Adaptive Distributed Source Coding

David Varodayan, Member, IEEE, Yao-Chung Lin, Student Member, IEEE, and Bernd Girod, Fellow, IEEE

Abstract—We consider distributed source coding in the presence of hidden variables that parameterize the statistical dependence among sources. We derive the Slepian-Wolf bound and devise coding algorithms for a block-candidate model of this problem. The encoder sends, in addition to syndrome bits, a portion of the source to the decoder uncoded as doping bits. The decoder uses the sum-product algorithm to simultaneously recover the source symbols and the hidden statistical dependence variables. We also develop novel techniques based on density evolution to analyze the coding algorithms. We confirm experimentally that our density evolution analysis closely approximates the proof’s selection of a constructive, the method of proof by random binning suggests using channel codes such as turbo codes [7]–[9] or low-density parity-check (LDPC) codes [10]. When LDPC codes are used, the encoder generates the syndrome of X. This encoding is a binning operation since an entire coset of X maps to the same syndrome. The decoder recovers X from its syndrome by iterative LDPC decoding using Y as side information. This practical decoding approximates the proof’s selection of a coset member that is jointly typical with Y.

In practice, the degree of statistical dependence between source and side information is varying in time and unknown in advance. So, it is better to model the bound on coding rate as the conditional entropy rate \( H(X|Y) \), \( R_Y \geq H(Y|X), \) \( R_X + R_Y \geq H(X,Y) \). Setting \( R_Y = H(Y) \) results in a special case shown in Fig. 1(b). Since we can achieve this rate by conventional entropy coding techniques, we consider \( Y \) available at the decoder as side information. The Slepian-Wolf conditions now reduce to \( R_X \geq H(X|Y) \), the conditional entropy of X given Y. In the remainder of this paper, we consider only this case, called source coding with side information at the decoder.

Although the proof of the Slepian-Wolf theorem is non-constructive, the method of proof by random binning suggests realistic models of the statistical dependence between source and side information often involve hidden variables. In [13]–[15], the statistical dependence is through a hidden vector connecting a pair of consecutive video frames. This paper develops and analyzes adaptive distributed source coding, a distributed source coding technique for adapting to hidden variables that parameterize the statistical dependence among sources.

I. INTRODUCTION

DISTRIBUTED source coding, the separate encoding and joint decoding of statistically dependent sources, has found numerous applications to video, such as low-complexity video encoding [1], [2], multiview coding [3], [4], and reduced-reference video quality monitoring [5]. In most implementations including those cited above, the decoding algorithm models the statistical dependence as direct correlations between pairs of samples. But the statistics of real video signals are often better modeled through hidden random variables that parameterize the sample-to-sample relationships. An example of such hidden variables might be the motion vectors connecting a pair of consecutive video frames. This paper develops and analyzes adaptive distributed source coding, a distributed source coding technique for adapting to hidden variables that parameterize the statistical dependence among sources.

A. Background on Distributed Source Coding

Fig. 1(a) depicts the lossless distributed source coding of two statistically dependent sources \( X \) and \( Y \), each of which are finite-alphabet random sequences of independent and identically distributed samples. The Slepian-Wolf theorem [6] states that \( X \) and \( Y \) can be recovered losslessly (up to an arbitrarily small probability of error) if and only if the coding rates \( R_X \) and \( R_Y \) satisfy the following entropy bounds:

\[
R_X \geq H(X|Y), \quad R_Y \geq H(Y|X), \quad R_X + R_Y \geq H(X,Y).
\]

Setting \( R_Y = H(Y) \) results in a special case shown in Fig. 1(b). Since we can achieve this rate by conventional entropy coding techniques, we consider \( Y \) available at the decoder as side information. The Slepian-Wolf conditions now reduce to \( R_X \geq H(X|Y) \), the conditional entropy of X given Y. In the remainder of this paper, we consider only this case, called source coding with side information at the decoder.

This paper was presented in part at IEEE International Conference on Image Processing, Hong Kong, September 2010.

David Varodayan is with the Department of Civil and Environmental Engineering, Stanford University, Stanford, CA 94305. Email: varodayan@stanford.edu.

Yao-Chung Lin and Bernd Girod are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305. Email: yclin79@stanfordalumni.org, bgirod@stanford.edu.

Fig. 1. Block diagrams of lossless distributed source coding with two signals: (a) separate encoding and joint decoding and (b) source coding with side information at the decoder.
Markov model, so the Baum-Welch algorithm [16] is used to learn the model. Likewise, we use expectation maximization [17] to decode images while adapting to hidden disparity [18]–[20] and motion [21].

B. Outline and Contributions

This paper treats adaptive distributed source coding comprehensively, making the following original contributions. Section II defines a model that relates the source signal to different candidates of side information through hidden random variables, and derives and evaluates the Slepian-Wolf bound in exact and approximate forms. In Section III, we describe the encoder and decoder, the latter completely in terms of the sum-product algorithm on a factor graph [22], and analyze the asymptotic complexity of the decoder. The factor graph formulation permits the analysis of coding performance via density evolution [23] in Section IV. The experimental results in Section V confirm that the density evolution analysis closely approximates practical coding performance. Section VI demonstrates an application of adaptive distributed source coding to video quality monitoring and channel tracking.

II. MODEL FOR SOURCE AND SIDE INFORMATION

A. Block-Candidate Model

Define the source \( X \) to be an equiprobable random binary vector of length \( n \) and the side information \( Y \) to be a random binary matrix of dimension \( n \times c \),

\[
X = \begin{pmatrix}
X_1 \\
X_2 \\
\vdots \\
X_n
\end{pmatrix}
\quad \text{and} \quad
Y = \begin{pmatrix}
Y_{1,1} & Y_{1,2} & \cdots & Y_{1,c} \\
Y_{2,1} & Y_{2,2} & \cdots & Y_{2,c} \\
\vdots & \vdots & \ddots & \vdots \\
Y_{n,1} & Y_{n,2} & \cdots & Y_{n,c}
\end{pmatrix}.
\]

(2)

Assume that \( n \) is a multiple of the block size \( b \). Then we further define blocks of \( X \) and \( Y \); namely, vectors \( x[i] \) of length \( b \) and matrices \( y[i] \) of dimension \( b \times c \). Finally, define a candidate \( y[i,j] \) to be a vector of length \( b \) that is the \( j \)th column of block \( y[i] \).

The statistics of the block-candidate model are illustrated in Fig. 2. The dependence between \( X \) and \( Y \) is through the vector \( Z = (Z_1, Z_2, \ldots, Z_2) \) of hidden random variables, each of which is uniformly distributed over \( \{1, 2, \ldots, c\} \). Given \( Z_i = z_i \), the block \( x[i] \) has a binary symmetric relationship of crossover probability \( \epsilon \) with the candidate \( y[i,z_i] \). That is, \( y[i,z_i] = x[i] + n[i] \) modulo 2, where \( n[i] \) is a random binary vector with independent elements equal to 1 with probability \( \epsilon \). All other candidates \( y[i,j] \neq z_i \) in this block are equiprobable random vectors, independent of \( x[i] \).

The block-candidate model introduced here is inspired by practical distributed source coding problems. For example, in low-complexity video encoding [21], the source \( X \) is the video frame being encoded. The blocks \( x[i] \) form a regular tiling of this frame with each tile comprising \( b \) pixels. The candidates \( y[i,j] \) of a block of \( Y \) represent the regions of the previously decoded frame that are searched among to find the one that matches \( x[i] \). Thus, the number of candidates \( c \) is the size of the notion search range in the previously decoded frame for each tile in the frame being encoded. In Section VI, we relate the block-candidate model to another problem in video quality monitoring and channel tracking.

B. Slepian-Wolf Bound

The Slepian-Wolf bound for the block-candidate model is the conditional entropy rate \( \mathcal{H}(X|Y) \), which can be expressed as

\[
\mathcal{H}(X|Y) = \mathcal{H}(X|Y,Z) + \mathcal{H}(Z|Y) - \mathcal{H}(Z|X,Y).
\]

(3)

The first term \( \mathcal{H}(X|Y,Z) = H(\epsilon) = -\epsilon \log_2 \epsilon - (1 - \epsilon) \log_2(1 - \epsilon) \) bit/bit, since no information about the hidden variables \( Z \) is revealed by the side information \( Y \) alone. Per block of \( b \) bits, each variable \( Z_i \) is uniformly distributed over \( c \) values.

The second term \( \mathcal{H}(Z|Y) = \mathcal{H}(Z) = \frac{1}{b} H(Z_i) = \frac{1}{b} \log_2 c \) bit/bit, since no information about the hidden variables \( Z \) is revealed by the side information \( Y \) alone. Per block of \( b \) bits, each variable \( Z_i \) is uniformly distributed over \( c \) values.

The third term \( \mathcal{H}(Z|X,Y) \) can be computed exactly by enumerating all joint realizations of blocks \( (x[i],y[i]) = (x,y) \) along with their probabilities \( P(x,y) \) and entropy terms \( H(Z_i|x,y) \).

\[
\mathcal{H}(Z|X,Y) = \frac{1}{b} H(Z_i|x[i],y[i]) \quad \text{(4)}
\]

\[
= \frac{1}{b} \sum_{x,y} P(x,y) H(Z_i|x,y) \quad \text{(5)}
\]

\[
= \frac{1}{b} \sum_{y|x=0} P(y|x=0) H(Z_i|x=0,y) \quad \text{(6)}
\]
The final equality sets $x$ to 0 because the probability and entropy terms are unchanged by flipping any bit in $x$ and the collocated bits in the candidates of $y$. Since the term $H(Z|y = 0)$ only depends on the number of bits in each candidate equal to 1, the calculation is tractable for small values of $b$ and $c$.

One way to approximate $H(Z|X, Y)$ is to evaluate the sum in (6) for only the realizations $y|x = 0$ that are strongly typical; that is, the statistically dependent candidate contains $\epsilon b$ bits of value 1 and all other candidates contain $\frac{1}{2} b$ bits of value 1 each. The typicality approximation gets better for larger values of $b$, for which the asymptotic equipartition property [24] deems \[
\sum_{y \text{ typical}} P(y|x = 0) \approx 1. \]
Thus,
\[
H(Z|X, Y) \approx \frac{1}{b} H(Z|y = 0, y \text{ strongly typical}) = \frac{1}{b} (-p_{\text{dep}} \log_2 p_{\text{dep}} - (c - 1) p_{\text{ind}} \log_2 p_{\text{ind}}), \tag{7}
\]
where
\[
p_{\text{dep}} = \frac{w_{\text{dep}}}{w_{\text{dep}} + (c - 1) w_{\text{ind}}}, \tag{8}
\]
\[
p_{\text{ind}} = \frac{w_{\text{ind}}}{w_{\text{dep}} + (c - 1) w_{\text{ind}}}. \tag{9}
\]
Here, $w_{\text{dep}} = (1 - \epsilon)^{(1-\epsilon)b} \epsilon^{b}$ and $w_{\text{ind}} = (1 - \epsilon)^{\frac{1}{2} b} \epsilon^{\frac{1}{2} b}$ are likelihood weights of the statistically dependent and independent candidates, respectively, being identified as the statistically dependent candidate. In this way, the expression in (8) computes the entropy of the index of the statistically dependent candidate.

Fig. 3 plots the Slepian-Wolf bounds for the block-candidate model as conditional entropy rates $H(X|Y)$, in exact form for tractable combinations of $b$ and $c$ and approximated under the typicality assumption for $b = 64$. The two computations agree for the combination $b = 64, c = 2$. We see that the block-candidate model offers greater potential compression than $H(\epsilon)$ by the dominance of the first two terms in the expression for $H(X|Y)$ in (3).

III. ADAPTIVE DISTRIBUTED SOURCE CODER

A. Encoder

The encoder codes $X$ into two segments. The doping bits are a sampling of the bits of $X$ sent directly at a fixed rate $R_{\text{fixed}}$ bit/bit. The syndrome bits of a rate-adaptive LDPC code are sent at a variable rate $R_{\text{adaptive}}$ bit/bit [12].

The purpose of doping, as also observed in [15], is to initialize the decoder with reliable information about $X$. The doping pattern is deterministic and regular, so that each block $x[i]$ contributes either $[bR_{\text{fixed}}]$ or $[bR_{\text{fixed}}]$ doping bits. The rate-adaptive LDPC code is constructed as described in [12] with code length $n$ set to the length of $X$. The variable rate $R_{\text{adaptive}} = \frac{t}{n}$, where $k$ is encoded data increment size and $t$ is the number of increments sent. For convenience, $R_{\text{fixed}}$ is chosen in advance to be a multiple of $\frac{k}{n}$ as well.

B. Decoder

The decoder recovers the source $X$ from the doping bits and the syndrome bits so far received from the encoder in conjunction with the block-candidate side information $Y$. We denote the received vectors of doping bits and syndrome bits as $(D_1, D_2, \ldots, D_{nR_{\text{fixed}}})$ and $(S_1, S_2, \ldots, S_{nR_{\text{adaptive}}})$, respectively.

The decoder synthesizes all this information by applying the sum-product algorithm on a factor graph structured like the one shown in Fig. 4 [22]. Each source node, which represents a source bit, is connected to doping, syndrome and side information nodes that bear some information about that source bit. The edges carry messages that represent probabilities about the values of their attached source bits. The sum-product algorithm iterates by message passing among the nodes so that ultimately all the information is shared across the entire factor graph. The algorithm converges successfully when the source bit estimates, when thresholded, are consistent with the vector of syndrome bits.

The nodes in the graph accept input messages from and produce output messages for their neighbors. These messages are probabilities that the connected source bits are 0, and are denoted by vectors $(p_{\text{in}}^1, p_{\text{in}}^2, \ldots, p_{\text{in}}^d)$ and $(p_{\text{out}}^1, p_{\text{out}}^2, \ldots, p_{\text{out}}^d)$, respectively, where $d$ is the degree of the node. By the sum-product rule, each output message $p_{\text{out}}^v$ is a function of all input messages except $p_{\text{in}}^v$, the input message on the same edge. Each source node additionally produces an estimate $p_{\text{est}}^v$ that its bit is 0, based on all the input messages. We now detail the computation rules of the nodes shown in Fig. 5.

1) Source Node Unattached to Doping Node: Fig. 5(a) shows a source node unattached to a doping node. There are two outcomes for the source bit random variable: it is 0 with likelihood weight $\prod_{v=1}^d p_{\text{in}}^v$ or 1 with likelihood weight $\prod_{v=1}^d (1 - p_{\text{in}}^v)$. Consequently,
\[
p_{\text{est}}^v = \frac{\prod_{v=1}^d p_{\text{in}}^v}{\prod_{v=1}^d p_{\text{in}}^v + \prod_{v=1}^d (1 - p_{\text{in}}^v)} \tag{11}
\]
Ignoring the input message $p_{\text{in}}^v$, the weights are $\prod_{v\neq u} p_{\text{in}}^v$ and $\prod_{v\neq u} (1 - p_{\text{in}}^v)$, so
\[
p_{\text{out}}^v = \frac{\prod_{v\neq u} p_{\text{in}}^v}{\prod_{v\neq u} p_{\text{in}}^v + \prod_{v\neq u} (1 - p_{\text{in}}^v)} \tag{12}
\]

2) Source Node Attached to Doping Node: Fig. 5(b) shows a source node attached to a doping node of binary value $D$. Recall that the doping bit specifies the value of the source bit, so the source bit estimate and the output messages are independent of the input messages,
\[
p_{\text{est}}^v = p_{\text{out}}^v = 1 - D. \tag{13}
\]
D1 S1 S2 sum equal to S
binary value S is equal to
with edge-perspective source and syndrome degree distributions

Fig. 4. Factor graph for adaptive distributed source decoder with parameters: code length n = 32, block size b = 8, number of candidates c = 4, R_{\text{fixed}} = \frac{1}{3}, and R_{\text{adaptive}} = \frac{1}{2}. The associated rate-adaptive LDPC code is regular with edge-perspective source and syndrome degree distributions \lambda_s(\omega) = \omega^2 and \rho_t(\omega) = \omega^3, respectively, where the code counter \(t = \frac{n}{b}R_{\text{adaptive}}\).

3) Syndrome Node: Fig. 5(c) shows a syndrome node of binary value S. Since the connected source bits have modulo 2 sum equal to S, the output message \(p_u^{\text{out}}\) is the probability that the modulo 2 sum of all the other connected source bits is equal to S. We argue by mathematical induction on \(d\) that

\[
p_u^{\text{out}} = \frac{1}{2} + \frac{1 - 2S}{2} \prod_{v \neq u} (2p_v^{\text{in}} - 1).
\]

4) Side Information Node: Fig. 5(d) shows a side information node of value \(y[i]\) consisting of \(b \times c\) bits, labeled in Fig. 5(d). As for the other types of node, the computation of the output message \(p_u^{\text{out}}\) depends on all input messages except \(p_u^{\text{in}}\). But since the source bits and the bits of the dependent candidate are related through a crossover probability \(\epsilon\), we define the noisy input probability of a source bit being 0 by

\[
p_v^{\text{noisy-in}} = (1 - \epsilon)p_v^{\text{in}} + \epsilon(1 - p_v^{\text{in}}).
\]

In computing \(p_u^{\text{out}}\), the likelihood weight of the candidate \(y[i,j]\) being the statistically dependent candidate equals the product of the likelihoods of that candidate’s \(b - 1\) bits excluding \(Y_{u,j}\),

\[
w_{u,j} = \prod_{v \neq u} \left( (\mathbb{I}[Y_{v,j} = 0]p_v^{\text{noisy-in}} + \mathbb{I}[Y_{v,j} = 1](1 - p_v^{\text{noisy-in}})) \right),
\]

where \(\mathbb{I}[\cdot]\) is the indicator function. We finally marginalize \(p_u^{\text{out}}\) as the normalized sum of weights for which \(Y_{u,j}\) is 0, passed through crossover probability \(\epsilon\),

\[
p_u^{\text{clean-out}} = \frac{1}{c} \sum_{j=1}^{c} \mathbb{I}[Y_{u,j} = 0]w_{u,j},
\]

\[
p_u^{\text{out}} = (1 - \epsilon)p_u^{\text{clean-out}} + \epsilon(1 - p_u^{\text{clean-out}}).
\]

C. Decoding Complexity

We now analyze the asymptotic complexity of the decoder as a function of block size b and number of candidates c. We need only consider the complexity of the \(\frac{n}{b}\) side information nodes because the rest of the factor graph is equivalent to an LDPC decoder of complexity \(O(n)\) [25] (assuming fixed degree distributions).

In each side information node, the complexity limiting computation is (16). This step computes \(bc\) likelihood weights,
we see that the numerator need only be calculated once for each value of \( i \). Therefore, the per-node complexity is \( O(nc) \). Since there are \( \frac{n}{b} \) such nodes, the overall decoding complexity is \( O(nc) \).

IV. Density Evolution Analysis

We use density evolution to determine whether the sum-product algorithm converges on the proposed factor graph [23]. Our approach first transforms the factor graph into a simpler one that is equivalent with respect to convergence. Next we define degree distributions for this graph. Finally, we describe density evolution for the adaptive decoder.

A. Factor Graph Transformation

The convergence of the sum-product algorithm is invariant under manipulations to the source, side information, syndrome and doping bits and the factor graph itself as long as the messages passed along the edges are preserved up to relabeling.

The first simplification is to reorder the candidates within each side information block so that statistically dependent candidate is in the first position \( y[i, 1] \). This shuffling has no effect on the messages.

We then replace each side information candidate \( y[i, j] \) with the modulo 2 sum of itself and its corresponding source block \( x[i] \), and set all the source bits, syndrome bits and doping bits to 0. The values of the messages would be unchanged if we would relabel each message to stand for the probability that the attached source bit (which is now 0) is equal to the original value of that source bit.

Finally, observe that any source node attached to a doping node always outputs deterministic messages equal to 1, since the doping bit \( D \) is set to 0 in (13). We therefore remove all instances of this node combination along with all their attached edges from the factor graph. In total, a fraction \( R_{\text{fixed}} \) of the source nodes are removed. Although some edges are removed at some syndrome nodes, no change is required to the syndrome node decoding rule because ignoring input messages \( p_v^{\text{in}} = 1 \) does not change the term \( \prod_{u \neq i} (2p_v^{\text{in}} - 1) \) in (14).

In contrast, side information nodes with edges removed must substitute the missing input messages with 1 in (15) to (18).

Applying these three manipulations to the factor graph of Fig. 4 produces the simpler factor graph, equivalent with respect to convergence, shown in Fig. 6, in which the values 0 and 1 are denoted light and dark, respectively. The syndrome nodes all have value 0, which is consistent with the source bits all being 0 as well. Only the side information candidates in the first position \( y[i, 1] \) are statistically dependent with respect to the source bits. In particular, the bits of \( y[i, 1] \) are independently equal to 0 with probability \( 1 - \epsilon \), while the

Fig. 5. Factor graph node combinations. Input and output messages and source bit estimate (if applicable) of (a) source node unattached to doping node, (b) source node attached to doping node, (c) syndrome node and (d) side information node.

Fig. 6. Transformed factor graph equivalent to that of Fig. 4 in terms of convergence. After doping, the edge-perspective source and syndrome degree distributions of the associated rate-adaptive LDPC code are \( \lambda_1(\omega) = \omega^2 \) and \( \rho_1(\omega) = \frac{2}{\pi} \omega^3 + \frac{2}{\pi} \omega^4 + \frac{12}{\pi^2} \omega^5 \). The edge-perspective side information degree distribution is \( \beta_1(\omega) = \frac{1}{\pi} \omega^6 + \frac{8}{\pi^2} \omega^7 \).
bits of $y[i, j \neq 1]$ are independently equiprobable. Note also the absence of combinations of source nodes linked to doping nodes.

B. Degree Distributions

Density evolution runs, not on a factor graph itself, but using degree distributions of that factor graph. Degree distributions exist in two equivalent forms, edge-perspective and node-perspective, and we represent both by polynomials in $\omega$. In an edge-perspective degree distribution polynomial, the coefficient of $\omega^d$ is the fraction of edges connected to a certain type of node of degree $d$ out of all edges connected to nodes of that type. In a node-perspective degree distribution polynomial, the coefficient of $\omega^d$ is the fraction of a certain type of node of degree $d$ out of all nodes of that type.

In total, we consider twelve degree distributions, six for each of two different graphs. The source, syndrome and side information degree distributions of the factor graph before transformation (like the one in Fig. 4) are respectively labeled $\lambda_i(\omega)$, $\rho_i(\omega)$ and $\beta_i(\omega)$ in edge perspective and $L_i(\omega)$, $R_i(\omega)$ and $B_i(\omega)$ in node perspective, where the code counter $t = \frac{\omega}{R_{\text{adaptive}}}$. Their expected counterparts in the factor graph after transformation (like the one in Fig. 6) are denoted $\lambda_i^*(\omega)$, $\rho_i^*(\omega)$ and $\beta_i^*(\omega)$ in edge perspective and $L_i^*(\omega)$, $R_i^*(\omega)$ and $B_i^*(\omega)$ in node perspective.

For source degree distributions $\lambda_i(\omega)$, $L_i(\omega)$, $\lambda_i^*(\omega)$ and $L_i^*(\omega)$, we count the source-side-edges, but neither the source-side-information nor source-doping edges. In this way, $\lambda_i(\omega)$, $\rho_i(\omega)$, $L_i(\omega)$ and $R_i(\omega)$ are the degree distributions of the rate-adaptive LDPC codes and, therefore, we take them as given in the following derivations.

1) Source Degree Distributions: During the factor graph transformation, a fraction $R_{\text{fixed}}$ of the source nodes are selected for removal regardless of their degrees. Therefore, the expected source degree distributions are preserved,

$$\lambda_i^*(\omega) = \lambda_i(\omega),$$

$$L_i^*(\omega) = L_i(\omega) = \frac{\int_0^\omega \lambda_i(\psi) d\psi}{\int_0^\omega \lambda_i(\psi) d\psi},$$

where the final step is by the edge-to-node-perspective conversion formula in [25].

2) Syndrome Degree Distributions: The factor graph transformation, by removing a fraction $R_{\text{fixed}}$ of the source nodes regardless of their degrees, removes the same fraction of source-syndrome edges in expectation. From the perspective of a syndrome node of original degree $d$, each edge is retained independently with probability $1 - R_{\text{fixed}}$. Consequently, the chance that it has degree $d^*$ after factor graph transformation is a binomial probability, denoted $\text{binopmf}(d, d^*, R_{\text{fixed}}) = \binom{d}{d^*} (1 - R_{\text{fixed}})^{d-d^*} R_{\text{fixed}}^{d-d^*}$. So if the node-perspective syndrome degree distribution before transformation is expressed as $R_i(\omega) = \sum_{d=1}^{d_{\text{max}}} A_d \omega^d$, then after transformation the expected node-perspective syndrome degree distribution is given by

$$R_i^*(\omega) = \sum_{d=1}^{d_{\text{max}}} \frac{A_d}{1 - (R_{\text{fixed}})R_{\text{fixed}}} \sum_{d^*=1}^d \text{binopmf}(d, d^*, R_{\text{fixed}}) \omega^{d^*},$$

where the normalization factors $\frac{1}{1 - (R_{\text{fixed}})R_{\text{fixed}}}$ account for the fact that degree $d^* = 0$ syndrome nodes are not included in the degree distribution. The edge-perspective $\rho_i^*(\omega)$ is obtained by differentiating and normalizing $R_i^*(\omega)$ according to the node-to-edge-perspective conversion formula in [25].

3) Side Information Degree Distributions: In the factor graph before transformation, all side information nodes have degree $b$, so the edge- and node-perspective side information degree distributions are

$$\beta_i(\omega) = \omega^{b-1},$$

$$B_i(\omega) = \omega^b,$$

Since the doping pattern is deterministic and regular at rate $R_{\text{fixed}}$, each side information node retains either $b^*$ or $b^* + 1$ edges in the transformed graph, where $b^* = \lfloor b(1 - R_{\text{fixed}}) \rfloor$. With fractional part $A = b(1 - R_{\text{fixed}}) - b^*$, the node- and edge-perspective side information degree distributions after transformation are

$$B_i^*(\omega) = (1 - A)\omega^{b^*} + A\omega^{b^*+1},$$

$$\beta_i^*(\omega) = \frac{(1 - A)b^*}{b^* + A}\omega^{b^* - 1} + \frac{A(b^* + 1)}{b^* + A}\omega^{b^*},$$

where $\beta_i^*(\omega)$ is obtained from $B_i^*(\omega)$ by differentiation and normalization.

C. Density Evolution

We now use densities to represent the distributions of messages passed among classes of nodes. The source-to-side-information, source-to-syndrome, syndrome-to-source and side-information-to-source densities are denoted $Q_{\text{so-si}}$, $Q_{\text{so-symp}}$, $Q_{\text{syn-so}}$ and $Q_{\text{si-so}}$, respectively. Another density $Q_{\text{source}}$ captures the distribution of source bit estimates.

Fig. 7 depicts a schematic of the density evolution process. The message densities are passed among three stochastic nodes that represent the side information, source and syndrome nodes. Inside the nodes are written the probabilities that the values at those positions are 0. Observe that the source and syndrome stochastic nodes are deterministically 0 and only the elements of the candidate in the first position of the side information stochastic node are biased towards 0, in accordance with the transformation in Section IV-A. Fig. 7 also shows the degree distributions of the transformed factor graph beside the stochastic nodes. Since every source node connects to exactly one side information node, the source degree distribution with respect to the side information nodes is 1.

During density evolution, each message density is stochastically updated as a function of the values and degree distributions associated with the stochastic node from which it originates and the other message densities that arrive at that stochastic node. After a fixed number of iterations of evolution,
To compute each updated sample $q_{\text{out}}$ of $Q_{\text{so-si}}$, let a set $\{q^\text{in}_{v j}\}_{v=1}^{b-1}$ consist of $\delta-1$ samples drawn randomly from $Q_{\text{so-si}}$ and $b-\delta$ samples equal to 1. The random degree $\delta$ is drawn equal to $d$ with probability equal to the coefficient of $\omega^{d-1}$ in edge-perspective $\beta^*_t(\omega)$, since there is one actual output message per edge. The samples set to 1 substitute for the messages on edges removed during factor graph transformation due to doping.

For each $q_{\text{out}}$, create also a realization of a $b \times c$ block of side information from the joint distribution induced by the side information stochastic node. That is, each element $Y_{v,j}$ is independently biased towards 0, with probability $1-\epsilon$ if $j=1$ or probability $\frac{1}{2}$ if $j \neq 1$.

Generate sets $\{q^\text{noisy-in}_{v}\}_{v=1}^{b-1}$ and $\{w_{j}\}_{j=1}^{c}$ before finally updating sample $q_{\text{out}}$, following (15) to (18).

\[
q^\text{noisy-in}_v = (1 - \epsilon)q^\text{in}_v + \epsilon(1 - q^\text{in}_v) \tag{29}
\]

\[
w_j = \prod_{v=1}^{b-1} (\mathbb{I}[Y_{v,j} = 0]q^\text{noisy-in}_v + \mathbb{I}[Y_{v,j} = 1](1 - q^\text{noisy-in}_v)) \tag{30}
\]

\[
q^\text{clean-out} = \frac{\sum_{j=1}^{c} \mathbb{I}[Y_{v,j} = 0]w_j}{\sum_{j=1}^{c} w_j} \tag{31}
\]

\[
q_{\text{out}} = (1 - \epsilon)q^\text{clean-out} + \epsilon(1 - q^\text{clean-out}) \tag{32}
\]

**V. Experimental Results**

We examine the role of doping in adaptive distributed source coding and analyze the complexity of the decoding algorithm. We find that good choices for the doping rate $R_{\text{fixed}}$ are required to achieve compression close to the Slepian-Wolf bound, and that density evolution analysis successfully guides those choices. Then we show empirically that the decoding complexity is consistent with the asymptotic value of $O(nc)$ obtained in Section III.

In these experiments, the adaptive distributed source code uses rate-adaptive LDPC codes of length $n = 4096$ bits, encoded data increment size $k = 32$ bits, and regular source degree distribution $\lambda_{\text{reg}}(\omega) = \omega^2$. Hence, $R_{\text{adaptive}}$ is in $\{\frac{1}{128}, \frac{1}{64}, \ldots, \frac{1}{2}\}$. For convenience, we allow $R_{\text{fixed}}$ to take values in $\{0, \frac{1}{128}, \frac{1}{64}, \ldots, \frac{1}{2}\}$. Our density evolution analysis is implemented as a Monte Carlo simulation using up to $2^{14}$ samples.

**A. Compression Performance With and Without Doping**

In Fig. 8, we fix block size $b = 64$ and number of candidates $c = 16$ and vary the entropy $H(\epsilon)$ of the noise between $X$ and.
the statistically dependent candidates of \( Y \). We first plot the coding rates achieved by the adaptive distributed source codec (averaged over 100 trials) and modeled by density evolution, when the doping rate \( R_{\text{fixed}} = 0 \). The performance is far from the Slepian-Wolf bound \( H(X|Y) \), since the adaptive decoder is not initialized with sufficient reliable information about \( X \). Nevertheless, density evolution models the poor empirical performance reasonably accurately.

Increasing the doping rate \( R_{\text{fixed}} \) initializes the decoder better, but increasing it too much penalizes the overall adaptive distributed source codec rate. We therefore search through all values \( R_{\text{fixed}} \in \{0, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots, \frac{1}{8}\} \) and let density evolution determine which \( R_{\text{fixed}} \) minimizes the coding rate for each \( H(\epsilon) \). Fig. 8 also shows the coding performance with these optimal doping rates. The adaptive distributed source codec operates close to the Slepian-Wolf bound, and is modeled by density evolution even better than when \( R_{\text{fixed}} = 0 \).

Notice that the optimal doping rates appear to decrease slowly as \( H(\epsilon) \) increases from 0 to 0.6 and increase sharply as \( H(\epsilon) \) increases from 0.6 to 0.8. (At \( H(\epsilon) = 0.9 \), coding is at full rate of 1 bit/bit, so no doping is necessary.) We explain these trends by considering the variation of additional doping bits. At low \( H(\epsilon) \), such bits are useful in identifying the matching candidate for each block. At high \( H(\epsilon) \), they are useful because not even the matching candidates provide very reliable information about the source bits. These results contrast with the observation in [15] that doping rate increases with conditional entropy due to the absence of multiple candidates of side information in that paper.

### B. Decoding Complexity

Fig. 9 plots average execution times for 100 iterations of the decoder’s sum-product algorithm using Matlab R2010b on one core of an Intel Core2 Duo 2.26 GHz processor. The times for 100 iterations are plotted because that is the maximum number of iterations allowed per attempted rate \( R_{\text{adaptive}} \). The maximum number of rates attempted is 128, but in practice it can be much fewer. In particular, we can use the results of density evolution to narrow down the possible range.

VI. APPLICATION EXAMPLE: VIDEO QUALITY MONITORING AND CHANNEL TRACKING

We now apply adaptive distributed source coding to a simple example of end-to-end video quality monitoring and show how density evolution is used to design the doping rate. Fig. 10 depicts a video transmission system with an attached system for reduced-reference video quality monitoring and channel tracking. A server sends \( c \) channels of encoded video to an intermediary. The intermediary transcodes the video into a single trajectory with channel changes (like the one shown in Fig. 11) and forwards it to a display device. We are interested in the attached monitoring and tracking system. The device encodes projection coefficients \( X \) of the video trajectory and sends them back to the server. A feedback channel from server to device is available for rate control. The server decodes and compares the coefficients with projection coefficients \( Y \) of the \( c \) channels of video, in order to estimate the transcoded video quality and track its channel change. We first describe the details of the video projection and peak signal-to-noise ratio (PSNR) estimator in Fig. 10. We then show that implementing the projection encoder and decoder with adaptive distributed source coding reduces the quality monitoring bit rate by about 75% compared to fixed length coding.
The video projection for both sets of coefficients $X$ and $Y$ is specified in ITU-T Recommendation J.240 [26], [27]. The projection partitions the luminance channel of the video into tiles, sizes of $16 \times 16$ or $8 \times 8$ pixels being typical. From each tile, a single coefficient is obtained by the process shown in Fig. 12. The tile is multiplied by a maximum length sequence [28], transformed using the 2D Walsh-Hadamard transform (WHT), multiplied by another maximum length sequence, and inverse transformed using the 2D inverse Walsh-Hadamard transform (IWH). Finally, one coefficient is sampled from the tile.

Recommendation J.240 also suggests a method to estimate the PSNR of the transcoded video with respect to the encoded video using the sets of coefficients $X$ and $Y$. Suppose that $\bar{X}$ and $\bar{Y}$, each of length $n$, denote the subvectors of coefficients of the transcoded trajectory and its matching encoded counterpart, respectively. Both vectors are assumed to be uniformly quantized with step size $Q$ into $\hat{X}$ and $\hat{Y}$. The PSNR of the transcoded video with respect to the original encoded video is estimated as

$$\text{MSE}_{J.240} = \frac{Q^2}{n} \sum_{i=1}^{n} (\hat{X}_i - \hat{Y}_i)^2$$  \hspace{1cm} (33)

$$\text{PSNR}_{J.240} = 10 \log_{10} \frac{255^2}{\text{MSE}_{J.240}}.$$  \hspace{1cm} (34)

But if the PSNR estimator is located at the server (as in Fig. 10), it has access to the unquantized coefficients $\bar{Y}$. In [29], we propose the following maximum likelihood (ML) estimation formulas, which support nonuniform quantization of $X$.

$$\text{MSE}_{\text{ML}} = \frac{1}{n} \sum_{i=1}^{n} E \left[ (\hat{X}_i - \hat{Y}_i)^2 | \hat{X}_i, \hat{Y}_i \right]$$  \hspace{1cm} (35)

$$\text{PSNR}_{\text{ML}} = 10 \log_{10} \frac{255^2}{\text{MSE}_{\text{ML}}}.$$  \hspace{1cm} (36)

In evaluating (35), we assume that $\hat{X}_i$ given $(\hat{X}_i, \hat{Y}_i)$ is distributed as a normalized truncated Gaussian with some variance $\sigma^2$, centered at $\bar{X}_i$ and truncated at the quantization boundaries surrounding $\hat{X}_i$. We iteratively update estimates of $\sigma^2$ and $\text{MSE}_{\text{ML}}$ until convergence using the technique in [30].

Fixed length coding is taken for granted by Recommendation J.240 for the projection encoder. Conventional variable length coding produces limited gains because the coefficients $X$ are approximately independent and identically distributed Gaussian random variables, due to the dimension reduction projection. In contrast, adaptive distributed source coding of $X$ offers better compression by exploiting the statistically related coefficients $Y$ at the projection decoder. This side information adheres to the block-candidate model with block size equal to $b$, the number of J.240 coefficients per frame, and number of candidates equal to $c$, the number of channels.

In the experiment, we use $c = 8$ channels of video (Foreman, Carphone, Mobile, Mother and Daughter, Table, News, Coastguard, Container) at resolution $176 \times 144$ and frame rate $30$ Hz. The first 256 frames of each sequence are encoded at the server using H.264/AVC [31] Baseline profile in IPPP... coding structure with quantization parameter set to 16. The intermediary transcodes each trajectory with JPEG using scaled versions of the quantization matrix specified in Annex K of the standard [32], with scaling factors 0.5, 1 and 2, respectively. The number of J.240 coefficients per frame, $b = 99$ or 396, depending on the tile size $16 \times 16$ or $8 \times 8$, respectively. Each coefficient is quantized to 1 bit, so that we can model the codec using density evolution exactly as described in Section IV. The multilevel extension of both the density evolution technique and these video quality monitoring results are presented in [33].

The optimal doping rates for the settings $(b = 99, c = 8)$ and $(b = 396, c = 8)$ are plotted in Fig. 13 for the entire range $0 \leq H(\epsilon) \leq 1$. Note that this density evolution analysis makes use of neither the test video sequences nor their statistics. We choose the values $R_{\text{fixed}} = \frac{1}{128}$ for $b = 99$ and $R_{\text{fixed}} = \frac{1}{128}$ for $b = 396$ because these are the maximum optimal doping rates for $H(\epsilon) \leq 0.2$, which covers about 90% of the empirical conditional entropy rates. The doping pattern is regular with respect to the side information nodes; that is the sequence is: $1, b + 1, 2b + 1, \ldots, n - b + 1, 2, b + 2, 2b + 2, \ldots, n - b + 2, \ldots$. In this way, the side information degree distributions are consistent with (25). A different choice of doping pattern may change the results, if it changes the degree distributions.

The codecs process 8 frames of coefficients at a time, using rate-adaptive LDPC codes with regular source degree distribution $\lambda_{\text{reg}}(\omega) = \omega^2$, length $n = 8b$ bits and data increment size $k = \frac{8b}{128}$ bits. Consequently, $R_{\text{adaptive}} \in \left\{ \frac{1}{128}, \frac{2}{128}, \ldots, 1 \right\}$.

The performance of different combinations of coding and estimation techniques is compared in Tables I and II for $b = 99$ and 396, respectively. The PSNR estimates are computed over group of picture (GOP) size of 256 frames. In all trials, the transcoded trajectory is correctly tracked and the PSNR estimation errors are computed with respect to the matching trajectory at the server. The quality monitoring bit rate is the transmission rate from projection encoder to decoder in kbit/s assuming the video has frame rate of 30 Hz. In both Tables I and II, PSNR estimation by the method in Recommendation J.240 yields very large mean absolute PSNR estimation errors that render the technique useless. The maximum likelihood
the convergence of the messages themselves, because the densities of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of distributions (or densities) of sum-product messages.

Table I: Video quality monitoring performance for $b = 99$, $c = 8$.

<table>
<thead>
<tr>
<th>Settings</th>
<th>Coding</th>
<th>Mean Absolute PSNR (dB)</th>
<th>Quality Monitoring Bit Rate (kbit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J.240</td>
<td>FLC</td>
<td>22.6</td>
<td>2.97</td>
</tr>
<tr>
<td>ML</td>
<td>FLC</td>
<td>1.1</td>
<td>2.97</td>
</tr>
<tr>
<td>ML</td>
<td>ADSC</td>
<td>1.1</td>
<td>0.81</td>
</tr>
</tbody>
</table>

VII. CONCLUSIONS

Adaptive distributed source coding enables the decoder to adapt to multiple candidates of side information without knowing in advance which candidate is most statistically dependent on the source. We define a block-candidate model for side information and compute its Slepian-Wolf bound. The iterative decoding algorithm is described entirely in terms of the sum-product algorithm on a factor graph. This formulation permits the analysis of coding performance via density evolution. The idea is that testing the convergence of distributions (or densities) of sum-product messages is more efficient than testing the convergence of the messages themselves, because the former does not require complete knowledge of the decoder’s factor graph. Our simulation experiments demonstrate that the analysis technique closely approximates empirical coding performance, and consequently enables tuning of parameters of the coding algorithms. The main finding is that the encoder usually sends a low rate of the source bits uncoded as doping bits in order to achieve coding performance close to the Slepian-Wolf bound. We finally apply our technique to a reduced-reference video quality monitoring and channel tracking system and design the codes using density evolution analysis. With adaptive distributed source coding, the reduced-reference bit rate is reduced by about 75% compared to fixed length coding.

REFERENCES


David Varodayan is currently a postdoctoral fellow with the Department of Civil and Environmental Engineering at Stanford University. He received the M.S. and Ph.D. degrees in Electrical Engineering at Stanford University in 2005 and 2010, respectively. He received the EURASIP Signal Processing Journal’s Most Cited Paper Award in 2009 and Best Student Paper Award on two occasions: IEEE Workshop on Multimedia Signal Processing in 2006 and European Signal Processing Conference in 2007. His research interests include distributed source coding, image and video processing, and signal processing for the smart grid. He is a Member of the IEEE.

Yao-Chung Lin received the Ph.D. degree from the Department of Electrical Engineering at Stanford University in 2010. He received the M.S. degree in Electrical Engineering and the B.S. degree in Computer Science and Information Engineering from National Chiao Tung University, Taiwan. His research interests include distributed source coding applications, multimedia systems, and video processing and compression.

Bernd Girod is Professor of Electrical Engineering and (by courtesy) Computer Science in the Information Systems Laboratory of Stanford University, California, since 1999. Previously, he was a Professor in the Electrical Engineering Department of the University of Erlangen-Nuremberg. His current research interests are in the areas of video compression, networked media systems, and image-based retrieval. He has published over 450 conference and journal papers, as well as 5 books, receiving the EURASIP Signal Processing Best Paper Award in 2002, the IEEE Multimedia Communication Best Paper Award in 2007, the EURASIP Image Communication Best Paper Award in 2008, as well as the EURASIP Technical Achievement Award in 2004. As an entrepreneur, Professor Girod has been involved with several startup ventures, among them Polycom, Vivo Software, 8x8, and RealNetworks. He received an Engineering Doctorate from University of Hannover, Germany, and an M.S. Degree from Georgia Institute of Technology. Prof. Girod is a Fellow of the IEEE, a EURASIP Fellow, and a member of the German National Academy of Sciences (Leopoldina).