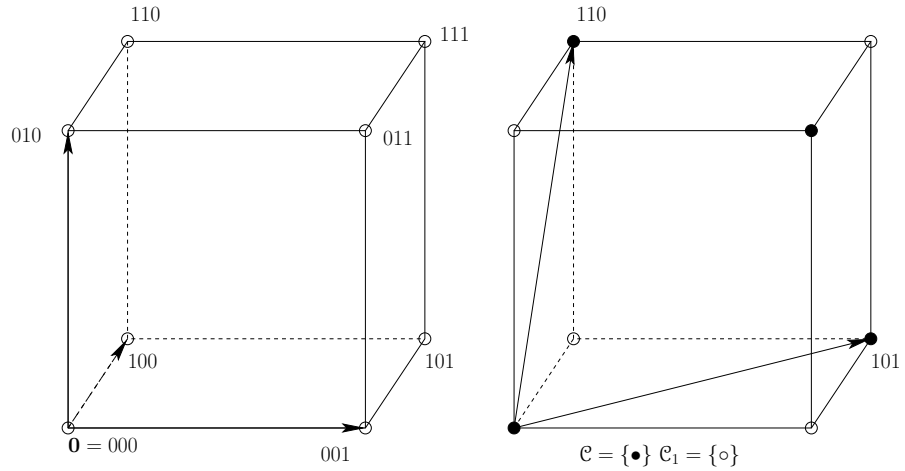



Figure 3: Construction of a binning scheme with a linear block code in $\{0,1\}^3$. The cube and the basis vector in $\{0,1\}^3$ (left); a code defined by the parity check matrix $H = (111)$ or equivalently the generator matrix $G = \binom{110}{101}$ (right).

If a codeword $\mathbf{x}$ is sent over a BSC with cross-over probability $p$ and error sequence $\mathbf{z}$, the received sequence is $\mathbf{y} = \mathbf{x} + \mathbf{z}$. Maximum likelihood (ML) decoding over the BSC, searches for the closest codeword to $\mathbf{y}$ with respect to the Hamming distance $d_H(.,.)$:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}:\mathbf{x}\in\mathcal{C}} d_H(\mathbf{x}, \mathbf{y}).$$

This can be implemented with *syndrome decoding*. This decoding relies on a function $f : \{0,1\}^{n-k} \to \{0,1\}^n$, that computes for each syndrome a representative called the *coset leader* that has minimum Hamming weight. Thus, a syndrome decoder first computes the syndrome of the received sequence: $\mathbf{s} = \mathbf{y}H^T$, then the coset leader $f(\mathbf{s})$. This coset leader is the ML estimate of the error pattern $\mathbf{z}$. Finally, the ML estimate of $\mathbf{x}$ is given by $\hat{\mathbf{x}} = \mathbf{y} \oplus f(\mathbf{y}H^T)$.

In order to use such a code for the asymmetric SW problem, [40] suggests to construct bins as cosets of a capacity-achieving parity-check code. Let $\mathbf{x}$ and $\mathbf{y}$ be two correlated binary sequences of length $n$. These sequences are the realizations of the sources $X$ and $Y$. The encoder computes and transmits the syndrome of $\mathbf{x}$: $\mathbf{s} = \mathbf{x}H^T$. The sequence $\mathbf{x}$ of $n$ input bits is thus mapped into its corresponding $(n-k)$ syndrome bits, leading to a compression ratio of $n : (n-k)$. The decoder, given the correlation between the sources $X$ and $Y$ and the received coset index $\mathbf{s}$, searches for the sequence in the coset that is closest to $\mathbf{y}$. In other words, ML decoding is performed in order to retrieve the original sequence $\mathbf{x}$:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}:\mathbf{x}\in\mathcal{C}_\mathbf{s}} d_H(\mathbf{x}, \mathbf{y}). \tag{2}$$

Note that the maximization is performed in a set of vectors with syndrome that may not be $\mathbf{0}$. Therefore, the classical ML channel decoder has to be adapted in order to be able to

enumerate all vectors in a given coset $\mathcal{C}_{\mathbf{s}}$. This adaptation is straightforward if syndrome decoding is performed. In this case, the decoder can first retrieve the error pattern (since for each syndrome, the corresponding coset leader is stored) and add it to the SI $\mathbf{y}$.

Syndrome decoding is used here as a mental representation rather than an efficient decoding algorithm, since it has complexity $O(n2^{n-k})$. There exists codes with linear decoding complexity and we will detail these constructions in Section 2.2. In this section, we have presented a structured binning scheme for the lossless source coding problem with SI at the decoder. However, the principle is quite general and can be applied to a large variety of problems as shown in [41].

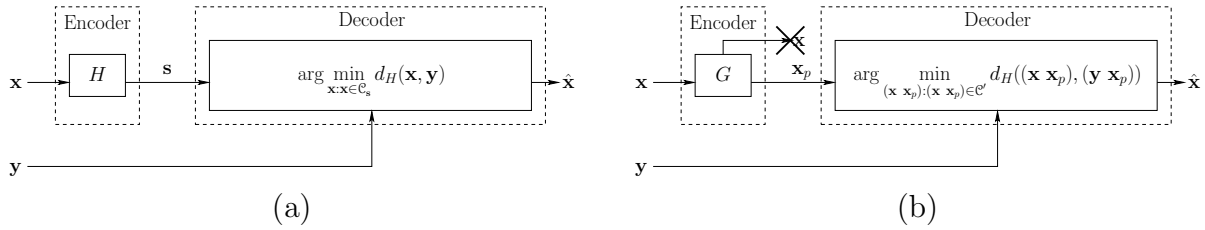



Figure 4: Asymmetric SW coding. The syndrome approach (left); the parity approach (right).

### 2.1.2 The parity approach

The above detailed syndrome approach is optimal [40], however it may be difficult to construct rate-adaptive codes by puncturing the syndrome. Therefore, another approach called *parity approach* has been proposed [9][2], [1].

Let $\mathcal{C}'$ be an $(n, 2n-k)$ systematic binary linear code, defined by its $(2n-k) \times n$ generator matrix $G = (I\ P)$:

$$\mathcal{C}' = \{\mathbf{x}G = (\mathbf{x}\ \mathbf{x}_p) : \mathbf{x} \in \{0,1\}^n\}.$$

The compression of the source $X$ is achieved by transmitting only the parity bits $\mathbf{x}_p$ of the source sequence $\mathbf{x}$ (see Figure 4). The systematic bits $\mathbf{x}$ are not transmitted. This leads to a compression ratio $n : (n-k)$. Here again, the correlation between the source $X$ and the SI $Y$ is modeled as a "virtual" channel, where the pair $(\mathbf{y}\ \mathbf{x}_p)$ is regarded as a noisy version of $(\mathbf{x}\ \mathbf{x}_p)$. The channel is therefore a parallel combination of a BSC and a perfect channel. The decoder corrects the "virtual" channel noise, and thus estimates $\mathbf{x}$ given the parity bits $\mathbf{x}_p$ and the SI $\mathbf{y}$ regarded as a noisy version of the original sequence $\mathbf{x}$. Therefore, the usual ML decoder must be adapted to take into account that some bits of the received sequence $(\mathbf{x}_p)$ are perfectly known. We will detail this in Section 2.2.

Interestingly, both approaches (syndrome or parity) are equivalent if the generator matrix (in the parity approach) is the concatenation of the identity matrix and the parity

[2]Note that this paper introduces the principle of the parity approach in the general case of non-asymmetric coding.

check matrix used in the syndrome approach: $G = (I\ H)$. However, the two codes $\mathcal{C}$ and $\mathcal{C}'$ (defined by $H$ and $G$ respectively) are not the same. They are not even defined in the same space ($\mathcal{C}$ is a subspace of $\{0,1\}^n$, whereas $\mathcal{C}'$ is a subspace of $\{0,1\}^{2n-k}$). Figure 4 compares the implementation of the syndrome and parity approaches.

## 2.2 Practical code design based on channel codes

### 2.2.1 The syndrome approach.

Practical solutions based on the syndrome approach first appeared in a scheme called DISCUS [21]. For block codes [21], syndrome decoding is performed. For convolutional codes, the authors in [21] propose to apply the Viterbi decoding on a modified trellis. In order to solve the problem (2), the decoder is modified such that it can enumerate all the codewords in a given coset. The method uses the linearity property (1). For systematic convolutional codes, a representative of the coset is the concatenation of the $k$-length zero vector and the $n-k$-length syndrome $\mathbf{s}$. This representative is then added to all the codewords labeling the edges of the trellis (see Figure 5). Note that the states of the trellis depend on the information bits, and thus on the systematic bits, only. Thus, there exists one trellis section per syndrome value. The decoder, knowing the syndrome $\mathbf{s}$, searches for the sequence with that particular syndrome that is closest to $\mathbf{y}$. First it builds the complete trellis. Each section of the trellis is determined by the syndrome. Once the whole trellis is built, the Viterbi algorithm chooses the closest sequence to the received $\mathbf{y}$.

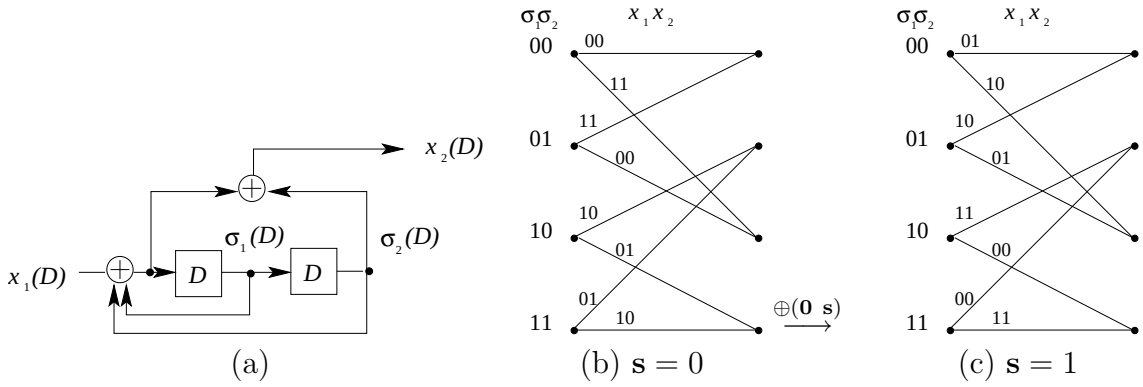



Figure 5: Block diagram (a) and trellises (b,c) for the systematic convolutional code $H = [1\frac{5}{7}]$. The diagram (a) defines the states $\sigma_1(D)$ $\sigma_2(D)$ of the trellis, where $D$ is a dummy variable representing a time offset. The trellis section in (b) corresponds to the syndrome value $\mathbf{s} = 0$ and is called the *principal trellis*. The *complementary trellis* corresponds to $\mathbf{s} = 1$ and is obtained by adding $(\mathbf{0}\ \mathbf{s})$ to all the codewords labeling the edges of the principal trellis.

A variation of the above method is proposed in [36], where the translation by a coset

representative is performed outside the decoder (see Figure 6). First the representative ($\mathbf{0}$ $\mathbf{s}$) is computed (this step is called "inverse syndrome former") and added to $\mathbf{y}$. Then the decoding is performed (in the coset of syndrome $\mathbf{0}$), and finally the representative is retrieved from the output of the decoder. The advantage of this method is to use a conventional Viterbi decoder without a need to modify it. This method can also be applied to turbo codes [36]. In this case, two syndromes ($\mathbf{s}_1$, $\mathbf{s}_2$) are computed, one for each constituent code. A representative of the coset is ($\mathbf{0}$ $\mathbf{s}_1$ $\mathbf{s}_2$).

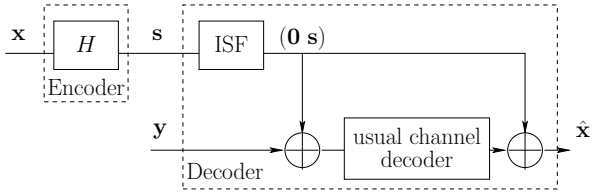



Figure 6: Implementation of the syndrome approach with a usual channel decoder. ISF stands for inverse syndrome former and computes a representative of the coset with syndrome $\mathbf{s}$.

The authors in [25] propose a SW scheme based on convolutional codes and turbo codes that can be used for any code (not only systematic convolutional code). Rather than considering the usual trellis based on the generator matrix of the code, the decoder is based on a syndrome trellis. This trellis has first been proposed for binary linear block codes [39], and was then generalized to convolutional codes [29]. Such a trellis can enumerate a set of sequences in any coset. More precisely, a section corresponds to the reception of $n$ bits of $\mathbf{x}$ and $n-k$ syndrome bits (see Figure 7). Thus, there exists $2^{n-k}$ possible trellis sections, one for each syndrome value. The decoder, knowing the syndrome $\mathbf{s}$, searches for the sequence with that particular syndrome which is closest to $\mathbf{y}$. First it builds the complete trellis. Each section of the trellis is determined by the syndrome. Once the whole trellis is built, the Viterbi algorithm is performed and chooses the closest sequence to the received $\mathbf{y}$.

For LDPC codes, the belief propagation decoder can be adapted to take into account the syndrome [17]. More precisely, the syndrome bits are added to the graph such that each syndrome bit is connected to the parity check equation it is related to. The update rule at a check node is modified in order to take into account the value of the syndrome bit (known perfectly at the decoder).

### 2.2.2 The parity approach

The parity approach presented in Section 2.1.2 has been implemented using various channel codes, for instance turbo-codes [1, 5]. Consider a turbo encoder formed by the parallel concatenation of two recursive systematic convolutional (RSC) encoders of rate $(n-1)/n$ separated by an interleaver (see Figure 8). For each sequence $\mathbf{x}$ (of length $n$) to be compressed, two sequences of parity bits $\mathbf{x}_p = (\mathbf{x}_p^1\ \mathbf{x}_p^2)$ are computed and transmitted without

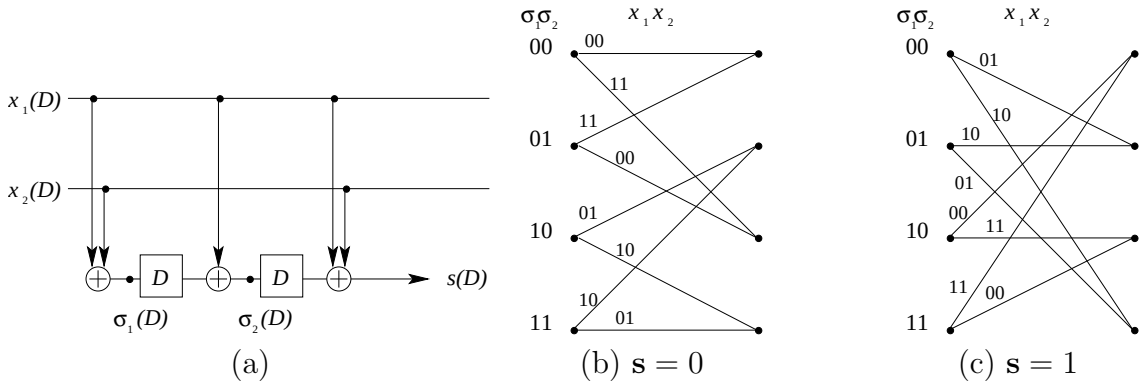



Figure 7: Block diagram (a) and syndrome trellises (b-c) for the rate 1/2 convolutional code $H = [5\ 7]$. The states $\sigma_1(D)\ \sigma_2(D)$ are defined in the block diagram (a). Two trellis sections are obtained: one for the syndrome value $\mathbf{s} = 0$ (b) and one for $\mathbf{s} = 1$ (c).

any error to the decoder. The turbo code is composed of two symbol Maximum A Posteriori (MAP) decoders. Each decoder receives the (unaltered) parity bits together with the SI $\mathbf{y}$, seen as a noisy version of the sequence $\mathbf{x}$. The concatenation $(\mathbf{y}\ \mathbf{x}_p)$ can be seen as a noisy version of the coded sequence $(\mathbf{x}\ \mathbf{x}_p)$, where the channel is a parallel combination of a BSC and of a perfect channel. Therefore, the usual turbo decoder must be matched to this combined channel. More precisely, the channel transitions in the symbol MAP algorithm which do not correspond to the parity bits are eliminated. Such a scheme achieves an $n : 2$ compression rate.

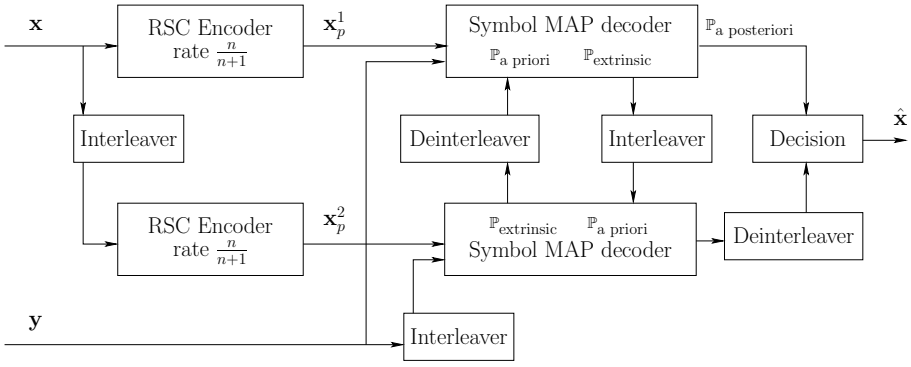



Figure 8: DSC system for two binary correlated sources using punctured turbo codes.

## 2.3 Rate adaptation

Note that in all methods presented so far, in order to select the proper code and code rate, the correlation between the sources needs to be known or estimated at the transmitter before the compression starts. However, for many practical scenarios, this correlation may

vary, and rate adaptive schemes have to be designed. When the correlation decreases, the rate bound moves away from the origin (see Figure 9). The rate of the code can then be controlled via a feedback channel. The decoder estimates the Bit Error Rate (BER) at the output of the decoder with the help of the log-likelihood ratios computed by the channel decoder. If the Bit Error Rate (BER) at the output of the decoder exceeds a given value, more bits are requested from the encoder. In this context of feedback controlled schemes, the code should be *incremental* such that the encoder does not need to re-encode the data. The first bits are kept, and additional bits are only sent upon request. In the following, we present various rate-adaptive methods and specify whether they are incremental or not.

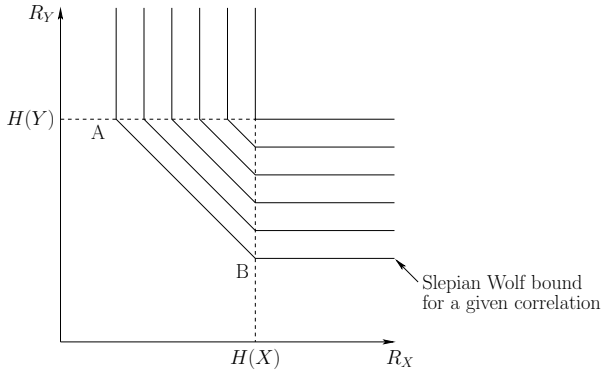



Figure 9: Evolution of the SW region, when the correlation between the sources varies.

### 2.3.1 The parity approach

The parity approach has been originally proposed in order to easily construct rate adaptive schemes (see Section 2.2). At the encoder, the parity bits are punctured (some parity bits are not transmitted) and the decoder compensates for this puncturing, using standard techniques coming from channel coding. The source sequence $\mathbf{x}$ is compressed through some punctured parity bits $\tilde{\mathbf{x}}_p$, and the decoder retrieves the original sequence aided by the SI $\mathbf{y}$. The sequence $(\mathbf{y}\ \tilde{\mathbf{x}}_p)$ can be seen as the output of a channel which is a combination of a perfect channel (unpunctured parity bits), an erasure channel (punctured parity bits) and a BSC channel (correlation between the sources $\mathbf{x}$ and $\mathbf{y}$). The method is by construction incremental.

### 2.3.2 The syndrome approach.

Due to the optimality of the syndrome approach [40], a natural design of a rate-adaptive scheme consists of puncturing the syndrome. However, using puncturing mechanisms leads to performance degradation as reported in [33, 15, 37]. More precisely, it is shown in [33] that the syndrome approach is very sensitive to errors or erasures, which indicates

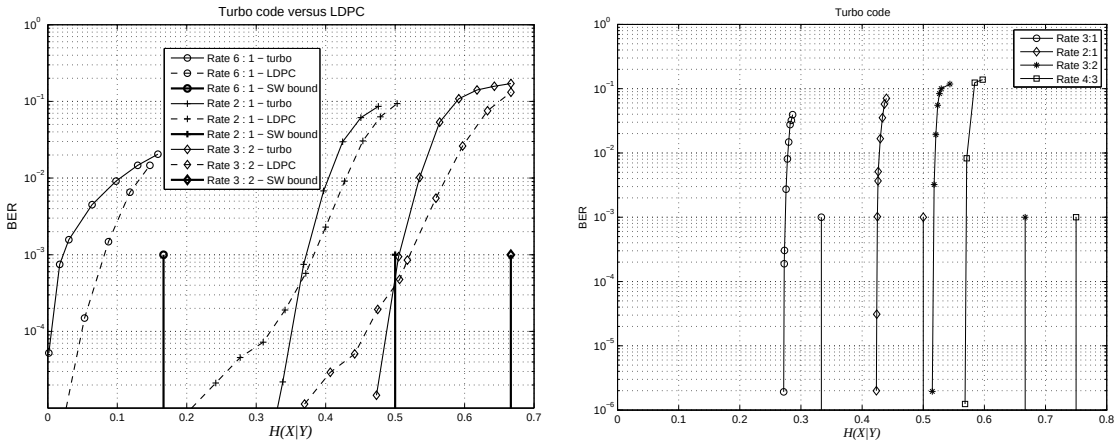
that the puncturing of syndromes will lead to performance degradation. Therefore, the first contributions avoid or compensate for this performance degradation.

A first method proposed in [15] punctures the parity bits rather than the syndrome bits. Each puncturing pattern defines a new code for which a parity check matrix can be computed. Then, one can construct a syndrome former and inverse syndrome former for the new punctured code (presented in Section 2.1.2). This method can be implemented using any code, in particular convolutional and turbo codes. However, the resulting code is not incremental.

A first incremental SW coding scheme based on Serially Concatenated Accumulate (SCA) codes is proposed in [7]. An inner encoder concatenates the first source bit with the modulo-2 sum of consecutive pairs of source bits. The output of the accumulator is interleaved and fed into a so-called base code, which is either an extended Hamming code or a product of single parity check codes, which then computes the syndrome bits. The accumulator code is of rate 1, so that the compression rate is controlled by the base code. The base code is first decoded using a MAP decoder for extended hamming codes or a belief propagation algorithm for product parity check codes. The accumulator code is decoded with a symbol MAP algorithm decoder. A turbo-like algorithm iterates between both decoders.

The authors in [37] investigate the puncturing of LDPC (Low density parity check code) encoded syndromes. To avoid degrading the performance of the LDPC code, the syndrome bits are first protected with an accumulator code before being punctured. Here also, the accumulator code is of rate 1, so that the compression rate is not modified. The combined effect of the puncturing and of the accumulator code is equivalent to merging some rows of the parity check matrix by adding them. This defines a set of parity check matrices, one for each rate. Then, for each parity check matrix, decoding is performed according to the modified sum-product algorithm [17] presented in Section 2.1.1. An interesting feature of this method is that it is incremental. If the merging of rows satisfies a regularity rule (one can only add two rows that have no "1" at the same column position), then the accumulated syndrome value of the new (smallest compression rate) matrix can be computed by adding the previous syndrome value (corresponding to the highest compression rate matrix) to the current one. The performance of the method is shown in Figure 10 (a). Another method is proposed in [14]. Instead of protecting the syndrome bits, the authors combine the syndrome and the parity approaches. Given an input vector $\mathbf{x}$ of length $n$, the encoder transmits $m$ syndrome bits plus $l$ parity bits, defining an $(n+l, n-m)$ linear error correcting code. Rate adaptation is then performed by puncturing the parity bits.

In [25], the optimal decoder for convolutional codes under syndrome puncturing is designed. It is shown that performance degradation due to syndrome puncturing can be avoided without a need to encode (or protect) the syndromes. When syndrome bits are punctured, the optimal decoder should in principle search for the closest sequence in a union of cosets. This union corresponds to all possible syndromes that equal the received syndrome in the unpunctured positions. Therefore, the number of cosets grows exponentially with the number of punctured bits. A brute force decoder consists in




(a) (b)

Figure 10: Syndrome puncturing of an LDPC code [37] and of a turbo code [25]: two rate adaptive schemes. Performance versus the entropy rate $H(X|Y)$ and comparison with the SW bounds. The 1/2-rate constituent code of the turbo code is defined by its parity check matrix $H = (23/33, 35/33)$. The overall compression rate for both schemes is $1:1$, and puncturing leads to various compression rates. (a) Comparison of the syndrome punctured LDPC and turbo code for a blocklength of 2046. (b) Performance of the turbo code for an interleaver of size $10^5$.

performing a ML algorithm in each coset, but then the decoding complexity of such a method would grow exponentially with the number of punctured bits. On the other hand, if the decoder is not modified, it leads to systematic errors, since the search may be performed in the wrong coset. The method proposed in [25] is based on the syndrome trellis presented in Section 2.1.1. Whenever some syndrome bits are punctured, a new trellis section is constructed as the union of the trellis sections compatible with the received punctured syndrome. This enumerates the sequences in the union of cosets. The resulting trellis is called the *super trellis* (see Figure 11) and the Viterbi algorithm is performed on this new trellis. The complexity of the proposed algorithm grows only linearly with the code blocklength and with the number of punctured positions. The authors in [25] also show that this decoder can be applied to turbo-codes. The performance of the method is shown in Figure 10.

# 3 Non-asymmetric SW coding

For the sake of simplicity, we focus again on the two-user setting and we now consider the case where both sources are compressed in order to reach any point of the segment between $A$ and $B$ in Figure 1 (b). For some applications, it may be desirable to vary the rates of each encoder while keeping the total sum-rate constant. This set-up is referred to as *non asymmetric*. It will be said to be *symmetric* when both sources are compressed

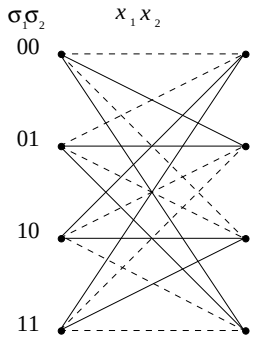



Figure 11: Super trellis for the rate 1/2 convolutional code defined by its polynomial parity check matrix $H(D) = [5\ 7]$. This trellis is the union of the two trellis sections of Figure 7.

at the same rate (point C in Figure 1 (b)). Note that both sources can be compressed at the same rate if and only if $\max(H(X|Y), H(Y|X)) \leq \frac{H(X,Y)}{2} \leq \min((H(X), H(Y))$. Several methods can be used to reach any point of the SW rate bound. The first approach called *time sharing* uses two asymmetric coder/decoder pairs alternatively. Other methods code each input sequence $\mathbf{x}$ and $\mathbf{y}$ with linear channel codes. For each source, part of the information bits plus syndrome or parity bits are transmitted. The decoder then needs to estimate the $n$-length sequences $\mathbf{x}$ and $\mathbf{y}$ knowing the partial information bits, the syndrome or parity bits and the correlation between $\mathbf{x}$ and $\mathbf{y}$.

## 3.1 Time-sharing

Let $\mathbf{x}$ and $\mathbf{y}$ again denote two random binary correlated sequences of source symbols of length $n$. It is assumed that the correlation is defined by a BSC of cross-over probability $p$. All points of the segment between A and B of the SW rate bound are achievable by time sharing. More precisely, a fraction $\alpha$ of samples ($\alpha.n$ samples, where $n$ is the sequence length) is coded at the vertex point $A$, that is, at rates $(H(Y), H(X|Y))$, with the methods described in Section 2, and a fraction $(1-\alpha)$ of samples is then coded at rates $(H(X), H(Y|X))$ corresponding to the corner point $B$ of the SW rate region. This leads to the rates $R_X = \alpha H(X) + (1-\alpha)H(X|Y)$ and $R_Y = (1-\alpha)H(Y) + \alpha H(Y|X)$.

## 3.2 The parity approach

The sequences $\mathbf{x}$ and $\mathbf{y}$ of length $n$ are fed into two linear channel coders (turbo coders [6] or LDPC coders [26]) which produce two subsequences $\mathbf{x}^h = (x_1 \dots x_l)$ and $\mathbf{y}^h = (y_{l+1} \dots y_n)$ of information bits and two sequences $x_1^p \dots x_a^p$ and $y_1^p \dots y_b^p$ of parity bits, where $a \geq (n-l)H(X|Y)$ and $b \geq lH(Y|X)$. The achievable rates for each source are then $R_X \geq \frac{l}{n}H(X) + \frac{a}{n} \geq \frac{l}{n}H(X) + \frac{n-l}{n}H(X|Y)$ and $R_Y \geq \frac{n-l}{n}H(Y) + \frac{b}{n} \geq \frac{n-l}{n}H(Y) + \frac{l}{n}H(Y|X)$. The SW boundary can be approached by varying the ratio $\frac{l}{n}$ between 0 and

1. Unlike the time-sharing approach, the parity bits are computed on the entire input sequences of length $n$. The sequence $(\mathbf{x}^h\mathbf{x}^p)$ is regarded as a noisy version of the sequence $(\mathbf{y}^p\mathbf{y}^h)$. As in the asymmetric set-up, the channel is a parallel combination of a BSC and of a perfect channel. The decoder then needs to estimate $\mathbf{x}$ and $\mathbf{y}$ given $(\mathbf{x}^h\mathbf{x}^p)$ and $(\mathbf{y}^p\mathbf{y}^h)$.

A first practical implementation of the parity approach is proposed in [9] with turbo codes, and considers the particular case of symmetric SW coding. The solution is then extended to reach any point of the SW boundary in [6], [8]. In [8], the two information sequences of length $n$ are encoded with a systematic punctured turbo code. The information bits of the second source are interleaved before being fed to the encoder. All the systematic bits plus a subset of the parity bits are punctured in order to leave only $a$ bits for the sequence $\mathbf{x}$ and $b$ bits for the sequence $\mathbf{y}$. Each turbo decoder proceeds in the standard way, with the difference that part of the bits are received via an ideal channel. After each iteration, the measures obtained as a result of a turbo decoding on each input bit of one sequence, e.g. $\mathbf{x}$, are passed as additional extrinsic information to the turbo decoder used to estimate the sequence $\mathbf{y}$, via the binary symmetric correlation channel (i.e. $\mathbb{P}(y_k = 0) = (1-p)\mathbb{P}(x_k = 0) + p\mathbb{P}(x_k = 1)$).

Another implementation of the parity approach is described in [26] with LDPC codes, and later studied in [27] for short to moderate length codes. The decoder needs to determine an $n$-length sequence from the part of information bits and parity bits received for the sequence $\mathbf{x}$ and from the correlated sequence $\mathbf{y}$. For this, a message-passing algorithm is used by setting the LLR (Log likelihood ratio) of the bits received via the ideal channel to infinity. The LLR of the bits received via the correlation channel are set to $\pm\log(\frac{1-p}{p})$. The LLR for the punctured bits are set to zero. Two sets of degree distributions for the variables nodes are used to account for the fact that information bits are received via the correlation channel whereas the parity bits are received via an ideal channel.

## 3.3 The syndrome approach

Let us consider an $(n, k)$ linear channel code $\mathcal{C}$ defined by its generator matrix $G_{k\times n}$ and its parity-check matrix $H_{(n-k)\times n}$. A syndrome approach was first proposed in [20], based on the construction of two independent linear binary codes $\mathcal{C}^1$ and $\mathcal{C}^2$ with $G_1$ and $G_2$ as generator matrices, obtained from the main code $\mathcal{C}$. More precisely, the generator matrices $G_1$ and $G_2$ of the two subcodes are formed by extracting $m_1$ and $m_2$ lines respectively, where $m_1+m_2 = k$, from the matrix $G$ of the code $\mathcal{C}$. The parity check matrices $H_1$ and $H_2$ are then of size $(n-m_1)\times n$ and $(n-m_2)\times n$ respectively. A geometric interpretation of the code splitting is shown in Figure 12. Each coder sends a syndrome, defined as $\mathbf{s}_x = \mathbf{x}H_1^T$ and $\mathbf{s}_y = \mathbf{y}H_2^T$ respectively, to index the cosets of $\mathcal{C}^1$ and $\mathcal{C}^2$ containing $\mathbf{x}$ and $\mathbf{y}$. The total rate for encoding the input sequences of length $n$ is then $n - m_1 + n - m_2 = 2n - k$ bits. The cosets containing $\mathbf{x}$ and $\mathbf{y}$ are translated versions of the codes $\mathcal{C}^1$ and $\mathcal{C}^2$ (i.e., the cosets having null syndromes) by vectors $\mathbf{t}_x$ and $\mathbf{t}_y$. This translation is illustrated in Figure 12, where the addition by the translation vector is performed over the finite field

$\{0,1\}$. The codes $\mathcal{C}^1$ and $\mathcal{C}^2$ must be chosen such that all pairs $(\mathbf{x},\mathbf{y})$ can be determined uniquely given the two coset indexes. We now explain how to determine a pair $(\mathbf{x},\mathbf{y})$ from a usual ML decoder.

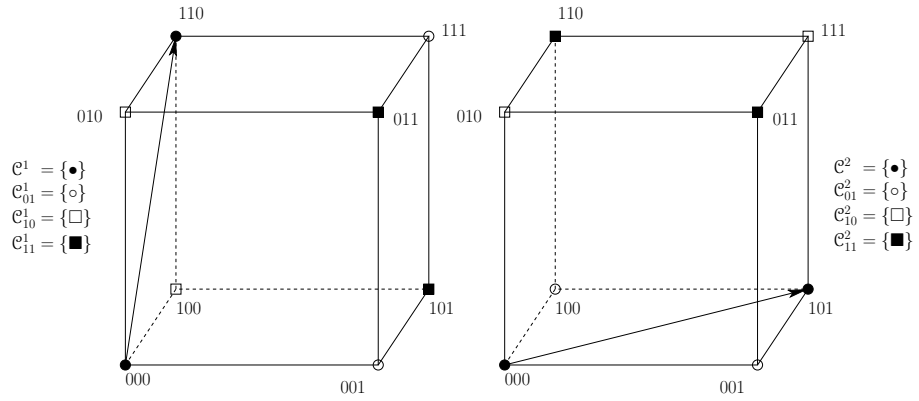



Figure 12: Splitting the code of Figure 3 into two subcodes: code $\mathcal{C}^1$ defined by $G_1 = (110)$ or $H_1 = \binom{110}{001}$ (left); code $\mathcal{C}^2$ defined by $G_2 = (101)$ or $H_2 = \binom{010}{101}$ (right).

Let $\mathbf{u}_x G_1$ and $\mathbf{u}_y G_2$ define codewords in $\mathcal{C}^1$ and $\mathcal{C}^2$. By definition these vectors have a null syndrome. All possible codewords in a coset of syndrome $\mathbf{s}$ can be enumerated by translating a codeword belonging to the coset of null syndrome by a vector $\mathbf{t}$ belonging to the coset of syndrome $\mathbf{s}$ (see equation (1)). If the code is systematic, a candidate for the translating vector is $\mathbf{t} = (\mathbf{0}\ \mathbf{s})$, as explained in Section 2.2. Thus, $\mathbf{x}$ and $\mathbf{y}$ can be seen as translated versions of codewords belonging to the cosets of null syndrome of the codes $\mathcal{C}^1$ and $\mathcal{C}^2$ respectively. The vectors $\mathbf{x}$ and $\mathbf{y}$ can thus be expressed as $\mathbf{x} = \mathbf{u}_x G_1 \oplus (\mathbf{0}\ \mathbf{s}_x)$, $\mathbf{y} = \mathbf{u}_y G_2 \oplus (\mathbf{0}\ \mathbf{s}_y)$, where $\mathbf{t}_x = (\mathbf{0}\ \mathbf{s}_x)$ and $\mathbf{t}_y = (\mathbf{0}\ \mathbf{s}_y)$ are the representatives of the cosets of $\mathcal{C}^1$ and $\mathcal{C}^2$ with syndromes $\mathbf{s}_x$ and $\mathbf{s}_y$. The decoder must search for a pair of codewords, one from each translated coset of $\mathcal{C}^1$ and $\mathcal{C}^2$. When receiving the syndromes $\mathbf{s}_x$ and $\mathbf{s}_y$, it first finds a codeword $\mathbf{c}$ of the main code $\mathcal{C}$ (which is actually the error pattern between $\mathbf{x}$ and $\mathbf{y}$) that is closest to the vector $\mathbf{t} = \mathbf{t}_x \oplus \mathbf{t}_y = (\mathbf{0}\ \mathbf{s}_x) \oplus (\mathbf{0}\ \mathbf{s}_y)$. The error pattern $\mathbf{x} \oplus \mathbf{y}$ is the minimum weight codeword having the same syndrome as the vector $\mathbf{t}$. If the code is systematic, the sequences $\mathbf{x}$ and $\mathbf{y}$ are reconstructed as $\hat{\mathbf{x}} = \hat{\mathbf{u}}_x G_1 \oplus \mathbf{t}_x$ and $\hat{\mathbf{y}} = \hat{\mathbf{u}}_y G_2 \oplus \mathbf{t}_y$, where $\hat{\mathbf{u}}_x$ and $\hat{\mathbf{u}}_y$ are the systematic bits of the codeword $\mathbf{c}$. This method achieves the optimal SW compression sum-rate if the global code $\mathcal{C}$ achieves the capacity of the equivalent correlation channel (between $\mathbf{x}$ and $\mathbf{y}$). The number of lines assigned to $G_1$ (and therefore to $G_2$) allows to choose any rate on the dominant face of the SW region (between points A and B in Figure 1 (b)). The above ideas are further developed in [31], [32], for non-systematic linear codes. The code construction has been extended in [28] to the case where the sources $X$ and $Y$ are binary but non uniformly distributed.

In a second approach proposed in [10], the vectors $\mathbf{x}$ and $\mathbf{y}$ belong to the same coset of a unique code $\mathcal{C}$, which is the translated version of the coset of null syndrome by a vector

$\mathbf{t}$. Each source vector is identified by a syndrome computed with the parity check matrix $H$ of the code $\mathcal{C}$ plus part of the source information bits [10]. The syndromes $\mathbf{s}_x^T = H\mathbf{x}^T$ and $\mathbf{s}_y^T = H\mathbf{y}^T$, of length $(n-k)$, are thus computed for both sequences and transmitted to the decoder. In addition, the $k'$ first bits of $\mathbf{x}$ (denoted $\mathbf{x}_1^{k'}$) and the $k-k'$ next bits for the source $\mathbf{y}$ (denoted $\mathbf{y}_{k'+1}^{k}$), are transmitted as systematic bits, where $k'$ is an integer so that $k' \in [0, k]$. The total rate for the sequences $\mathbf{x}$ and $\mathbf{y}$ of length $n$ is respectively $n-k+k'$ and $n-k'$ bits. The structure of the coders is depicted in Figure 13. Note that $k'=k$ and $k'=0$ correspond to the two asymmetric set-ups with rates given by the corner points $A$ and $B$ of the SW region.

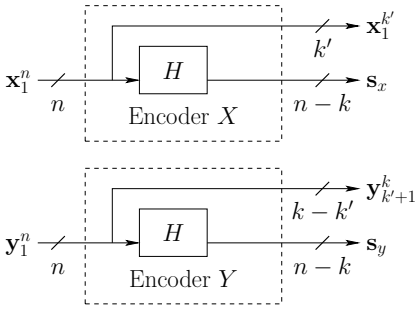



Figure 13: The non asymmetric coders in the syndrome approach [10]. A single code $\mathcal{C}$ (determined by the parity check matrix $H$) is used for each sequence to be compressed.

As in the asymmetric case, the representative of the coset is thus the concatenation of the transmitted information bits ($k'$ bits for the sequence $\mathbf{x}$) and of the $(n-k)$-length syndrome vector. As in the first approach, when receiving the syndromes $\mathbf{s}_x$ and $\mathbf{s}_y$, the decoder must find a codeword $\mathbf{c}$ of the code $\mathcal{C}$ that is closest to the translating vector $\mathbf{t} = \mathbf{t}_x \oplus \mathbf{t}_y = (\mathbf{0}\ \mathbf{s}_x) \oplus (\mathbf{0}\ \mathbf{s}_y)$. For this, it first computes $\mathbf{s}_z = \mathbf{s}_x \oplus \mathbf{s}_y$, which is the syndrome of the error pattern $\mathbf{z}$ between $\mathbf{x}$ and $\mathbf{y}$ ($\mathbf{z} = \mathbf{x} \oplus \mathbf{y}$). In [10], a modified LDPC decoder estimates $\hat{\mathbf{z}} = \widehat{\mathbf{x} \oplus \mathbf{y}}$ from the syndrome $\mathbf{s}_z$ and the all-zero word of size $n$ (see Figure 14) [16]. The error pattern $\mathbf{z}$ is the smallest weight codeword with syndrome $\mathbf{s}_z$. When convolutional and turbo codes are used, the estimation of the error pattern can be performed with a conventional Viterbi decoder and an inverse syndrome former (ISF) [34] or with a syndrome decoder [25] (see Section 2.2 for further details).

Once the error pattern is found, the subsequences of information bits $\mathbf{x}_{k'+1}^{k}$ and $\mathbf{y}_1^{k'}$ can be retrieved from the error pattern $\mathbf{z} = \mathbf{x} \oplus \mathbf{y}$ as (see Figure 14-right)

$$\hat{\mathbf{x}}_{k'+1}^{k} = \mathbf{y}_{k'+1}^{k} \oplus \hat{\mathbf{z}}_{k'+1}^{k} \text{ and } \hat{\mathbf{y}}_1^{k'} = \mathbf{x}_1^{k'} \oplus \hat{\mathbf{z}}_1^{k'}. \tag{3}$$

Subsequences of $n-k$ bits, i.e. $\mathbf{x}_{k+1}^{n}$ and $\mathbf{y}_{k+1}^{n}$, remain to be computed for both sequences. Let us assume that $H = (A\ B)$, where $B$ is an invertible square matrix of dimension $(n-k) \times (n-k)$. Note that for a rate $k/n$ channel code, the parity check matrix $H$ has rank $n-k$. Therefore, one can always find a permutation to be applied on the columns of $H$ so that the resulting parity check matrix has the right form. Thus, $\mathbf{s}_x = H\mathbf{x} = (A\ B)\mathbf{x} = A\ \mathbf{x}_1^k \oplus B\ \mathbf{x}_{k+1}^n$, and the remaining $n-k$ unknown bits of the

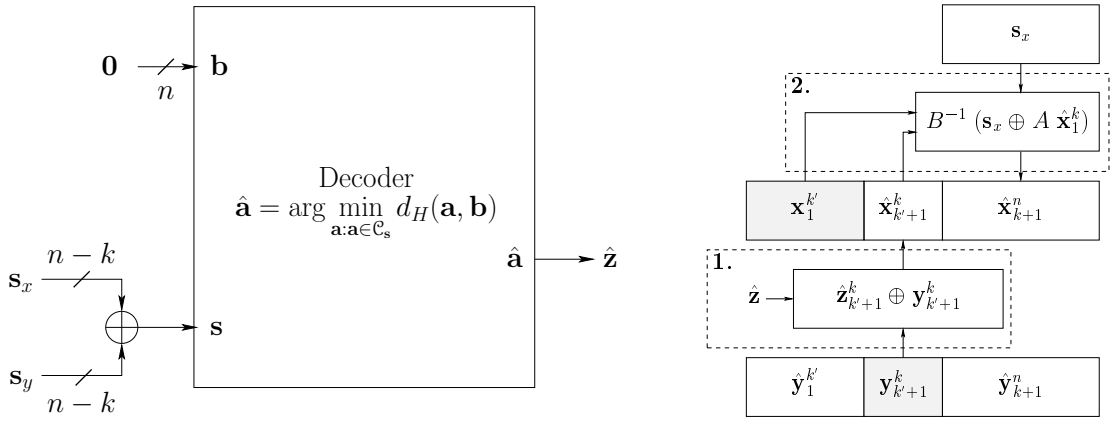



Figure 14: Non-asymmetric decoder: estimation of the error pattern $\mathbf{z} = \mathbf{x} \oplus \mathbf{y}$ (left); reconstruction of the source sequence $\mathbf{x}$ (right).

sequence $\mathbf{x}$ (and similarly for the sequence $\mathbf{y}$) can be computed as

$$\hat{\mathbf{x}}_{k+1}^n = B^{-1}\ (\mathbf{s}_x \oplus A\ \hat{\mathbf{x}}_1^k), \tag{4}$$

where $B^{-1}$ denotes the inverse of the matrix $B$ [10]. The authors in [10] have used LDPC codes.

If the error pattern is not perfectly estimated, the estimation of the remaining symbols $\mathbf{x}_{k+1}^n$ in equation (4) may yield error propagation. Note that the unknown positions in the vector $\mathbf{x}$ are design parameters. These positions (or equivalently the columns of the matrix $H$ to be extracted in order to build the matrix $B$) can be chosen such that the inverse $B^{-1}$ is as sparse as possible. Figure 15 shows the BER of the error pattern $\mathbf{z}$ (continuous line) and its effect on the estimation of the source sequence $\mathbf{x}$. Performance is shown for a convolutional code. The dotted curve represents the BER, when equation (4) is performed with an arbitrary invertible matrix $B$. This BER can be lowered to the dashed curve, if a matrix $B$ with sparse inverse is chosen. This limits the error propagation. Finally, the error propagation can be further reduced, if one applies a modified decoder to solve equation (4). This decoder (Viterbi for a convolutional code) is matched to a channel which combines a perfect channel (for $\mathbf{x}_1^{k'}$), a BSC (for $\hat{\mathbf{x}}_{k'+1}^k$ with crossover probability, the BER of $\mathbf{z}$) and an erasure channel (for $\mathbf{x}_{k+1}^n$). Interestingly, with this decoder the BER of $\mathbf{x}$ remains almost the same as that of $\mathbf{z}$.

## 3.4 Source splitting

Source splitting [22] is another approach to vary the rates between the two encoders. The approach is asymptotic and involves splitting the source into sub-sources having a lower entropy. More precisely, it transforms the two source problem into a three source problem where one source, e.g. $X$ is split into two i.i.d discrete sources $U$ and $V$ as $U = XT$ and $V = Y(1-T)$. $T$ is an i.i.d. binary source (which can be seen as a time-sharing sequence). By varying $\mathbb{P}(T = 1)$, one can vary $H(Y|U)$ between $H(Y)$ and $H(Y|X)$. The sum-rate

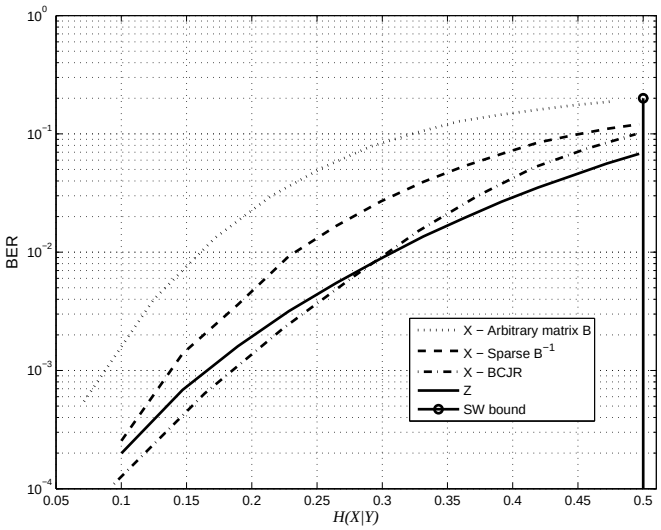



Figure 15: Error propagation in the estimation of the source $X$. The convolutional code is defined by its parity check matrix $H = \begin{pmatrix} 11 & 15 & 06 \\ 15 & 12 & 17 \end{pmatrix}$ and is punctured over a period of four trellis sections in order to get a $2:1$ compression rate for the source.

becomes $R_X + R_Y = R_U + R_V + R_Y = H(U|T) + H(Y|U) + H(V|U,Y)$. A random coding argument is used to show that the rate triple is achievable [22], however no constructive codes are proposed.

## 3.5 Rate adaptation

In practical applications, the correlation between the two sources may not be known at the time of code design. It may thus be necessary, as in the asymmetric set-up, to have solutions allowing for flexible rate adaptation to any correlation between the two sources. Non-asymmetric rate-adaptive SW coding/decoding schemes are direct extensions of their asymmetric counterparts. Rate adaptation is performed by simply puncturing information, parity or syndrome bits in the schemes described above.

In the parity approach, we have seen that for two correlated sequences of length $n$, the rate allocation between the two sequences is controlled by the number $l$ of information bits transmitted for each. The change in the rate allocation between the two sources is thus performed by puncturing more or less information bits. For a given rate share between the two sources, rate adaptation to varying correlation is achieved by simply puncturing the parity bits. Standard decoding of punctured channel codes can then be applied.

For the syndrome approaches, as explained in Section 2.3, the application of the puncturing is less straightforward. Puncturing the syndrome bits may degrade the performance of the code. The approach considered in [35], similarly to [37] for the asymmetric set-up, consists of first protecting the syndromes with an accumulator code. The effect of the accumulator code followed by the puncturing is equivalent to merging some rows of the parity check matrix $H$ by adding them, thus constructing a matrix $H_i$ of dimension

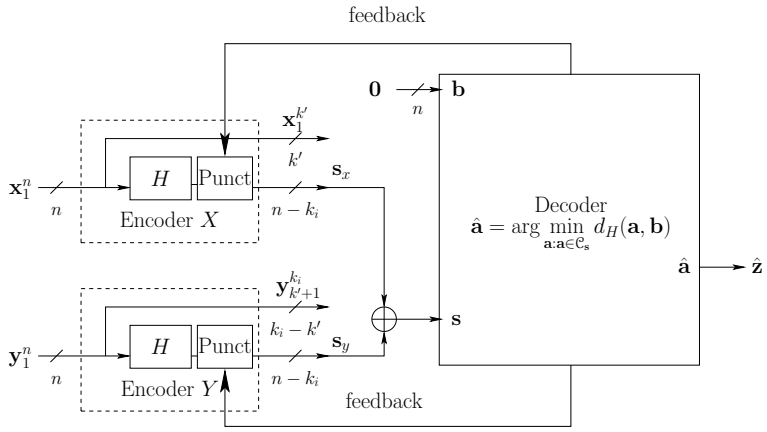



Figure 16: Rate-adaptive coding/decoding scheme.

$(n-k_i)\times n$, of rank $n-k_i$. In the matrix $H_i$, with appropriate permutation, one may exhibit a submatrix $B_i$ of dimension $(n-k_i)\times(n-k_i)$ which is invertible (see equation (4)). The positions of the $k'$ ($k'\in[0,k_i]$) and $k_i-k'$ information bits transmitted for the sequences $\mathbf{x}$ and $\mathbf{y}$ respectively must then be chosen so that they correspond to the positions of the remaining non-free $k_i$ columns of $H_i$. If the correlation is not known to the encoder, the rate may be controlled via a feedback channel. In this case, once the error pattern $\hat{\mathbf{z}}$ has been estimated, the decoder verifies if the condition $\hat{\mathbf{z}}H^T=\mathbf{s}_z$ is satisfied. If this condition is not satisfied, it then requests for more syndrome bits. Note that, if the error pattern is not perfectly estimated, the last step of the decoding algorithm represented by equation (4) may yield error propagation. The coding and decoding structures are depicted in Figure 16. Figure 17 illustrates the rate points achieved with a non-symmetric SW coding/decoding scheme based on LDPC codes.

# 4 Advanced topics

## 4.1 Practical code design based on source codes

**Code design based on source alphabet partitioning:** The use of source codes for SW coding has first been explored in [2] considering Huffman codes. The approach relies on partitioning the source alphabet into groups of symbols. One source $Y$ is coded with a Huffman code designed according to its stationary probability $\mathbb{P}(Y)$. The alphabet of the second source $X$ is partitioned so that two symbols $x_i$ and $x_j$ are not grouped together if there exists a symbol $y$ such that $\mathbb{P}(x_i,y)>0$ and $\mathbb{P}(x_j,y)>0$. The joint p.m.f of $(X,Y)$ may need to be slightly modified, e.g. by thresholding the smallest values and then renormalizing, so that this condition is verified. In addition, the entropy of the group index must be minimum, and approach at best $H(X|Y)$. Knowing the SI symbol $y$, and the group index, the decoder can find the transmitted symbol $x$. The above code

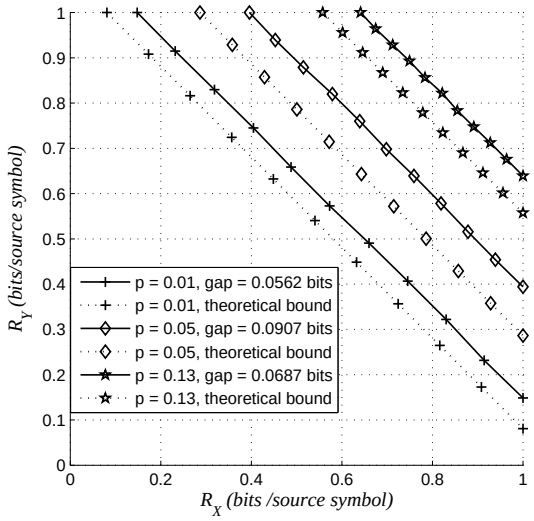



Figure 17: Rate points achieved for different parameters $p$ of the binary symmetric correlation channel.

design based on alphabet partitioning has been formalized and generalized in [42] for arbitrary p.m.f. $\mathbb{P}(X, Y)$, assuming memoryless sources. The partition can be coded with a Huffman or an arithmetic code.

**Punctured and overlapped arithmetic codes:** Different solutions based on overlapped arithmetic [11] and quasi-arithmetic (QA) [3] codes, as well as punctured (quasi)-arithmetic codes [18] have appeared recently for the case of binary sources. Arithmetic coding recursively divides the interval $[0, 1)$ according to the source probabilities. Let us consider a binary memoryless source with probabilities denoted $P_0$ and $P_1$. At each symbol clock instant $n$, the current interval $[L_n, H_n)$ is divided into two subintervals whose widths are proportional to $P_0$ and $P_1 = 1 - P_0$ respectively. One of these subintervals is selected according to the value of the next symbol and becomes the current interval. Once the last symbol is encoded, the encoder produces a sequence of bits identifying the final sub-interval. Practical implementations of arithmetic coding have been first introduced in [23] and [19], and further developed in [24], [38].

The encoding and decoding processes can be modeled as state machines. But, with optimal arithmetic coding, the number of states can grow exponentially with the sequence length. Controlled approximations can reduce the number of possible states without significantly degrading compression performances [13]. This fast, but reduced precision, implementation of arithmetic coding is called *quasi-arithmetic coding* [13]. Instead of using the real interval $[0, 1[$, quasi-arithmetic coding is performed on an integer interval $[0, T[$. The value of $T$ controls the trade-off between complexity, the number of states, and compression efficiency.

The average code length achieved per input source symbol is very close to the source entropy. To further reduce the rate down to the conditional entropy of two correlated

sources, two strategies can be considered: puncturing the sequence of encoded bits [18] or overlapping the symbol probability intervals of the arithmetic [11] or quasi-arithmetic coder [3]. For the overlapped (quasi)-arithmetic codes, at a given instant $n$, the probability interval $[L_n, H_n)$ is partitioned into two sub-intervals $[L_n, P_0(H_n - L_n) + \rho T/2)$ associated to the symbol 0 and $[P_0(H_n - L_n) - \rho T/2, H_n)$ associated to the symbol 1. The parameter $\rho$ controls the overlap between the sub-intervals. The larger the overlap, the lower the bit rate but the higher the uncertainty on the decoded sequence. In both cases, overlapped and punctured codes, the resulting code is not uniquely decodable anymore. The ambiguity on the encoded sequence of symbols can be removed with the help of the SI (correlated source) and of a symbol MAP decoder.

To explain the decoding algorithm, let us first consider the punctured solution. Let $\mathbf{x}$ and $\mathbf{y}$ be two correlated sequences of length $L(\mathbf{x})$. The sequence $\mathbf{x}$ is coded with a QA code producing a sequence of bits $\mathbf{u}$ of length $L(\mathbf{u})$ which is then punctured. Given the received QA coded bitstream $u_1^{L(\mathbf{u})}$ and the correlated sequence $y_1^{L(\mathbf{x})}$ (SI), the symbol posterior marginals $\mathbb{P}(X_n = x_n \,|\, u_1^{L(\mathbf{u})}, y_1^{L(\mathbf{x})})$ are computed by running a BCJR algorithm [4] on the state model of the QA automaton. To cope with the fact that the QA automaton contains transitions corresponding to a varying number of symbols and a varying number of bits, the state model used in the decoding must keep track of the number of symbols being coded at a particular bit clock instant $k$. It is hence defined by the pairs of random variables $\mathcal{V}_k = (N_k, M_k)$, where $N_k$ denotes the state of the QA automaton, and $M_k$ represents the symbol clock value at the bit clock instant $k$ [12]. Since the variable $M_k$ corresponds to the symbol clock, it accounts for the correlation between the encoded sequence and the SI. The transition probabilities on the state model $\mathcal{V}_k = (N_k, M_k)$ are equal to the transition probabilities between states $N_k$ of the QA automaton, if the numbers of bits and symbols associated to this transition match the one defined by the automaton. The transition probabilities are equal to zero otherwise. The probabilities of transition between states $N_k$ of the QA automaton depend on the source statistics, and can account for source memory. Thus, for each state $\nu = (n, m)$, the BCJR algorithm computes the probabilities $\alpha_k(\nu) = \mathbb{P}(\mathcal{V}_k = \nu; \mathbf{U}_1^k)$ and $\beta_k(\nu) = \mathbb{P}(\mathbf{U}_{k+1}^{L(\mathbf{U})} \,|\, \mathcal{V}_k = \nu)$. The branch metric $\gamma_k(\nu'|\nu)$ used for the transition $\tau$ between two states $\nu = (n, m)$ and $\nu' = (n', m')$ is given by $\gamma_k(\nu'|\nu) = \mathbb{P}(\nu'|\nu) \times \mathbb{P}(\mathbf{b}_\tau) \times \mathbb{P}(\mathbf{x}_\tau \,|\mathbf{y}_\tau)$, where $\mathbf{b}_\tau$ and $\mathbf{x}_\tau$ denote the subsequences of bits and symbols associated to the given transition $\tau$ on the QA automaton. The probability $\mathbb{P}(\mathbf{b}_\tau)$ is computed considering a probability of 0.5 for the punctured positions. The term $\mathbb{P}(\mathbf{x}_\tau \,|\mathbf{y}_\tau)$ represents the correlation channel between $X$ and $Y$.

The decoding for the overlapped quasi-arithmetic codes proceeds in the same manner as for the punctured solution. The only difference is in the automaton which, in the case of overlapped codes, does not only depend of the distribution of the source $X$ but also of the overlap factor $\rho$ (for details see [18] and [3]). A second difference is in the branch metric $\gamma_k(\nu'|\nu)$ in which the term $\mathbb{P}(\mathbf{b}_\tau)$ disappears, since, in this case, all the coded bits are known to the decoder (and not punctured as in the previous method).

## 4.2 Generalization to non-binary sources

In the chapter so far, only binary correlated sources have been considered. Practical applications often involve non-binary, e.g. $q$-ary, sources. In practice, before being encoded, the $q$-ary sources are first binarized. Each resulting sequence of bits is then fed to the SW coder. This is, for example, the case in [43] where the authors describe the principle using turbo codes to compress both sources $X$ and $Y$. For each source, the sequences of bits at the output of the "symbol-to-bit" converter are interleaved and input to a punctured turbo coder. The turbo decoders act at a bit level in a standard fashion, but information exchange is performed between the two turbo decoders at the symbol level. The *a posteriori* probabilities computed by a turbo decoder for each input bit is converted into a measure on the corresponding $q$-ary symbol. The resulting symbol probabilities are then multiplied by the conditional probabilities characterizing the correlation channel and fed into the second turbo decoder as extrinsic information.

## 4.3 Generalization to $M$ sources

The asymmetric and non-asymmetric coding schemes presented above can be extended to more than 2 sources. The practical code design approach proposed in [20] has been extended in [31] and [32] for $M$ sources (see Section 3.3). Each sub-code is constructed by selecting a subset of rows from the generator matrix $G$ of the starting code $\mathcal{C}$. Examples of design are given for IRA codes. However, the method is shown to be suboptimal when the number of sources is greater (or equal) than three. The parity approach described in [26] has also been extended to $M$ sources in [27]. The sequences $\mathbf{x}_i, i = 1, \ldots M$, of length $n$ are fed into $M$ LDPC coders which produce subsequences of $a_i \times k$ information bits and of $(1 - a_i) \times k$ parity bits. Joint decoding of the $M$ sources is then performed.

# 5 Conclusions

This chapter has explained how near-capacity achieving channel codes can be used to approach the SW rate bound. Although the problem is posed as a communication problem, classical channel decoders need to be modified. This chapter has outlined the different manners to adapt channel decoding algorithms to the Slepian-Wolf decoding problem. In the approaches presented, it is assumed that syndrome or parity bits are transmitted over a perfect channel. As explained above, the equivalent communication channel is thus the parallel combination of a BSC and a perfect channel. In some applications, this may not be the case: syndrome or parity bits may have to be transmitted on an erroneous communication channel. Distributed joint source-channel coding schemes, in which the SW code acts as a single code for both source coding and channel error correction, can be found in the literature to address this particular scenario. This last problem however remains a field of research. Finally, the use of SW codes is currently being explored for a number of applications such as low-power sensor networks, compression of hyperspectral images, mono-view and multi-view video compression, error-resilient video transmission,

and secure fingerprint biometrics. These applications are presented in other chapters of this book.

# References


[1] A. Aaron and B. Girod. Compression with side information using turbo codes. *Proceedings of the IEEE International Data Compression Conference (DCC)*, 0(0):252-261, Apr 2002.

[2] A.K. Al Jabri and S. Al-Issa. Zero-error codes for correlated information sources. *Proceedings of Cryptography*, 0(0):17-22, Dec. 1997.

[3] X. Artigas, S. Malinowski, C. Guillemot, L. Torres. Overlapped quasi-arithmetic codes for distributed video coding. *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pp. II.9-II.12, Sept. 2007

[4] 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*, IT-20, pp. 284-287, Mar. 1974.

[5] J. Bajcsy and P. Mitran. Coding for the Slepian-Wolf problem with turbo codes. *Proceedings of the IEEE International Global Communications Conference (Globecom)*, pp. 1400-1404, Dec. 2001.

[6] F. Cabarcas and J. Garcia-Frias. Approaching the Slepian-Wolf boundary using practical channel codes. *Proceedings IEEE International Symposium on Information Theory*, pp. 330, June 2004.

[7] J. Chen, A. Khisti, D.M. Malioutov and J.S. Yedidia. Distributed source coding using serially-concatenated-accumulate codes, *roceedings of the IEEE International Information Theory Workshop*, pp.209-214, Oct. 2004.

[8] J. Garcia-Frias and F. Cabarcas. Approaching the Slepian-Wolf boundary using practical channel codes. *Signal Processing*, 86(11):3096-3101, 2006.

[9] J. Garcia-Frias and Y. Zhao. Compression of correlated binary sources using turbo codes. *IEEE Communications Letters*, 5(0):417-419, Oct. 2001.

[10] N. Gehrig and P.L. Dragotti. Symmetric and asymmetric slepian–wolf codes with systematic and non–systematic linear codes. *IEEE Communications Letters*, 9(1):61-63, Jan. 2005.

[11] M. Grangetto, E. Magli, and G. Olmo. Distributed arithmetic coding. *IEEE Communications letters*, 11(11):883-885, Nov. 2007.

[12] T. Guionnet and C. Guillemot. Soft and joint source-channel decoding of quasi-arithmetic codes. *EURASIP Journal on applied signal processing*, 3(0):394-411, Mar. 2004.

[13] P. G. Howard and J. S. Vitter. *Image and Text Compression*. Kluwer Academic Publisher, pp. 85-112, 1992.

[14] J. Jiang, D. He and A. Jagmohan, Rateless Slepian-Wolf coding based on rate adaptive Low-Density-Parity-Check codes. *Proceedings of the IEEE International Symposium on Information Theory*, pp. 1316-1320, June 2007.

[15] J. Li and H. Alqamzi. An optimal distributed and adaptive source coding strategy using rate–compatible punctured convolutional codes. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP)*, vol.3, pp. iii/685-iii/688, pp. 18-23 Mar. 2005.

[16] 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*, 6(10):440-442, Oct. 2002.

[17] 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*, 6(0):440–42, Oct. 2002.

[18] S. Malinowski, X. Artigas, C. Guillemot and L. Torres. Distributed source coding using punctured quasi-arithmetic codes for memory and memoryless sources. *submitted to IEEE International Workshop on Signal Processing Systems*, Oct. 2008.

[19] R. Pasco. Source coding algorithms for fast data compression. PhD thesis, Dept. of Electrical Engineering, Stanford Univ., Stanford Calif., 1976.

[20] S. S. Pradhan and K. Ramchandran. Distributed source coding: symmetric rates and applications to sensor networks. *Proccedings of the IEEE International Data Compression Conference (DCC)*, 0(0):363-372, Mar. 2000.

[21] S. S. Pradhan and K. Ramchandran. Distributed source coding using syndromes (DISCUS): Design and construction. *Proceedings of the IEEE International Data Compression Conference (DCC)*, 0(0):158-167, Mar. 1999.

[22] B. Rimoldi and R. Urbanke. Asynchronous Slepian-Wolf Coding via Source-Splitting. *Proceedings of the IEEE International Symposium on Information Theory*, pp. 271, July 1997.

[23] J. J. Rissanen. Generalized Kraft inequality and arithmetic coding. *IBM J. Res. Develop.*, vol. 20, pp. 198-203, May 1976.

[24] ——, Arithmetic codings as number representations, *Acta Polytech. Scand. Math.*, vol. 31, pp. 44-51, Dec. 1979.

[25] A. Roumy, K. Lajnef, and C. Guillemot. Rate-adaptive turbo-syndrome scheme for Slepian-Wolf coding. *Proceedings of the IEEE Asilomar Conference on Signals, Systems, and Computers*, vol. 0, Nov. 2007.

[26] M. Sartipi and F. Fekri. Distributed source coding in wireless sensor networks using LDPC coding: The entire Slepian-Wolf rate region. *Proceedings of the IEEE Wireless Communications and Networking Conference*, 4(0):1939-1944, Mar. 2005.

[27] M. Sartipi and F. Fekri Distributed Source Coding Using Short to Moderate Length Rate-Compatible LDPC Codes: The Entire Slepian-Wolf Rate Region *IEEE Transactions on Communications*, Vol. 56, No. 3, pp. 400-411, Mar. 2008.

[28] D. Schonberg, K. Ramchandran and S.S. Pradhan. Distributed code constructions for the entire Slepian-Wolf rate region for arbitrarily correlated sources *Proceedings of the IEEE International Data Compression Conference (DCC)*, 0(0):292-301, Mar. 2004.

[29] V. Sidorenko and V. Zyablov. Decoding of convolutional codes using a syndrome trellis. *IEEE Transactions on Information Theory*, 0(0):1663-1666, Sept. 1994.

[30] D. Slepian and J.K. Wolf. Noiseless Coding of Correlated Information Sources. *IEEE Transactions on Information Theory*, vol. IT-19, pp. 471-480, July 1973.

[31] V. Stankovic, A. D. Liveris, Z. Xiong, and C. N. Georghiades. Design of Slepian-Wolf codes by channel code partitioning. *Proceedings of the IEEE International Data Compression Conference, (DCC)*, 0(0):302-311, 2004.

[32] V. Stankovic, A. D. Liveris, Z. Xiong, and C. N. Georghiades. On code design for the Slepian-Wolf problem and lossless multiterminal networks. *IEEE Transactions on Information Theory*, 52(4):1495-1507, Apr. 2006.

[33] P. Tan and J. Li. Enhancing the robustness of distributed compression using ideas from channel coding. *Proceedings of the IEEE International Global Telecommunications Conference, (GLOBECOM)*, vol. 4, pp. 5, Dec. 2005.

[34] P. Tan and J. Li. A practical and optimal symmetric Slepian-Wolf compression strategy using syndrome formers and inverse syndrome formers. *Proceeding of 43rd Annual Allerton Conference on Communication, Control and Computing*, Sept. 2005.

[35] V. Toto-Zarasoa, A. Roumy and C. Guillemot. Rate-adaptive codes for the entire Slepian-Wolf region and arbitrarily correlated sources. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2008.

[36] Z. Tu, J. Li, and R. S. Blum. An efficient SF-ISF approach for the Slepian-Wolf source coding problem. *Eurasip Journal on Applied Signal Processing*, 6(0):961-971, May 2005.

[37] D. Varodayan, A. Aaron, and B. Girod. Rate-adaptive codes for distributed source coding. *EURASIP Signal Processing*, 86(11):3123–3130, Nov. 2006.

[38] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, June 1987.

[39] J. K. Wolf. Efficient Maximum Likelihood decoding of linear block codes using a trellis. *IEEE Transactions on Information Theory*, 24(0):76-80, Jan. 1978.

[40] A. Wyner. Recent results in the Shannon theory. *IEEE Transactions on Information Theory*, 20(0):2-10, 1974.

[41] R. Zamir, S. Shamai, and U. Erez. Nested linear/lattice codes for structured multiterminal binning. *IEEE Transactions on Information Theory*, 48:1250-1276, 2002.

[42] Q. Zhao and M. Effros, Optimal code design for lossless and near lossless source Coding in multiple access network. *Proceedings of the IEEE International Data Compression Conference (DCC)*, 0(0):263-272, Mar. 2001.

[43] Y. Zhao. and J. Garcia-Frias, Data compression of correlated non-binary sources using punctured turbo codes. *Proceedings of the IEEE International Data Compression Conference (DCC)*, 0(0):242-251, Apr. 2002.