Density Evolution for the Simultaneous Decoding of LDPC-based Slepian-Wolf Source Codes

Andrew W. Eckford and Wei Yu
The E. S. Rogers Sr. Dept. of Electrical and Computer Engineering, University of Toronto
10 King's College Road, Toronto, Ontario, Canada M5S 3G4
E-mail: andrew.eckford@utoronto.ca, weiyu@comm.utoronto.ca

***Abstract*— This paper deals with the design and analysis of low-density parity-check (LDPC) codes for the Slepian-Wolf problem. The main contribution is a code design method based on a density evolution (DE) analysis for the cases where multiple LDPC codes are simultaneously decoded at the decoder. Good source codes are designed both for memoryless sources and sources with Markov memory. Further, simultaneous decoding is generalized to the case of source splitting, which allows non-corner points of the Slepian-Wolf region to be achieved even for sources with equiprobable marginal distributions.**

I. Introduction

Interest in the elegant Slepian-Wolf theorem for coding of correlated sources recently was invigorated by a practical scheme to achieve the theorem's performance bound. Inspired by some early work by Wyner [1], the method of *distributed source coding using syndromes* (DISCUS) [2] adapted channel codes to give a practical method to perform both encoding and decoding for two (or more) sources, so long as the encoders were restricted to particular points on the boundary of allowed rates (known as *corner points*). As a result, a large and growing body of research is adapting Turbo codes and LDPC codes to approach the Slepian-Wolf bound, much as they have been used to approach the Shannon bound for channel coding.

It is well known that corner points are achievable with LDPC codes, whether the source is memoryless [3] or has Markov memory [4], [5]. Density evolution (DE) [6], a technique for determining asymptotic performance of LDPC codes, has been applied to both these cases. The question of achieving non-corner points has been addressed in two ways: the sources x and y can be split up into three virtual sources $x = (x^{(1)}, x^{(2)})$ and y, to form virtual corner points [7], [8], or one can encode both sources x and y as LDPC codes and simultaneously decode the two encoded sources, so the encoders can offset each others' rates [9], [10].

Both approaches have certain drawbacks. In source splitting, the conventional approach is to decode the source codes serially, obtaining $x^{(1)}$, then y, then $x^{(2)}$. Although this approach can theoretically achieve the Slepian-Wolf bound, it potentially sacrifices flexibility, performance, and code length. On the other hand, the simultaneous decoding in [9], [10] has limited applicability. It is only possible to decode both codes for certain skewed source distributions, where the marginal source distributions are far from equiprobable.

We argue that a better approach is to do both rate splitting and simultaneous decoding at once. That is, we can split the sources into $x = (x^{(1)}, x^{(2)})$ and y, and simultaneously decode $x^{(2)}$, and y (since, at a virtual corner point, $x^{(1)}$ is encoded independently). In this paper, we combine source splitting and simultaneous decoding and use this combined framework to achieve non-corner points in the Slepian-Wolf rate region, even for sources with an equiprobable marginal distribution. Furthermore, this method applies to sources both with and without memory.

This paper makes the following contributions:

- *Simultaneous decoding for sources with memory.* Previous work on coding for sources with memory (such as in [4], [5]) has focused on corner points on the Slepian-Wolf bound. Simultaneous decoding is used to avoid this restriction.
- *Density evolution analysis of simultaneous decoding.* Density evolution (DE) [6] has previously been used in LDPC source code design (such as in [3], [5], [9], [11]), but it has not been adapted to the case of simultaneously decoding the LDPC codes.
- *Design of LDPC codes using density evolution.* We use the density evolution algorithm we derived to design good source codes for sources with and without memory.

The remainder of this paper is organized as follows. In Section II, we describe the system and discuss simultaneous decoding of the LDPC codes. In Section III, we show that source splitting can be included as a generalization of our simultaneous decoding framework. In Section IV, we discuss the extension of our framework to cases where the sources have memory. In Section V, we present good degree sequences for LDPC Slepian-Wolf codes that were found using our techniques.

II. Simultaneous Decoding

In this section, we discuss the simultaneous decoder first outlined in [9] and [10], and show how to analyze it using DE. The following definition will be useful in this section: the function $\sigma : \{0, 1\} \rightarrow \{+1, -1\}$ converts between two different types of binary alphabets, where $\sigma(0) = +1$ and $\sigma(1) = -1$.

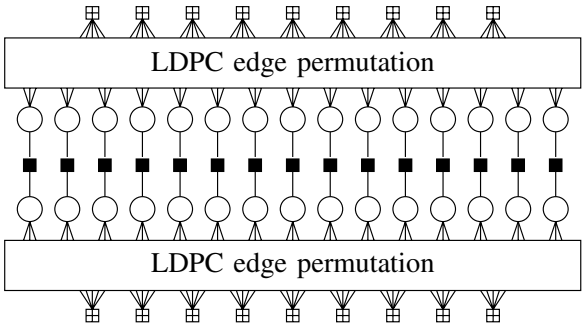

Fig. 1. A depiction of a simultaneous LDPC source decoder.

A. Model

Let $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$ represent binary sources observed at different Slepian-Wolf encoders (expressed as row vectors), where x_i and x_j (resp., y_i and y_j) are independent for all $i \neq j$. The joint PMF $p_{X_i,Y_i}(x_i, y_i)$ is the same for all i, and parameterizes the source coding problem by establishing the statistical relationship between x_i and y_i.

Encoding is accomplished using LDPC codes, as follows. An LDPC code is associated with both x and y, and in general the two codes are different. Let $\mathbf{H}^{(x)}$ and $\mathbf{H}^{(y)}$ represent the parity check matrices of the LDPC code associated with x and y, respectively. For each source vector, we form the *syndromes* (i.e., the vectors of parity bits) $s^{(x)} = x\mathbf{H}^{(x)}$ and $s^{(y)} = y\mathbf{H}^{(y)}$. Each encoder transmits only its syndrome, so the encoded sequence for x and y are $s^{(x)}$ and $s^{(y)}$, respectively.

As usual for an LDPC code, the decoder may be represented using a factor graph, and decoded using the sum-product algorithm (SPA) [12], with a couple of caveats:

- The syndromes $s^{(x)}$ and $s^{(y)}$ represent the parity checks at which even parity is replaced with odd parity. If the message m is calculated by the usual SPA at the output of a parity check node, and the syndrome bit s is associated with that parity check, the correct value of the message becomes $m\sigma(s)$.
- The prior probabilities on the source letters (or intrinsic probabilities) are calculated by $\sum_{x_i} p_{X_i,Y_i}(x_i, y_i)$ for y_i, and $\sum_{y_i} p_{X_i,Y_i}(x_i, y_i)$ for x_i. At subsequent iterations, extrinsic information messages $p^{(E)}_{X_i}(x_i)$ and $p^{(E)}_{Y_i}(y_i)$ are obtained from the decoding of the LDPC code, and the prior probabilities then become $\sum_x p_{X_i,Y_i}(x_i, y_i) p^{(E)}_{X_i}(x_i)$ and $\sum_y p_{X_i,Y_i}(x_i, y_i) p^{(E)}_{Y_i}(y_i)$ for y_i and x_i, respectively. (Messages are passed as log-likelihood ratios.)

This setup is depicted in Fig. 1.

By *simultaneous decoding*, we mean that the two LDPC codes are decoded at the same time, and that the two decoders exchange SPA messages to help each other decode. (The alternative would be to wait until one LDPC code completes its decoding before passing messages to the other, which we call *serial decoding*.) As a message-passing schedule, we propose the parallel update schedule, in which updates occur at each variable node simultaneously, and then at each factor node simultaneously, regardless of type.

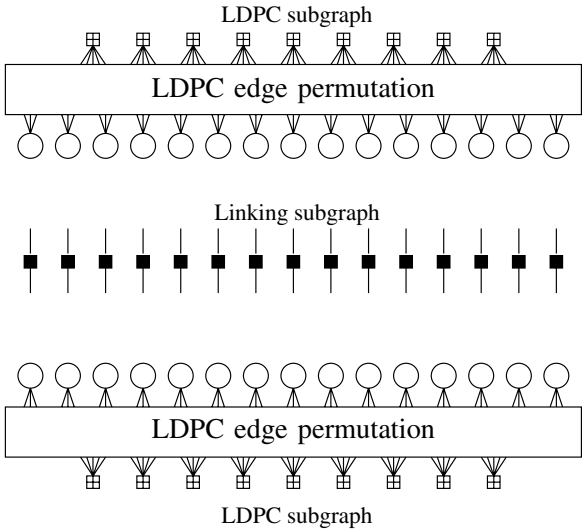

Fig. 2. The three decoding subgraphs in a simultaneous LDPC source decoder.

B. Message calculation and DE

The decoder may be broken down into three subgraphs: the LDPC decoder for the symbols x, the LDPC decoder for the symbols y, and the linking channel nodes linking the two decoders, as in Fig. 2. The SPA is well understood for the two decoder subgraphs. For the linking nodes, the message passed from source symbol x_i to source symbol y_i, written $c^{(y)}_i$, is a function of the extrinsic message for x_i, designated $m^{(x)}_i$. Let $\psi : \{0,1\} \times \mathcal{R} \to [0,1]$ represent the function $\psi(a,b) = \frac{1}{2} + \frac{1}{2}\sigma(a)\tanh\left(\frac{b}{2}\right)$, and note that $p^{(E)}_{X_i}(x_i) = \psi(x_i, m^{(x)}_i)$. Then we have

$$c^{(y)}_i = \log \frac{\sum_{x_i} p_{X_i,Y_i}(x_i, y_i = 0)\psi(x_i, m^{(x)}_i)}{\sum_{x_i} p_{X_i,Y_i}(x_i, y_i = 1)\psi(x_i, m^{(x)}_i)}, \quad (1)$$

and similarly for the message from y to x.

When operating at a corner point of the Slepian-Wolf rate region, there is a clear equivalence between a source and a binary noise sequence in a binary symmetric channel. A similar equivalence exists for joint decoding. Define a joint binary symmetric channel (JBSC) as a channel with two binary inputs $[u_{1,i}, u_{2,i}]$, two binary outputs $[v_{1,i}, v_{2,i}]$, and binary noise $[z_{1,i}, z_{2,i}]$, so that $v_{1,i} = u_{1,i} \oplus z_{1,i}$, and $v_{2,i} = u_{2,i} \oplus z_{2,i}$. Let $\Pr_{Z_{1,i},Z_{2,i}}(z_{1,i}, z_{2,i})$ represent the noise process PMF for this JBSC. If the channel inputs are LDPC codewords from the codes in Fig. 1, and $\Pr_{X_i,Y_i}(z_{1,i}, z_{2,i}) = \Pr(z_{1,i}, z_{2,i})$ for every setting of $[z_{1,i}, z_{2,i}]$, then it is straightforward to show that the two decoders have the same probability of error. In particular, for a joint binary source (x, y), we say that the source decoder is *equivalent to a channel decoder for a joint binary symmetric channel where (x, y) is the pair of noise sequences.*

As a consequence of this equivalence, we can perform DE with respect to a JBSC rather than the more complicated case of an LDPC coset code, and the decoder structure remains the same as in Fig. 2. For the joint nodes, the message passed from a symbol $u_{1,i}$ to a symbol $u_{2,i}$, written $c_i^{(u_2)}$, is a function of the channel outputs $[v_{1,i}, v_{2,i}]$, and the extrinsic message for $u_{1,i}$, designated $m_i^{(u_1)}$. Thus, this message is given by

$$c_i^{(u_2)} = \log \frac{\sum_{u_{1,i}} p_{V_{1,i},V_{2,i}}(v_{1,i},v_{2,i}|u_{1,i},u_{2,i}=0)\psi(u_{1,i},m_i^{(u_1)})}{\sum_{u_{1,i}} p_{V_{1,i},V_{2,i}}(v_{1,i},v_{2,i}|u_{1,i},u_{2,i}=1)\psi(u_{1,i},m_i^{(u_1)})}. \quad (2)$$

DE in each decoder subgraph is well understood. For the nodes in the linking subgraph, which calculate the function in (2), given an input density for $m_i^{(u_1)}$ and $[v_{i,1}, v_{i,2}]$, many methods exist for obtaining the density of $c_i^{(u_2)}$. As one simple example, the input densities could be sampled, and each sample associated with a probability. That probability is then associated with the probability of the corresponding output $c_i^{(u_2)}$. Furthermore, it is straightforward to show that the cycle-free assumption is satisfied in a graph such as in Fig. 1.

C. A remark on SPA decoding

Suppose we had a source distribution given by

$$p_{X,Y}(x,y) = \begin{cases} 0.49, & x = y = 0, x = y = 1; \\ 0.01, & x \neq y. \end{cases} \quad (3)$$

It is easy to calculate that the conditional entropies for this source are $H(X|Y) = H(Y|X) = 0.141$, which is low enough to make it an excellent candidate for Slepian-Wolf encoding. However, the marginal distributions of x and y are equiprobable, so from the perspective of the decoder in Fig. 1, the system starts off with no prior information at all about x or y – in fact, their prior information behaves exactly like erasures.

Under SPA decoding, for a parity check node with degree greater than one, if any input message is an erasure, the output message will always be an erasure. Thus, when all the input prior information is initially "erased," such as in a case like (3), the SPA can never begin decoding, because the output messages at the parity check nodes will always be erasures. To initiate the SPA process, there must be some way of receiving prior information about at least some fraction of the source symbols. This suggests the use of source splitting, and in the next section we introduce a generalization of the simultaneous decoder that we call *source splitting with simultaneous decoding*.

III. Source splitting with simultaneous decoding

Source splitting (also called rate splitting) is a technique used in Slepian-Wolf coding [7]. In the case with two sources (as before, designated x and y), the source x is broken into two sub-sources: $x^{(1)}$ and $x^{(2)}$. This is done using a random sequence $t \in \{0,1\}^n$, which is available to both the encoder and decoder (for example, using a pseudo-random sequence with the same seed), where $\phi := \Pr(t_i = 1)$ is the source splitting parameter. The source alphabets for $x^{(1)}$ and $x^{(2)}$ are ternary, so that $x^{(1)} \in \{0,1,E\}^n$ and $x^{(2)} \in \{0,1,E\}^n$. If $t_i = 0$, then $x_i^{(1)} = x_i$ and $x_i^{(2)} = E$; otherwise, if $t_i = 1$, then $x_i^{(1)} = E$ and $x_i^{(2)} = x_i$.

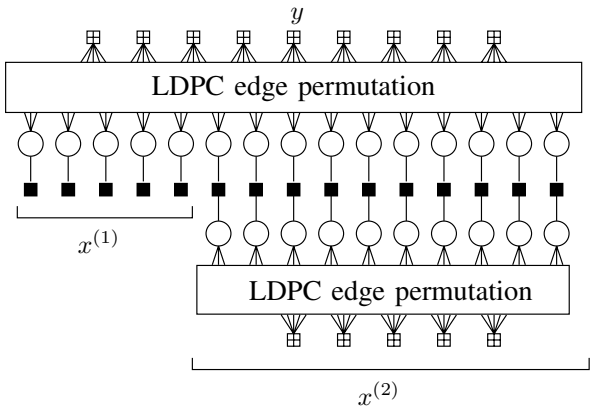

Fig. 3. A depiction of source splitting with simultaneous decoding.

The key observation is that these three pseudo-sources, $x^{(1)}$, $x^{(2)}$, and y, can be encoded as a vertex point in their Slepian-Wolf rate region. We firstly encode $x^{(1)}$ as an independent source, then we encode y assuming knowledge of $x^{(1)}$, and finally we encode $x^{(2)}$ assuming knowledge of $x^{(1)}$ and y.

Notice that the shared random vector t tells the decoder exactly when the letter E occurs in both $x^{(1)}$ and $x^{(2)}$, so only the $\{0,1\}$ portions need to be encoded. Since the sources are assumed memoryless, and letting w_t represent the Hamming weight of t, without loss of generality we can re-arrange the elements of $x^{(1)}$ so that the first w_t are the $\{0,1\}$ elements. Similarly, we can re-arrange $x^{(2)}$ so that the first $n - w_t$ are the $\{0,1\}$ elements. The E elements of $x^{(1)}$ and $x^{(2)}$ need not be encoded, and are thus ignored.

A. A simultaneous decoding view of source splitting

We can encode $x^{(1)}$ in an arbitrary manner (so long as its rate is at the source entropy), but we wish to encode $x^{(2)}$ and y using LDPC codes. From the perspective of y, all the values of $x^{(1)}$ are perfectly known, but the values of $x^{(2)}$ are provided via an LDPC decoder. From the perspective of $x^{(2)}$, all its inputs are related to values of y, which are provided by its LDPC decoder. This setup is depicted in Fig. 3.

Concerning the encoding of multiple sources, [11] and [14] suggest that the decoders should not decode serially, since under the source-splitting assumption, entropy-approaching codes likely exist that do not require simultaneous decoding. We take a different approach, explicitly requiring the decoders to operate simultaneously, and designing them accordingly. This gives the system designer flexibility: if serial decoding is enforced, then once ϕ is fixed, the required rates of $x^{(1)}$, $x^{(2)}$, and y are also fixed;

however, simultaneous decoding permits a range of possible rates of the three sources for a given ϕ. Fixing ϕ may lead to lower complexity at the encoder by, for instance, fixing the memory sizes for $x^{(1)}$ and $x^{(2)}$. Certainly, simultaneous decoding will have a positive impact on probability of error and required code length.

B. Message calculation and DE

To take the perfectly known source symbols $x^{(1)}$ into account, we notice that perfect knowledge of one source letter is equivalent to perfect knowledge of the corresponding symbol x_i. Let $\hat{c}_i^{(y)}$ represent the message from x to y if x_i is perfectly known. Then

$$\hat{c}_i^{(y)} = \log \frac{p_{X_i,Y_i}(x_i, y_i = 0)}{p_{X_i,Y_i}(x_i, y_i = 1)}.$$

The actual value of the overall message from x to y is a function of t_i, as follows:

$$c_i^{(y)}(t_i) = \begin{cases} \hat{c}_i^{(y)}, & t_i = 0, \\ c_i^{(y)}, & t_i = 1; \end{cases} \tag{4}$$

where $c_i^{(y)}$ is the quantity calculated in (1). Meanwhile, the message $c_i^{(x)}$ from y to x is always given by the value from (1), since the values of y are never revealed.

Once again, we may invoke the equivalence of this decoder to that of a JBSC to simplify DE for this system, and maintain the analogy of $u_{1,i}$ to x and $u_{2,i}$ to y. The case where $t_i = 0$ is equivalent to exact knowledge of the element $z_{1,i}$ (or the symbol $u_{1,i}$), and $\hat{c}_i^{(u_2)}$ is a function of $[v_{1,i}, v_{2,i}]$. In density evolution, we perform a mixture of the densities of these two message, weighted by the source-splitting parameter ϕ. Techniques such as those used to calculate DE in section II may be re-used in this case.

IV. Sources with memory

A. Model

Although it is commonly assumed for convenience's sake that consecutive source values are memoryless, in reality it is extremely common for physical phenomena to be time-correlated. Once again, let $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$ represent binary sources observed at different encoders. Furthermore, for some discrete, finite alphabet $\mathcal{S}$, let $s \in \mathcal{S}^n$ represent a vector of hidden *source states*, and we assume that s forms a Markov chain with transition probability matrix $\mathbf{P}$, where the state transition probabilities are independent of x and y. For each possible state $\rho \in \mathcal{S}$, there exist source statistics $p_{X_i,Y_i}(x_i, y_i | s_i = \rho)$, so the source is parameterized by the matrix $\mathbf{P}$ and the set of source statistics $\{p_{X_i,Y_i}(x_i, y_i | s_i = \rho) : \rho \in \mathcal{S}\}$. For the remainder of the paper, we will be concerned with the case where $\mathcal{S} = \{1, 2\}$, and as a slight generalization of the model in [4], we allow the two conditional joint source PMFs to be arbitrary.

Encoding is performed in the same manner as in the memoryless case. In terms of decoding, the operations in the LDPC subgraphs are clearly the same as before, but the joint channel nodes must contend with the hidden Markov chain. There are two new messages passed through the Markov chain: the forward message α, passed from state s_i to s_{i+1}; the backward message β, passed from state s_{i+1} to state s_i, which are functions of the extrinsic messages $p_{X_i}^{(E)}(x_i)$ and $p_{Y_i}^{(E)}(y_i)$. These messages are given by

$$\begin{aligned} \alpha(s_{i+1}) &= K \sum_{x_i, y_i, s_i} p_{X_i,Y_i}(x_i, y_i | s_i) \alpha(s_i) \\ &\quad \cdot p_{S_{i+1}}(s_{i+1} | s_i) p_{X_i}^{(E)}(x_i) p_{Y_i}^{(E)}(y_i) \\ \beta(s_i) &= K \sum_{x_i, y_i, s_{i+1}} p_{X_i,Y_i}(x_i, y_i | s_i) \beta(s_{i+1}) \\ &\quad \cdot p_{S_{i+1}}(s_{i+1} | s_i) p_{X_i}^{(E)}(x_i) p_{Y_i}^{(E)}(y_i) \end{aligned}$$

where K represents an optional normalization constant. The calculation of the messages $c_{1,i}$ and $c_{2,i}$ are also modified, but we omit them for reasons of space.

B. A comment on source splitting

In the memoryless case, the source was split using a random sequence t, which divided the original source x into two sub-sources. In a source with memory with length n, we take the first $n\phi$ source symbols as $x^{(1)}$, and the remaining $n(1-\phi)$ source symbols as $x^{(2)}$. Since x is an arbitrarily long hidden Markov source, then if $\phi > 0$ and for almost all $\mathbf{P}$, the entropy rates and conditional entropy rates for the subsets $x^{(1)}$ and $x^{(2)}$ will be $H(X^{(1)}) = H(X^{(1)}|X^{(2)}) = \phi H(X)$, and $H(X^{(2)}) = H(X^{(2)}|X^{(1)}) = (1-\phi)H(X)$. Thus their combined entropy rate will be $H(X^{(1)}) + H(X^{(2)}|X^{(1)}) = \phi H(X) + (1-\phi)H(X) = H(X)$. Practically, the asymptotically small loss in splitting the source x down the middle can be mitigated by using the decoded version of $x^{(1)}$ to aid in decoding $x^{(2)}$, which is permitted under source splitting.

From the perspective of source y, splitting x into two chunks seems to lead to two different channels, one benefiting from the perfectly known $x^{(1)}$, and the other having no such benefit. We instead create a single equivalent channel by implementing an interleaver (known by the decoder) at the encoder for y. Because the interleaver is known at the decoder, the source memory is preserved, but the known instants of $x^{(1)}$ are spread randomly throughout the source from the perspective of the decoder for y. This step is optional, and we do it to reduce the complexity of density evolution.

C. Density evolution

We firstly extend our observation on decoder equivalence to the case of joint sources with Markov memory. Define a Markov-modulated joint binary symmetric channel (MMJBSC) as a generalization of the JBSC to the case where a hidden Markov channel state selects the joint inversion probabilities. That is, there exists a hidden Markov state sequence s with transition probability matrix $\mathbf{P}$, and conditioned on the state s_i, the channel at any given time is a JBSC with noise process PMF $\Pr_{Z_{1,i},Z_{2,i}}(z_{1,i}, z_{2,i} \,|\, s_i)$.

In particular, letting $\Pr_{X_i,Y_i}(x_i, y_i|s_i)$ represent the joint conditional source statistics, if $\Pr_{X_i,Y_i}(z_{1,i}, z_{2,i}|s_i) = \Pr_{Z_{1,i},Z_{2,i}}(z_{1,i}, z_{2,i}|s_i)$ for every setting of $[z_{i,1}, z_{i,2}]$ and s_i, and the transition probability matrices $\mathbf{P}$ are the same, then it is easy to show that the two decoders have the same probability of error.

Armed with this observation, we can use density evolution to analyze the simultaneous decoder for a source with memory. There are many possible ways to do so; for instance, the density evolution may be computed explicitly, as in the case of the Gilbert-Elliott channel in [15]. Another possibility is to use Monte Carlo simulation to obtain a histogram of the output messages for the source state estimator. We used the Monte Carlo method, since it easily scales to a large number of states.

V. Results

We are given two correlated sources, possibly with time dependence. In designing our two encoders, we fix the two check degree sequences, $\rho^{(x)}$ and $\rho^{(y)}$, and vary the variable degree sequences, $\lambda^{(x)}$ and $\lambda^{(y)}$, to minimize the transmission rate for a given source and a given rate-splitting parameter ϕ. A maximum variable degree was enforced, and differential evolution was used to search for good codes. The total rate of the Slepian-Wolf encoder is given by

$$R = \frac{\sum_i \rho_i^{(y)}/i}{\sum_j \lambda_j^{(y)}/j} + \frac{\sum_i \rho_i^{(x)}/i}{\sum_j \lambda_j^{(x)}/j}\phi + R_{x^{(1)}},$$

where $R_{x^{(1)}}$ is the rate used in encoding the independently coded portion of x. The objective is then to get as close to the Slepian-Wolf bound as possible while preserving the ability to decode the codes. For each candidate, consisting of $(\lambda^{(x)}, \rho^{(x)})$, $(\lambda^{(y)}, \rho^{(y)})$, and ϕ, we use DE to verify successful decoding.

For a memoryless source with joint probabilities given by

$$p_{X_i,Y_i}(x_i, y_i) = \begin{cases} 0.473, & x_i = y_i, \\ 0.027, & x_i \neq y_i; \end{cases},$$

with joint entropy $H(X, Y) = 1.3032$. Setting $\phi = 0.2$, we have found degree sequences given as follows:

$$\begin{aligned} \lambda^{(x)} &= [0\ 0.4068\ 0.4347\ 0.0425\ 0\ 0\ 0.0804\ 0.0356] \\ \rho^{(x)} &= [0\ 0\ 0\ 0\ 0\ 0\ 1] \\ \lambda^{(y)} &= [0\ 0.4333\ 0.2866\ 0.0357\ 0\ 0\ 0.1132\ 0.1312] \\ \rho^{(y)} &= [0\ 0\ 0\ 0\ 0\ 0\ 1] \end{aligned}$$

with rates $R_{x^{(2)}} = 0.4446$ and $R_y = 0.4712$. Assuming $x^{(1)}$ is encoded at entropy, the overall rate is $R = 1.3602$, roughly 4% greater than the joint entropy. For higher check degrees, we have found a code

$$\begin{aligned} \lambda^{(x)} &= [0\ 0.3119\ 0.4255\ 0.125\ 0\ 0\ 0.0843\ 0.04\ 0.0133] \\ \rho^{(x)} &= [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1] \\ \lambda^{(y)} &= [0\ 0.3317\ 0.2722\ 0.0078\ 0\ 0\ 0.0678\ 0.2783\ 0.0422] \\ \rho^{(y)} &= [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1] \end{aligned}$$

with rates $R_{x^{(2)}} = 0.4110$ and $R_y = 0.4643$. Assuming $x^{(1)}$ is encoded at entropy, the overall rate is $R = 1.3465$, roughly 3% greater than the joint entropy.

For a source with memory, with $p_{X_i,Y_i}(x_i, y_i|s_i)$ given by

x_i, y_i	0,0	0,1	1,0	1,1
$s_i = 1$	0.86	0.02	0.02	0.1
$s_i = 2$	0.6	0.05	0.05	0.3

and transition probability matrix $\mathbf{P}$ with 0.95 in the diagonal elements and 0.05 in the off-diagonal elements, we have found degree sequences given by $\lambda_2^{(x)} = 0.1661$, $\lambda_3^{(x)} = 0.7183$, $\lambda_5^{(x)} = 0.0024$, $\lambda_7^{(x)} = 0.1132$; $\lambda_2^{(y)} = 0.2608$, $\lambda_3^{(y)} = 0.6356$, $\lambda_4^{(y)} = 0.0024$, $\lambda_7^{(y)} = 0.0988$, $\lambda_9^{(y)} = 0.0024$; and $\rho_6^{(x)} = \rho_6^{(y)} = 1$, again with $\phi = 0.2$. The joint entropy rate of the source is given by $H(X, Y) = 1.1154$, and the two given codes have rates of $R_{x^{(2)}} = 0.4914$ and $R_y = 0.4665$ (assuming $x^{(1)}$ is encoded at entropy rate), which approaches to about 6% of the bound.

References

[1] A. D. Wyner, "Recent results in the Shannon theory," *IEEE Trans. Inform. Theory*, vol. 44, no. 1, pp. 2-10, Jan. 1974.

[2] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): Design and construction," *IEEE Trans. Inform. Theory*, vol. 49, no. 3, pp. 626-643, Mar. 2003.

[3] D. Schonberg, K. Ramchandran, and S. S. Pradhan, "LDPC codes can approach the Slepian Wolf bound for general binary sources," *Proc. 40th Allerton Conf. on Commun., Control, and Computing*, Monticello, IL, 2002.

[4] J. Garcia-Frias and W. Zhong, "LDPC codes for compression of multi-terminal sources with hidden Markov correlation," *IEEE Commun. Letters*, vol. 7, no. 3, pp. 115-117, Mar. 2003.

[5] T. Tian, J. Garcia-Frias, and W. Zhong, "Density evolution analysis of correlated sources compressed with LDPC codes," *Proc. 2003 ISIT*, Yokohama, Japan, 2003.

[6] T. J. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599-618, Feb. 2001.

[7] B. Rimoldi and R. Urbanke, "Asynchronous Slepian-Wolf coding via source-splitting," *Proc. 1997 ISIT*, Ulm, Germany, 1997.

[8] F. Cabarcas and J. Garcia-Frias, "Approaching the Slepian-Wolf boundary using practical channel codes," *Proc. 2004 ISIT*, Chicago, IL, 2004.

[9] D. Schonberg, K. Ramchandran, and S. S. Pradhan, "Distributed code constructions for the entire Slepian-Wolf rate region for arbitrarily correlated sources," *Proc. 2004 Data Compression Conference (DCC)*, Snowbird, UT, 2004.

[10] T. P. Coleman, A. H. Lee, M. Médard, and M. Effros, "On some new approaches to practical Slepian-Wolf compression inspired by channel coding," *Proc. 2004 DCC*, Snowbird, UT, 2004.

[11] A. D. Liveris, C.-F. Lan, K. R. Narayanan, Z. Xiong, and C. N. Georghiades, "Slepian-Wolf coding of three binary sources using LDPC codes," *Proc. 3rd Intl. Symp. on Turbo codes*, pp. 63-66, Brest, France, 2003.

[12] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 498-519, Feb. 2001.

[13] S. Cheng and Z. Xiong, "Channel symmetry in Slepian-Wolf code design based on LDPC codes with application to the quadratic Gaussian Wyner-Ziv problem," *Proc. 2004 ITW*, San Antonio, TX, 2004.

[14] C.-F. Lan, A. D. Liveris, K. R. Narayanan, Z. Xiong, and C. N. Georghiades, "Slepian-Wolf coding of multiple M-ary sources using LDPC codes," *Proc. 2004 DCC*, Snowbird, UT, 2004.

[15] A. W. Eckford, F. R. Kschischang, and S. Pasupathy, "Analysis of low-density parity-check codes for the Gilbert-Elliott channel," to appear in *IEEE Trans. Inform. Theory*.