Adaptive Distributed Source Coding

David Varodayan, *Member, IEEE*, Yao-Chung Lin, *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 (DE) to analyze the coding algorithms. We experimentally confirm that our DE analysis closely approximates practical performance. This result allows us to efficiently optimize parameters of the algorithms. In particular, we show that the system performs close to the Slepian–Wolf bound when an appropriate doping rate is selected. We then apply our coding and analysis techniques to a reduced-reference video quality monitoring system and show a bit rate saving of about 75% compared with fixed-length coding.**

***Index Terms*—Source coding, sum product algorithm, video signal processing.**

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 statistical dependence as a direct correlation between pairs of samples. However, 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 (ADSC), 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 losslessly recovered (up to an arbitrarily small probability of error) if and only if coding rates R_X and R_Y satisfy the following entropy bounds:

$$R_X \geq H(X|Y), \quad R_Y \geq H(Y|X), \quad R_X+R_Y \geq H(X,Y). \tag{1}$$

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, which is called source coding with side information at the decoder.

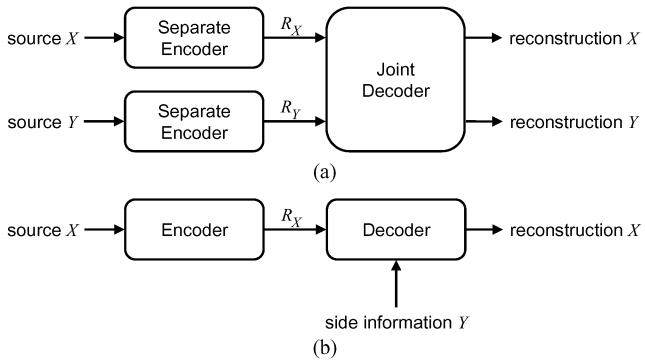


Fig. 1. Block diagrams of lossless distributed source coding with two signals: (a) separate encoding and joint decoding and (b) source coding with side information at the decoder.

Although the proof of the Slepian–Wolf theorem is nonconstructive, 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. Therefore, it is better to model the bound on coding rate as the conditional entropy rate $\mathcal{H}(\mathbf{X}|\mathbf{Y})$ of source and side information sequences $\mathbf{X}$ and $\mathbf{Y}$, respectively, rather than the conditional entropy $H(X|Y)$ of their samples. Practical codes might need to adapt their coding rates in response to varying statistics. We construct the first rate-adaptive LDPC codes in [11] and [12].

Realistic models of the statistical dependence between source and side information often involve hidden variables.

Manuscript received March 02, 2011; revised August 17, 2011; accepted October 31, 2011. Date of publication November 15, 2011; date of current version April 18, 2012. This paper was presented in part at the IEEE International Conference on Image Processing, Hong Kong, September 2010. The work in this paper was performed at the Department of Electrical Engineering, Stanford University, Stanford, CA. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Anthony Vetro.

The authors are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: varodayan@alumni.stanford.edu; yclin79@stanfordalumni.org; bgirod@stanford.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2011.2175936

In [13]–[15], the statistical dependence is through a hidden Markov model; hence, 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 ADSC 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 it derives and evaluates the Slepian–Wolf bound in exact and approximate forms. In Section III, we describe the encoder and the 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 (DE) [23] in Section IV. The experimental results in Section V confirm that the DE analysis closely approximates practical coding performance. Section VI demonstrates an application of ADSC to video quality monitoring and channel tracking.

II. Model for Source and Side Information

A. Block-Candidate Model

Define the source $\mathbf{X}$ to be an equiprobable random binary vector of length n and the side information $\mathbf{Y}$ to be a random binary matrix of dimension $n \times c$, i.e.,

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

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

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

The block-candidate model introduced here is inspired by practical distributed source coding problems. For example, in low-complexity video encoding [21], the source $\mathbf{X}$ is the video frame being encoded. The blocks $\mathbf{x}[i]$ form a regular tiling of this frame with each tile comprising b pixels. The candidates $\mathbf{y}[i, j]$ of a block of $\mathbf{Y}$ represent the regions of the previously decoded frame that are searched among to find the one that matches $\mathbf{x}[i]$. 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.

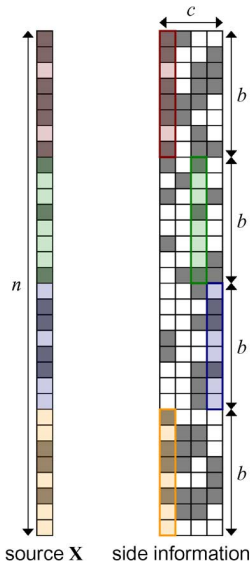


Fig. 2. Block-candidate model of statistical dependence. The source $\mathbf{X}$ is an equiprobable binary vector of length n bits, and the side information $\mathbf{Y}$ is a binary matrix of dimension $n \times c$. The binary values are shown as light/dark. For each block of b bits of $\mathbf{X}$ (among n/b such nonoverlapping blocks), the corresponding $b \times c$ block of $\mathbf{Y}$ contains exactly one candidate dependent on $\mathbf{X}$ (shown color coded). The dependence is binary symmetric with crossover probability ϵ. All other candidates are independent of $\mathbf{X}$.

B. Slepian–Wolf Bound

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

$$\mathcal{H}(\mathbf{X}|\mathbf{Y}) = \mathcal{H}(\mathbf{X}|\mathbf{Y}, \mathbf{Z}) + \mathcal{H}(\mathbf{Z}|\mathbf{Y}) - \mathcal{H}(\mathbf{Z}|\mathbf{X}, \mathbf{Y}). \quad (3)$$

The first term $\mathcal{H}(\mathbf{X}|\mathbf{Y}, \mathbf{Z}) = H(\epsilon) = -\epsilon \log_2 \epsilon - (1 - \epsilon) \log_2 (1 - \epsilon)$ bit/bit since each block $\mathbf{x}[i] = \mathbf{y}[i, z_i] + \mathbf{n}[i]$ modulo 2. Given $\mathbf{Y}$ and $\mathbf{Z}$, the only randomness is supplied by $\mathbf{n}[i]$, the random binary vectors with independent elements equal to 1 with probability ϵ.

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

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

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

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

$$= \frac{1}{b} \sum_{y} P\{y|x = 0\} H(Z_i|x = 0, y). \quad (6)$$

The final equality sets x to 0 because the probability and entropy terms are unchanged by flipping any bit in x and the colocated bits in the candidates of y. Since the term $H(Z_i|x = 0, y)$ only

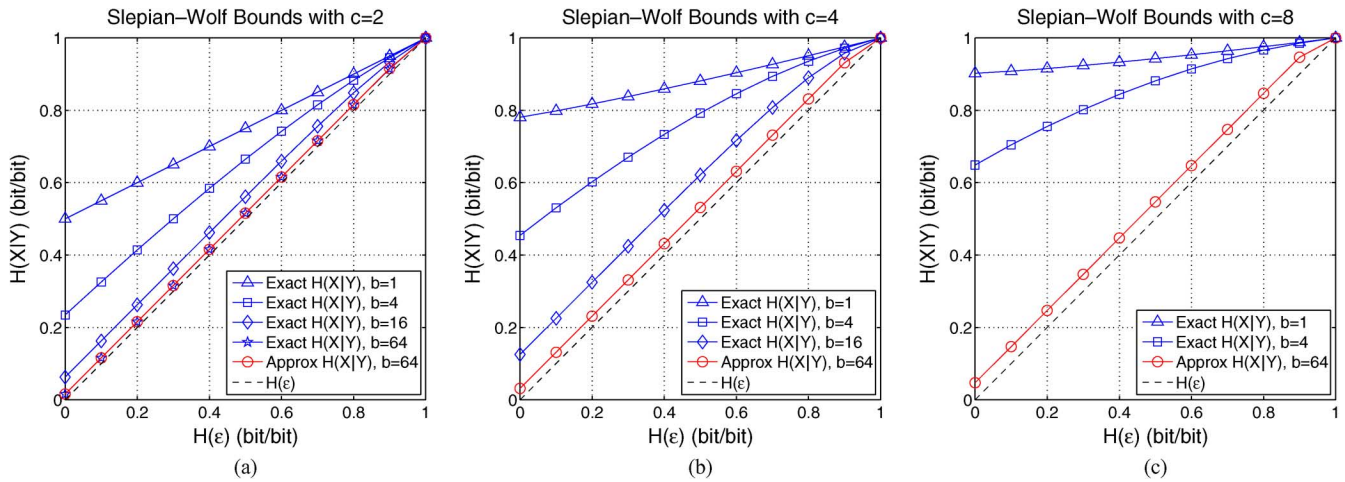


Fig. 3. Slepian–Wolf bounds for the block-candidate model shown as conditional entropy rates $\mathcal{H}(\mathbf{X}|\mathbf{Y})$ for the number of candidates c equal to (a) 2, (b) 4, and (c) 8. The exact form is shown for tractable values of block size b, and the approximation (under the typicality assumption) is shown for $b = 64$. Note that the exact and approximate forms agree for the combination $b = 64, c = 2$.

depends on the number of bits in each candidate equal to 1, calculation is tractable for small values of b and c.

One way to approximate $\mathcal{H}(\mathbf{Z}|\mathbf{X},\mathbf{Y})$ is to evaluate the sum in (6) for only the realizations $y|x = 0$ that are strongly typical, i.e., the statistically dependent candidate contains ϵb bits of value 1 and all other candidates contain $(1/2)b$ 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,

$$\mathcal{H}(\mathbf{Z}|\mathbf{X},\mathbf{Y}) \approx \frac{1}{b} H(Z_i|x=0, y \text{ strongly typical}) \tag{7}$$

$$= \frac{1}{b}\left(-p_{\text{dep}} \log_2 p_{\text{dep}} - (c-1) p_{\text{ind}} \log_2 p_{\text{ind}}\right) \tag{8}$$

where

$$p_{\text{dep}} = \frac{w_{\text{dep}}}{w_{\text{dep}} + (c-1)w_{\text{ind}}} \tag{9}$$

$$p_{\text{ind}} = \frac{w_{\text{ind}}}{w_{\text{dep}} + (c-1)w_{\text{ind}}}. \tag{10}$$

Here, $w_{\text{dep}} = (1-\epsilon)^{(1-\epsilon)b}\epsilon^{\epsilon b}$ and $w_{\text{ind}} = (1-\epsilon)^{(1/2)b}\epsilon^{(1/2)b}$ are likelihood weights of the statistically dependent and independent candidates, respectively, being identified as the statistically dependent candidate. 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 $\mathcal{H}(\mathbf{X}|\mathbf{Y})$ 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 when $\epsilon \ll 0.5$. Note that $\mathcal{H}(\mathbf{Z}|\mathbf{X},\mathbf{Y})$ is close to zero in this useful regime. For this reason alone, it is not surprising that the exact and approximate evaluations closely match in Fig. 3. We also explain that the Slepian–Wolf bound is decreasing in block size b and increasing in both number of candidates c and $H(\epsilon)$ by the dominance of the first two terms in the expression for $\mathcal{H}(\mathbf{X}|\mathbf{Y})$ in (3).

III. Adaptive Distributed Source Coder

A. Encoder

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

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

B. Decoder

The decoder recovers the source $\mathbf{X}$ from the doping bits and the syndrome bits so far received from the encoder in conjunction with the block-candidate side information $\mathbf{Y}$. We denote the received vectors of doping 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 successfully converges when the source bit estimates, when thresholded, are consistent with the vector of syndrome bits.

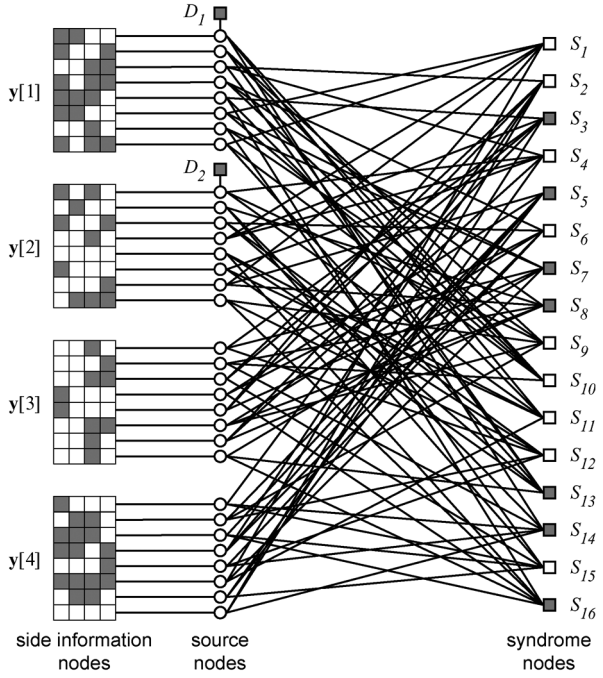


Fig. 4. Factor graph for adaptive distributed source decoder with parameters: code length $n = 32$, block size $b = 8$, number of candidates $c = 4$, $R_{\text{fixed}} = 1/16$, and $R_{\text{adaptive}} = 1/2$. The associated rate-adaptive LDPC code is regular with edge-perspective source and syndrome degree distributions $\lambda_t(\omega) = \omega^2$ and $\rho_t(\omega) = \omega^5$, respectively, where code counter $t = (n/k)R_{\text{adaptive}}$.

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 they are denoted by vectors $(p_1^{\text{in}}, p_2^{\text{in}}, \ldots, p_d^{\text{in}})$ and $(p_1^{\text{out}}, p_2^{\text{out}}, \ldots, p_d^{\text{out}})$, respectively, where d is the degree of the node. By the sum–product rule, each output message p_u^{out} is a function of all input messages except p_u^{in}, which is the input message on the same edge. Each source node additionally produces an estimate p^{est} that its bit is 0 based on all the input messages. We now detail the computation rules of the nodes shown in Fig. 5.

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, i.e., it is 0 with likelihood weight $\prod_{v=1}^{d} p_v^{\text{in}}$ or 1 with likelihood weight $\prod_{v=1}^{d}(1 - p_v^{\text{in}})$. Consequently

$$p^{\text{est}} = \frac{\prod_{v=1}^{d} p_v^{\text{in}}}{\prod_{v=1}^{d} p_v^{\text{in}} + \prod_{v=1}^{d} (1 - p_v^{\text{in}})}. \tag{11}$$

Ignoring the input message p_u^{in}, the weights are $\prod_{v \neq u} p_v^{\text{in}}$ and $\prod_{v \neq u}(1 - p_v^{\text{in}})$; thus

$$p_u^{\text{out}} = \frac{\prod_{v \neq u} p_v^{\text{in}}}{\prod_{v \neq u} p_v^{\text{in}} + \prod_{v \neq u} (1 - p_v^{\text{in}})}. \tag{12}$$

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; therefore, the source bit estimate and the output messages are independent of the input messages, i.e.,

$$p^{\text{est}} = p_u^{\text{out}} = 1 - D. \tag{13}$$

Syndrome Node: Fig. 5(c) shows a syndrome node of binary value S. Since the connected source bits have a modulo 2 sum equal to S, the output message p_u^{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

$$p_u^{\text{out}} = \frac{1}{2} + \frac{1 - 2S}{2} \prod_{v \neq u} \left(2p_v^{\text{in}} - 1\right). \tag{14}$$

Side Information Node: Fig. 5(d) shows a side information node of value $\mathbf{y}[i]$ 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^{out} depends on all input messages except p_u^{in}. However, since the source bits and the bits of the dependent candidate are related through a crossover probability ϵ, we define the noisy input probability of a source bit being 0 by

$$p_v^{\text{noisy-in}} = (1 - \epsilon)p_v^{\text{in}} + \epsilon\left(1 - p_v^{\text{in}}\right). \tag{15}$$

In computing p_u^{out}, the likelihood weight of the candidate $\mathbf{y}[i, j]$ being the statistically dependent candidate is equal to the product of the likelihood values of that candidate's $b - 1$ bits excluding $Y_{u,j}$

$$w_{u,j} = \prod_{v \neq u} \left(\mathbb{1}_{[Y_{v,j}=0]} p_v^{\text{noisy-in}} + \mathbb{1}_{[Y_{v,j}=1]}(1 - p_v^{\text{noisy-in}})\right) \tag{16}$$

where $\mathbb{1}_{[\cdot]}$ is the indicator function. We finally marginalize p_u^{out} as the normalized sum of weights for which $Y_{u,j}$ is 0, passed through crossover probability ϵ

$$p_u^{\text{clean-out}} = \frac{\sum_{j=1}^{c} \mathbb{1}_{[Y_{u,j}=0]} w_{u,j}}{\sum_{j=1}^{c} w_{u,j}} \tag{17}$$

$$p_u^{\text{out}} = (1 - \epsilon)p_u^{\text{clean-out}} + \epsilon\left(1 - p_u^{\text{clean-out}}\right). \tag{18}$$

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 n/b side information nodes because the rest of the factor graph is equivalent to an LDPC decoder of complexity $\mathcal{O}(n)$ [25] (assuming fixed degree distributions).

In each side information node, the complexity limiting computation is (16). This step computes bc likelihood weights, each of which is the product of $b-1$ terms; hence, the per-node complexity appears to be $\mathcal{O}(b^2 c)$. However, rewriting (16) as

$$w_{u,j} = \frac{\prod_v \left(\mathbb{1}_{[Y_{v,j}=0]} p_v^{\text{noisy-in}} + \mathbb{1}_{[Y_{v,j}=1]} \left(1 - p_v^{\text{noisy-in}}\right)\right)}{\mathbb{1}_{[Y_{u,j}=0]} p_u^{\text{noisy-in}} + \mathbb{1}_{[Y_{u,j}=1]} \left(1 - p_u^{\text{noisy-in}}\right)} \tag{19}$$

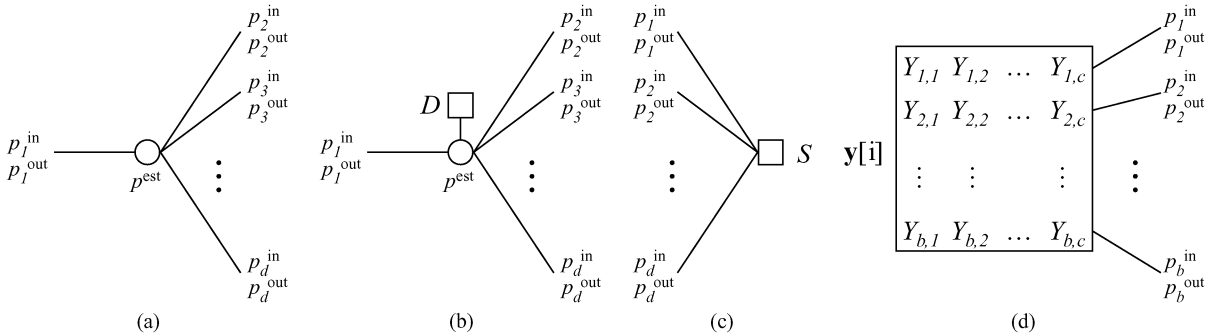


Fig. 5. Factor graph node combinations. Input and output messages and source bit estimate (if applicable) of (a) source node unattached to doping node, (b) source node attached to doping node, (c) syndrome node, and (d) side information node.

we see that the numerator need only be calculated once for each value of j. Therefore, the per-node complexity is $\mathcal{O}(bc)$. Since there are n/b such nodes, the overall decoding complexity is $\mathcal{O}(nc)$.

IV. DE Analysis

We use DE 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 DE 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 the statistically dependent candidate is in the first position $\mathbf{y}[i,1]$. This shuffling has no effect on the messages.

We then replace each side information candidate $\mathbf{y}[i,j]$ with the modulo 2 sum of itself and its corresponding source block $\mathbf{x}[i]$, and we set all the source, syndrome, 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_{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 $\prod_{v\neq u}(2p_v^{\text{in}} - 1)$ in (14). In contrast, side information nodes with edges removed must substitute the missing input messages with 1 in (15) (16) (17) (18).

Applying these three manipulations to the factor graph in Fig. 4 produces the simpler factor graph, equivalent with respect to convergence, as 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 $\mathbf{y}[i,1]$ are statistically dependent with respect to the source bits. In particular, the bits of $\mathbf{y}[i,1]$ are independently equal to 0 with probability $1-\epsilon$, whereas the bits of $\mathbf{y}[i,j\neq 1]$ are independently equiprobable. Note also the absence of combinations of source nodes linked to doping nodes.

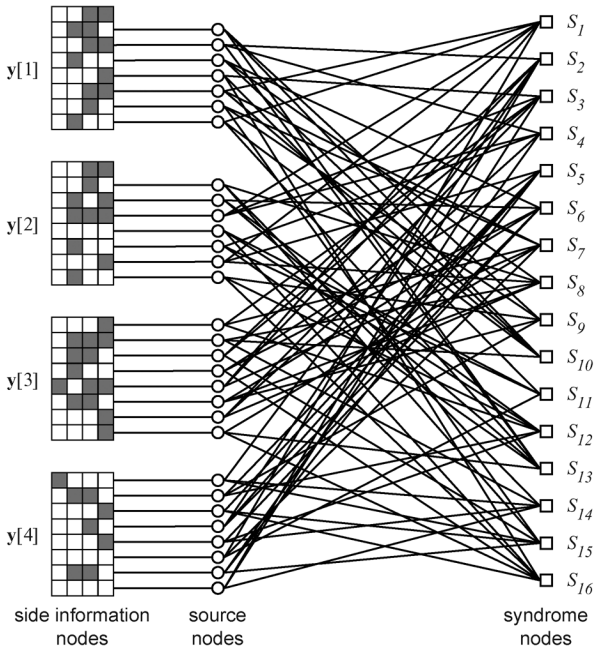


Fig. 6. Transformed factor graph equivalent to that in Fig. 4 in terms of convergence. After doping, the edge-perspective source and syndrome degree distributions of the associated rate-adaptive LDPC code are $\lambda_t^*(\omega) = \omega^2$ and $\rho_t^*(\omega) = (2/45)\omega^3 + (2/9)\omega^4 + (11/15)\omega^5$, respectively. The edge-perspective side information degree distribution is $\beta_t^*(\omega) = (7/15)\omega^6 + (8/15)\omega^7$.

B. Degree Distributions

DE runs, not on a factor graph itself, but using degree distributions of that factor graph. Degree distributions exist in two equivalent forms, namely, edge perspective and node perspective, and we represent both by polynomials in ω. In an edge-perspective degree distribution polynomial, the coefficient of ω^{d-1} 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 ω^d is the fraction of a certain type of node of degree d out of all nodes of that type.

In total, we consider 12 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 $\lambda_t(\omega)$, $\rho_t(\omega)$, and $\beta_t(\omega)$ in edge perspective and $L_t(\omega)$, $R_t(\omega)$, and $B_t(\omega)$ in node perspective, where the code counter $t = (n/k)R_{\text{adaptive}}$. Their expected counterparts in the factor graph after transformation (like the one in Fig. 6) are denoted as $\lambda_t^*(\omega)$, $\rho_t^*(\omega)$, and $\beta_t^*(\omega)$ in edge perspective and $L_t^*(\omega)$, $R_t^*(\omega)$, and $B_t^*(\omega)$ in node perspective.

For source degree distributions $\lambda_t(\omega)$, $L_t(\omega)$, $\lambda_t^*(\omega)$, and $L_t^*(\omega)$, we count the source–syndrome edges, but neither the source–side information nor source–doping edges. This way, $\lambda_t(\omega)$, $\rho_t(\omega)$, $L_t(\omega)$, and $R_t(\omega)$ are the degree distributions of the rate-adaptive LDPC codes, and therefore, we take them as given in the following derivations.

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

$$\lambda_t^*(\omega) = \lambda_t(\omega) \tag{20}$$

$$L_t^*(\omega) = L_t(\omega) = \frac{\int_0^\omega \lambda_t(\psi)d\psi}{\int_0^1 \lambda_t(\psi)d\psi} \tag{21}$$

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

Syndrome Degree Distributions: The factor graph transformation, by removing a fraction R_{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 independently retained with probability $1 - R_{\text{fixed}}$. Consequently, the chance that it has degree d^* after factor graph transformation is a binomial probability, denoted as $\text{binopmf}(d, d^*, R_{\text{fixed}}) = \binom{d}{d^*}(1 - R_{\text{fixed}})^{d^*}(R_{\text{fixed}})^{d-d^*}$. Therefore, if the node-perspective syndrome degree distribution before transformation is expressed as $R_t(\omega) = \sum_{d=1}^{d_{\max}} A_d\omega^d$, then, after transformation, the expected node-perspective syndrome degree distribution is given by

$$R_t^*(\omega) = \sum_{d=1}^{d_{\max}} \frac{A_d}{1-(R_{\text{fixed}})^d} \sum_{d^*=1}^{d} \text{binopmf}(d, d^*, R_{\text{fixed}})\omega^{d^*} \tag{22}$$

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

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

$$\beta_t(\omega) = \omega^{b-1} \tag{23}$$

$$B_t(\omega) = \omega^b. \tag{24}$$

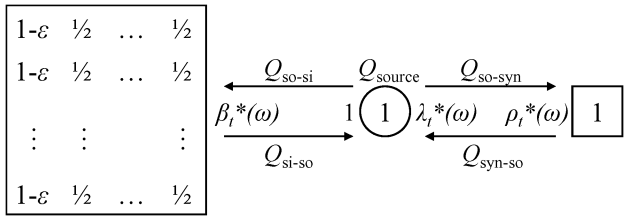


Fig. 7. DE for adaptive distributed source decoder. The three stochastic nodes are representatives of the side information, source, and syndrome nodes, respectively, of a transformed factor graph, like the one in Fig. 6. The quantities inside the nodes are the probabilities that the values at those positions are 0. Beside the nodes, written are the expected edge-perspective degree distributions of the transformed factor graph. The arrow labels $Q_{\text{so}-\text{si}}$, $Q_{\text{so}-\text{syn}}$, $Q_{\text{syn}-\text{so}}$, and $Q_{\text{si}-\text{so}}$ are the densities of messages passed in the transformed factor graph, and Q_{source} is the density of source bit estimates.

Since the doping pattern is deterministic and regular at rate R_{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

$$B_t^*(\omega) = (1 - A)\omega^{b^*} + A\omega^{b^*+1} \tag{25}$$

$$\beta_t^*(\omega) = \frac{(1 - A)b^*}{b^* + A}\omega^{b^*-1} + \frac{A(b^* + 1)}{b^* + A}\omega^{b^*} \tag{26}$$

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

C. DE

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 as $Q_{\text{so}-\text{si}}$, $Q_{\text{so}-\text{syn}}$, $Q_{\text{syn}-\text{so}}$, and $Q_{\text{si}-\text{so}}$, respectively. Another density Q_{source} captures the distribution of source bit estimates.

Fig. 7 depicts a schematic of the DE 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 toward 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 DE, 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, Q_{source} is evaluated. The sum–product algorithm is deemed to converge for the factor graph in question if and only if the density of source bit estimates Q_{source}, after thresholding, converges to the source bit value 0.

The rest of this section provides the stochastic update rules for the densities in a Monte Carlo simulation of DE. For the simulation, each of the densities Q_{source}, $Q_{\text{so-si}}$, $Q_{\text{so-syn}}$, $Q_{\text{syn-so}}$, and $Q_{\text{si-so}}$ is defined to be a set of samples q, each of which is a probability that its associated source bit is 0. At initialization, all samples q of all densities are set to 1/2.

Source Bit Estimate Density: To compute each sample q^{out} of Q_{source}, let a set Q^{in} consist of one sample drawn randomly from $Q_{\text{si-so}}$ and δ samples drawn randomly from $Q_{\text{syn-so}}$. Random degree δ is drawn equal to d with probability equal to the coefficient of ω^d in node-perspective $L_t^*(\omega)$ since there is one actual source bit estimate per node. Then, according to (11)

$$q^{\text{out}} = \frac{\prod_{q^{\text{in}} \in Q^{\text{in}}} q^{\text{in}}}{\prod_{q^{\text{in}} \in Q^{\text{in}}} q^{\text{in}} + \prod_{q^{\text{in}} \in Q^{\text{in}}} (1 - q^{\text{in}})}. \tag{27}$$

Source-to-Side-Information Message Density: To compute each updated sample q^{out} of $Q_{\text{so-si}}$, let a set Q^{in} consist of δ samples drawn randomly from $Q_{\text{syn-so}}$. Random degree δ is drawn equal to d with probability equal to the coefficient of ω^d in node-perspective $L_t^*(\omega)$ since there is one actual output message per node. Using (12), the updated formula is the same as (27).

Source-to-Syndrome Message Density: To compute each updated sample q^{out} of $Q_{\text{so-syn}}$, let a set Q^{in} consist of one sample drawn randomly from $Q_{\text{si-so}}$ and $\delta - 1$ samples drawn randomly from $Q_{\text{syn-so}}$. Random degree δ is drawn equal to d with probability equal to the coefficient of ω^{d-1} in edge-perspective $\lambda_t^*(\omega)$ since there is one actual output message per edge. Using (12), the updated formula is the same as (27).

Syndrome-to-Source Message Density: To compute each updated sample q^{out} of $Q_{\text{syn-so}}$, let a set Q^{in} consist of $\delta - 1$ samples drawn randomly from $Q_{\text{so-syn}}$. Random degree δ is drawn equal to d with probability equal to the coefficient of ω^{d-1} in edge-perspective $\rho_t^*(\omega)$ since there is one actual output message per edge. The syndrome value is deterministically 0. Then, according to (14)

$$q^{\text{out}} = \frac{1}{2} + \frac{1}{2} \prod_{q^{\text{in}} \in Q^{\text{in}}} (2q^{\text{in}} - 1). \tag{28}$$

Side-Information-to-Source Message Density: To compute each updated sample q^{out} of $Q_{\text{si-so}}$, let a set $\{q_v^{\text{in}}\}_{v=1}^{b-1}$ consist of $\delta - 1$ samples drawn randomly from $Q_{\text{so-si}}$ and $b - \delta$ samples equal to 1. Random degree δ is drawn equal to d with probability equal to the coefficient of ω^{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^{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 toward 0, with probability $1 - \epsilon$ if $j = 1$ or probability 1/2 if $j \neq 1$.

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

$$q_v^{\text{noisy-in}} = (1 - \epsilon) q_v^{\text{in}} + \epsilon \left(1 - q_v^{\text{in}}\right) \tag{29}$$

$$w_j = \prod_{v=1}^{b-1} \left(\mathbb{1}_{[Y_{v,j}=0]} q_v^{\text{noisy-in}} + \mathbb{1}_{[Y_{v,j}=1]} \left(1 - q_v^{\text{noisy-in}}\right) \right) \tag{30}$$

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

$$q^{\text{out}} = (1 - \epsilon) q^{\text{clean-out}} + \epsilon (1 - q^{\text{clean-out}}). \tag{32}$$

V. Experimental Results

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

In these experiments, the adaptive distributed source codec 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_{adaptive} is in $\{1/128, 2/128, \ldots, 1\}$. For convenience, we allow R_{fixed} to take values in $\{0, 1/128, 2/128, \ldots, 1/8\}$. Our DE 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 $\mathbf{X}$ and the statistically dependent candidates of $\mathbf{Y}$. We first plot the coding rates achieved by the adaptive distributed source codec (averaged over 100 trials) and modeled by DE, when doping rate $R_{\text{fixed}} = 0$. The performance is far from the Slepian–Wolf bound $\mathcal{H}(\mathbf{X}|\mathbf{Y})$ since the adaptive decoder is not initialized with sufficient reliable information about $\mathbf{X}$. Nevertheless, DE models the poor empirical performance reasonably accurately.

Increasing the doping rate R_{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_{fixed} in $\{0, 1/128, 2/128, \ldots, 1/8\}$ and let DE determine which R_{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 it is modeled by DE even better than when $R_{\text{fixed}} = 0$.

Notice that the optimal doping rates appear to slowly decrease as $H(\epsilon)$ increases from 0 to 0.6 and sharply increase as $H(\epsilon)$ increases from 0.6 to 0.8. (At $H(\epsilon) = 0.9$, coding is at full rate of 1 bit/bit; hence, no doping is necessary.) We explain these trends by considering the value of additional doping bits. At low

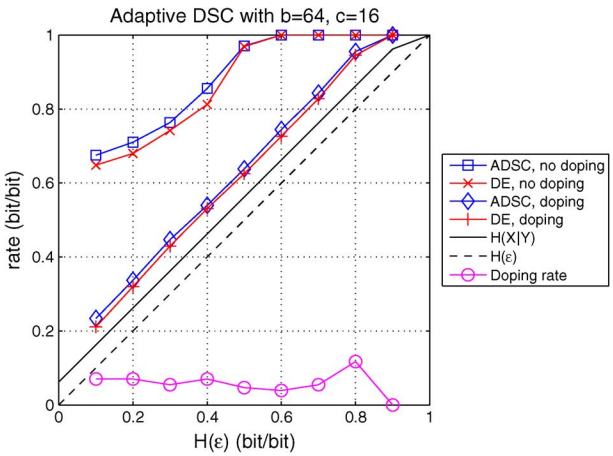


Fig. 8. Performance of ADSC with doping rate $R_{\text{fixed}} = 0$ and optimally set by DE.

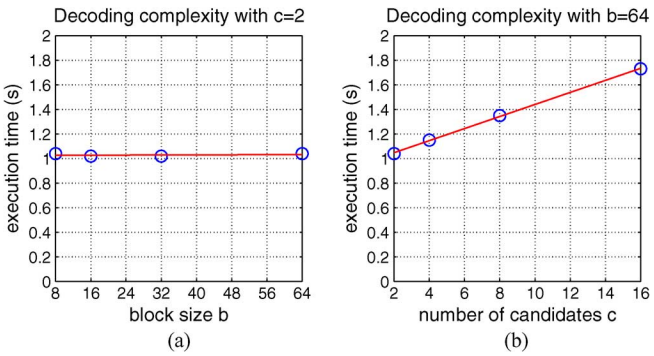


Fig. 9. Execution times for 100 iterations of the decoder's sum–product algorithm. Empirical points are circles in blue, and fitted trend lines are in red.

$H(\epsilon)$, 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 Core 2 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_{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 DE to narrow down the possible range.

Fig. 9(a) fixes $c = 2$ and shows that complexity is constant in b, and Fig. 9(b) fixes $b = 64$ and shows that complexity is linear in c. Both of these results are consistent with the asymptotic decoding complexity of $\mathcal{O}(nc)$ derived in Section III.

VI. Application Example: Video Quality Monitoring and Channel Tracking

We now apply ADSC to a simple example of end-to-end video quality monitoring and show how DE is used to design

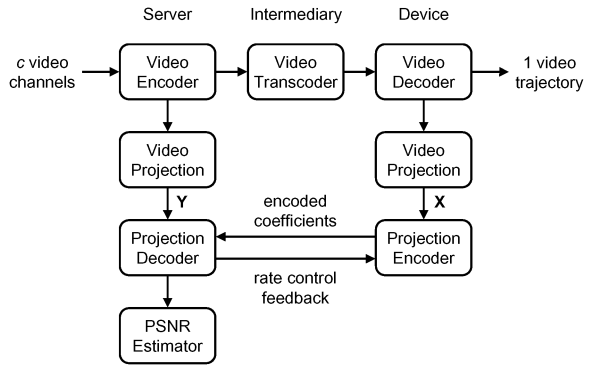


Fig. 10. Video transmission with quality monitoring and channel tracking.

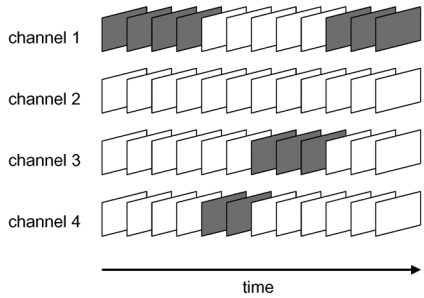


Fig. 11. Trajectory through four channels of video.

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 the 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 $\mathbf{X}$ of the video trajectory and sends them back to the server. A feedback channel from the server to the device is available for rate control. The server decodes and compares the coefficients with projection coefficients $\mathbf{Y}$ of the c channels of the 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 ADSC reduces the quality monitoring bit rate by about 75% compared with fixed-length coding (FLC).

The video projection for both sets of coefficients $\mathbf{X}$ and $\mathbf{Y}$ is specified in ITU-T Recommendation J.240 [26], [27]. The projection partitions the luminance channel of the video into tiles, sizes of 16×16 or 8×8 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 2-D Walsh–Hadamard transform (WHT), multiplied by another maximum length sequence, and inversely transformed using the 2-D inverse WHT (IWHT). 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 $\mathbf{X}$ and $\mathbf{Y}$. Suppose that $\bar{X}$ and $\bar{Y}$, each of length n, denote the subvectors of coefficients

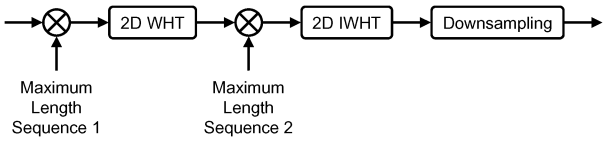


Fig. 12. Dimension reduction projection of ITU-T Recommendation J.240.

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

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

$$\mathrm{PSNR}_{\mathrm{J.240}} = 10\log_{10}\frac{255^2}{\mathrm{MSE}_{\mathrm{J.240}}}. \tag{34}$$

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

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

$$\mathrm{PSNR}_{\mathrm{ML}} = 10\log_{10}\frac{255^2}{\mathrm{MSE}_{\mathrm{ML}}}. \tag{36}$$

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

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

In the experiment, we use $c = 8$ channels of video (*Foreman, Carphone, Mobile, Mother and Daughter, Table, News, Coastguard, and Container*) at resolution of 176 × 144 and frame rate of 30 Hz. The first 256 frames of each sequence are encoded at the server using the H.264/AVC [31] baseline profile in interprediction (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, i.e., 16 × 16 or 8 × 8, respectively. Each coefficient is quantized to 1 bit so that we can model the codec using DE exactly as described in Section IV. The multilevel extensions of both the DE technique and these video quality monitoring results are presented in [33].

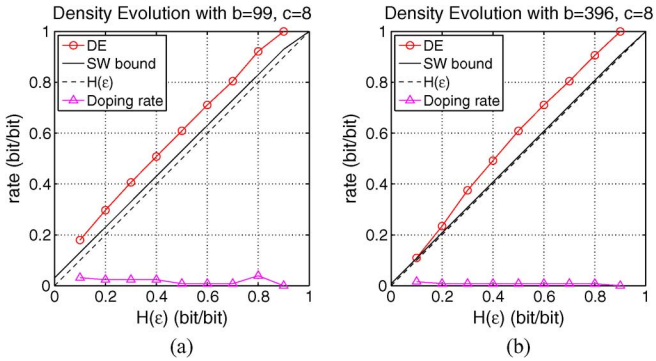


Fig. 13. DE analysis of ADSC under the settings (a) $b = 99$, $c = 8$ and (b) $b = 396$, $c = 8$.

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 $0 \le H(\epsilon) \le 1$. Note that this DE analysis makes use of neither the test video sequences nor their statistics. We choose the values $R_{\mathrm{fixed}} = 4/132$ for $b = 99$ and $R_{\mathrm{fixed}} = 2/132$ for $b = 396$ because these are the maximum optimal doping rates for $H(\epsilon) \le 0.2$, which covers about 90% of the empirical conditional entropy rates. The doping pattern is regular with respect to the side information nodes, i.e., the sequence is $1, b+1, 2b+1, \ldots, n-b+1, 2, b+2, 2b+2, \ldots, n-b+2, \ldots$. 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 eight frames of coefficients at a time using rate-adaptive LDPC codes with regular source degree distribution $\lambda_{\mathrm{reg}}(\omega) = \omega^2$, length $n = 8b$ bits, and data increment size $k = 8b/132$ bits. Consequently, $R_{\mathrm{adaptive}} \in \{1/132, 2/132, \ldots, 1\}$.

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 a group of picture 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 the projection encoder to the decoder in kilobits per second, assuming that the video has a 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 ML technique in [29] reduces the estimation error to useful values of 1.1 and 0.7 dB for $b = 99$ and 396, respectively. ADSC then reduces the quality monitoring bit rate by 73% and 76%, respectively, as compared with FLC.

VII. Conclusion

ADSC 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 have defined a block-candidate model for side information and compute its Slepian–Wolf bound. The iterative decoding algorithm

TABLE I
VIDEO QUALITY MONITORING PERFORMANCE FOR $b = 99$, $c = 8$

Settings		Performance	
Estimation	Coding	Mean Absolute PSNR Estimation Error (dB)	Quality Monitoring Bit Rate (kbit/s)
J.240	FLC	22.6	2.97
ML	FLC	1.1	2.97
ML	ADSC	1.1	0.81

TABLE II
VIDEO QUALITY MONITORING PERFORMANCE FOR $b = 396$, $c = 8$

Settings		Performance	
Estimation	Coding	Mean Absolute PSNR Estimation Error (dB)	Quality Monitoring Bit Rate (kbit/s)
J.240	FLC	30.0	11.88
ML	FLC	0.7	11.88
ML	ADSC	0.7	2.81

is described entirely in terms of the sum–product algorithm on a factor graph. This formulation permits the analysis of coding performance via DE. 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 a DE analysis. With ADSC, the reduced-reference bit rate is reduced by about 75% compared with FLC.

ACKNOWLEDGMENT

The authors would like to thank A. Montanari for valuable discussions on density evolution.

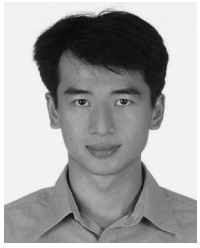
REFERENCES

[1] B. Girod, A. M. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proc. IEEE*, vol. 93, no. 1, pp. 71–83, Jan. 2005.
[2] R. Puri, A. Majumdar, and K. Ramchandran, "PRISM: A video coding paradigm with motion estimation at the decoder," *IEEE Trans. Image Process.*, vol. 16, no. 10, pp. 2436–2448, Oct. 2007.
[3] X. Zhu, A. M. Aaron, and B. Girod, "Distributed compression for large camera arrays," in *Proc. IEEE Workshop Stat. Signal Process.*, St. Louis, MO, 2003, pp. 30–33.
[4] G. Toffetti, M. Tagliasacchi, M. Marcon, A. Sarti, S. Tubaro, and K. Ramchandran, "Image compression in a multi-camera system based on a distributed source coding approach," in *Proc. Euro. Signal Process. Conf.*, Antalya, Turkey, 2005.
[5] K. Chono, Y.-C. Lin, D. P. Varodayan, and B. Girod, "Reduced-reference image quality estimation using distributed source coding," in *Proc. IEEE Int. Conf. Multimedia Expo.*, Hannover, Germany, Jun. 2008, pp. 609–612.
[6] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 4, pp. 471–480, Jul. 1973.
[7] J. Bajcsy and P. Mitran, "Coding for the Slepian–Wolf problem with turbo codes," in *Proc. IEEE Global Commun. Conf.*, San Antonio, TX, 2001, pp. 1400–1404.
[8] J. Garcıacute;a-Frıacute;as, "Compression of correlated binary sources using turbo codes," *IEEE Commun. Lett.*, vol. 5, no. 10, pp. 417–419, Oct. 2001.
[9] A. M. Aaron and B. Girod, "Compression with side information using turbo codes," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, 2002, pp. 252–261.
[10] A. Liveris, Z. Xiong, and C. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," *IEEE Commun. Lett.*, vol. 6, no. 10, pp. 440–442, Oct. 2002.
[11] D. P. Varodayan, A. M. Aaron, and B. Girod, "Rate-adaptive distributed source coding using low-density parity-check codes," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, 2005, pp. 1203–1207.
[12] D. P. Varodayan, A. M. Aaron, and B. Girod, "Rate-adaptive codes for distributed source coding," *EURASIP Signal Process. J.*, vol. 86, no. 11, pp. 3123–3130, Nov. 2006.
[13] J. Garcıacute;a-Frıacute;as and W. Zhong, "LDPC codes for asymmetric compression of multi-terminal sources with hidden Markov correlation," in *Proc. CTA Commun. Netw. Symp.*, College Park, MD, 2003.
[14] Y. Zhao and J. Garcıacute;a-Frıacute;as, "Turbo codes for symmetric compression of correlated binary sources with hidden Markov correlation," in *Proc. CTA Commun. Netw. Symp.*, College Park, MD, 2003.
[15] J. Garcıacute;a-Frıacute;as and W. Zhong, "LDPC codes for compression of multi-terminal sources with hidden Markov correlation," *IEEE Commun. Lett.*, vol. 7, no. 3, pp. 115–117, Mar. 2003.
[16] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, vol. 41, no. 1, pp. 164–171, Feb. 1970.
[17] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc., Ser. B*, vol. 39, no. 1, pp. 1–38, 1977.
[18] D. P. Varodayan, A. Mavlankar, M. Flierl, and B. Girod, "Distributed coding of random dot stereograms with unsupervised learning of disparity," in *Proc. IEEE Int. Workshop Multimedia Signal Process.*, Victoria, BC, Canada, 2006, pp. 5–8.
[19] D. P. Varodayan, A. Mavlankar, M. Flierl, and B. Girod, "Distributed grayscale stereo image coding with unsupervised learning of disparity," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, 2007, pp. 143–152.
[20] D. P. Varodayan, Y.-C. Lin, A. Mavlankar, M. Flierl, and B. Girod, "Wyner–Ziv coding of stereo images with unsupervised learning of disparity," in *Proc. Picture Coding Symp.*, Lisbon, Portugal, 2007.
[21] D. P. Varodayan, D. M. Chen, M. Flierl, and B. Girod, "Wyner–Ziv coding of video with unsupervised motion vector learning," *EURASIP Signal Process. Image Commun. J*, vol. 23, no. 5, pp. 369–378, Jun. 2008.
[22] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum–product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
[23] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.
[24] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ: Wiley, 1991.
[25] T. J. Richardson and R. L. Urbanke, *Modern Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2008.
[26] ITU-T Recommendation J.240: Framework for Remote Monitoring of Transmitted Picture Signal-to-Noise Ratio Using Spread-Spectrum and Orthogonal Transform Jun. 2004.
[27] R. Kawada, O. Sugimoto, A. Koike, M. Wada, and S. Matsumoto, "Highly precise estimation scheme for remote video PSNR using spread spectrum and extraction of orthogonal transform coefficients," *Electron. Commun. Jpn. (Part I)*, vol. 89, no. 6, pp. 51–62, Jun. 2006.
[28] S. W. Golomb, *Shift Register Sequences*. San Francisco, CA: Holden-Day, 1967.
[29] Y.-C. Lin, D. P. Varodayan, and B. Girod, "Video quality monitoring for mobile multicast peers using distributed source coding," in *Proc. Int. Mobile Multimedia Commun. Conf.*, London, U.K., Sep. 2009, p. 31.

[30] S. M. LoPresto, K. Ramchandran, and M. T. Orchard, "Image coding based on mixture modeling of wavelet coefficients and a fast estimation–quantization framework," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, 1997, pp. 221–230.
[31] T. Wiegand, G. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[32] "ITU-T Recommendation T.81: Digital Compression and Coding of Continuous-Tone Still Images," Sep. 1992.
[33] D. P. Varodayan, "Adaptive distributed source coding," Ph.D. dissertation, Dept. Electr. Eng., Stanford Univ., Stanford, CA, 2010.

David Varodayan (M'11) received the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 2005 and 2010, respectively.

Dr. Varodayan was a recipient of the European Association for Signal Processing (EURASIP) Signal Processing Journal's Most Cited Paper Award in 2009 and the Best Student Paper Award on two occasions: IEEE Workshop on Multimedia Signal Processing in 2006 and European Signal Processing Conference in 2007.



Yao-Chung Lin (M'09) received the B.S. degree in computer science and information engineering and M.S. degree in electrical engineering from National Chiao Tung University, Hsinchu, Taiwan, and the Ph.D. degree from Stanford University, Stanford, CA, in 2010.

His research interests include distributed source coding applications, multimedia systems, and video processing and compression.

Bernd Girod (F'98) received the Engineering Doctorate from the University of Hannover, Hannover, Germany, and the M.S. degree from Georgia Institute of Technology, Atlanta.

He has been a Professor of electrical engineering and (by courtesy) computer science in the Information Systems Laboratory, Stanford University, Stanford, CA, since 1999. Previously, he was a Professor in the Department of Electrical, Electronics, and Information Technology, University of Erlangen-Nuremberg. He has published more than 450 conference and journal papers, as well as 5 books. His current research interests are in the areas of video compression, networked media systems, and image-based retrieval. As an entrepreneur, he has been involved with several startup ventures, among them are Polycom, Vivo Software, 8 $\times$8, and RealNetworks.

Dr. Girod was the recipient of the European Association for Signal Processing (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, and the EURASIP Technical Achievement Award in 2004. He is a Fellow of EURASIP and a member of the German National Academy of Sciences (Leopoldina).