On Some New Approaches to Practical Slepian-Wolf Compression Inspired by Channel Coding

Todd P. Coleman [*], Anna H. Lee, Muriel Médard [†], Michelle Effros [‡]

{colemant,annalee,medard}@mit.edu; effros@caltech.edu

Massachusetts Institute of Technology; California Institute of Technology

Laboratory for Information and Decision Systems; Data Compression Laboratory

Cambridge, MA 02139; Pasadena, CA 91125

Abstract

We introduce three new innovations for compression using LDPCs for the Slepian-Wolf problem. The first is a general iterative Slepian-Wolf decoding algorithm that incorporates the graphical structure of all the encoders and operates in a 'turbo-like' fashion. The second innovation introduces source-splitting to enable low-complexity pipelined implementations of Slepian-Wolf decoding at rates besides corner points of the Slepian-Wolf region. This innovation can also be applied to single-source block coding for reduced decoder complexity. The third approach is a linear programming relaxation to maximum-likelihood sequence decoding that exhibits the ML-certificate property. This can be used for decoding a single binary block-compressed source as well as decoding at vertex points for the binary Slepian-Wolf problem. All three of these innovations were motivated by recent analogous results in the channel coding domain.

1 Introduction

The problem of distributed lossless compression of correlated sources is relevant for numerous applications, such as sensor networks. For a set of m memoryless sources $(U^1, \ldots, U^m)$ with joint probability distribution $P\left(u^1, \ldots, u^m\right)$, the achievable rate region is given [1] by

$$\sum_{i \in S} R_i > H\left(U(S)|U(S^c)\right) \ \forall \ S \subseteq \{1, 2, \ldots, m\}$$

where $U(S) = \{U^j, j \in S\}$. Vertices are the rate tuples $(R_1, \ldots, R_m)$ that are obtained by expanding $H(U^1, \ldots, , U^m)$ into m terms by successive applications of the chain rule and assigning to each rate the unique term in the expansion. Each unique vertex corresponds to a rate-tuple that is single-user decodable given side information of the previously decoded users. The vertices in the two-user setting correspond to the rate pairs $(R_1, R_2) = \left(H(U^1), H(U^2|U^1)\right)$ and $(R_1, R_2) = \left(H(U^1|U^2), H(U^2)\right)$.

[*]supported by an NSF Graduate Research Fellowship

[†]supported by the HP-MIT Wireless Networking Alliance

[‡]supported by NSF Grant No. CCR-0220039 and Caltech's Lee Center for Advanced Networking

In [2], Csiszár showed that random linear block codes asymptotically achieve the optimal performance for the Slepian-Wolf problem under a universal minimum-entropy decoding mechanism. Recent work has addressed the Slepian-Wolf problem using linear codes and iterative decoding. Low-density parity-check code (LDPC) formulations [3, 4, 5] and turbo code formulations [6, 7, 8, 9] have been shown to provide excellent performance with low-complexity iterative decoding for the binary problem at vertex rate points.

In this paper, we consider three issues that we believe make the Slepian-Wolf source coding problem more amenable to being used in practice. Section 2 discusses a general joint iterative decoding algorithm for LDPCs that has a 'turbo-style' interpretation. Section 3 discusses source-splitting, which can be used to allow the decoder to operate in a pipelined fashion with reduced complexity. Section 3.1 illustrates how source-splitting yields rates besides vertices in the Slepian-Wolf region that can be decoded successively and shows a simple MAP decoder exists based upon the outputs of iterative decoders. Section 3.2 illustrates how source-splitting can be applied to single-source block coding for reduced decoder complexity. Again, it is shown that a simple MAP decoder exists based upon the outputs of iterative decoders. Finally, section 4 discusses a linear programming relaxation to the maximum-likelihood (ML) sequence decoding problem for source coding. It is shown that this relaxation has the *ML-certificate* property: if an integral solution is found, it is the ML solution. Formulations for both the binary single-source block coding case and the binary Slepian-Wolf vertex case are constructed. All of these results stem from similar results in the channel coding literature.

2 Iterative Decoding of LDPC codes for Slepian-Wolf

We discuss in this section a general algorithm for the use of LDPCs as syndrome-formers [10] for the two-user binary Slepian-Wolf problem. We note that the algorithm generalizes to an arbitrary number n of users using LDPCs over $GF(2^m)$. The correlated memoryless sources U and V are jointly distributed according to $P_{U,V}(u,v)$ and are independently encoded using the same block length n. User $\underline{u}$ ($\underline{v}$) uses a k_u by n (k_v by n) matrix H_u (H_v) to construct the length k_u (k_v) syndrome $\underline{s}_u$ ($\underline{s}_v$):

$$\underline{s}_u = H_u \underline{u} \quad \underline{s}_v = H_v \underline{v}$$

The resultant rates are $\frac{k_u}{n}$ and $\frac{k_v}{n}$. The decoder's job is to infer u and v from the syndromes.

2.1 Graphical Realizations

Graphical representations denote the dependencies between codewords based upon the constraints they must satisfy. For a linear code, each local constraint is a smaller linear code. The leftmost side of figure 1 illustrates a normal graph representation [11], where bits are associated with edges and constraint codes are associated with nodes. A node with a '+' sign and degree d is a $(d, d-1, 2)$ single parity check code that imposes the constraint that the bits lying on the d edges adjacent to that node must sum (modulo 2) to 0. A node of degree d' with an '=' sign is a $(d', 1, 2)$ repetition code and imposes the constraint that the bits lying on the d' adjacent edges must be equal. There are n nodes with a '=' label for user $u(v)$, each of which corresponds to a single repetition code and has a connection to one of the input variable symbols. There are $k_u(k_v)$ nodes with a '+' label, each of which corresponds to a single parity-check code and has a connection to one

of the output syndrome symbols for user $u(v)$. The ith parity-check node is connected to the jth repetition node for user $u(v)$ if and only if the i,j entry of $H_u(H_v)$ is 1, i.e. if the jth variable node is involved in the computation of the ith syndrome symbol. The set of all valid (u, s_u) input-output sequences is the set of (u, s_u) pairs that satisfy all local constraints. The same follows for (v, s_v) input-output sequences.

2.2 Iterative Decoding Using the Joint Distribution

We consider a belief propagation-style iterative decoding approach, using a method analogous to the multiple access iterative decoding approach introduced in [12]. For the code described by H_u (H_v), we assume all parity check and repetition nodes are labeled and define a bipartite graph $G_u = (V_u, E_u)$ $(G_v = (V_v, E_v))$. For any node $i_u \in V_u$ $(i_v \in V_v)$, we define $N(i_u)$ $(N(i_v))$ to be its set of neighbors: $N(i_u) = \{j_u \in V_u \; s.t. \; (i_u, j_u) \in E_u\}$ $(N(i_v) = \{j_v \in V_v \; s.t. \; (i_v, j_v) \in E_v\})$.

Check Nodes:

At a check node of G_u, the update rule is the same as in a single-user LDPC code. The bit value of any edge is equal to the (modulo-2) sum of the remaining bits adjacent to a check node. We assume the graph has no cycles so that we have a sum of independent random variables [11]. Thus the outgoing message along any edge is the convolution of the messages along all other incoming edges. The Fourier transforms of the distributions are multiplied and the outgoing message along any edge $(i_u, j_u) \in E_u$ for a parity check node i_u is given by

$$\hat{P}_{i_u \to j_u}(u) = \prod_{k_u \in N(i_u) \setminus \{j_u\}} \hat{P}_{k_u \to i_u}(u) \tag{1}$$

where $\hat{P}$ is the Fourier transform of the distribution P. An analogous equation holds for parity check nodes in G_v.

Variable Nodes:

At a variable node, each incoming message along the edges of G_u is assumed to be independent of the estimate of u. Similarly, each incoming message along the edges of G_v is an independent estimate of v. Moreover, for sufficiently distinct graphs, with high probability the message coming in from both the graphs are independent of each other. In addition to these messages, we also have the joint distribution between U and V. So for a variable node $j_u \in V_u$, there is a corresponding $j_v \in V_v$ which represents the same time index. This is illustrated in the lefthand side of figure 1 with ovals around the corresponding nodes. By the usual belief propagation rule, we get the outgoing joint message along an edge (j_u, i_u) of the graph:

$$P_{j_u \to i_u}(u, v) \propto P_{U,V}(u, v) \prod_{k_u \in N(j_u) \setminus \{i_u\}} P_{k_u \to j_u}(u) \prod_{k_v \in N(j_v)} P_{k_v \to j_v}(v). \tag{2}$$

Thus, the joint message passed along an edge of G_u depends on the joint distribution, *all* the incoming messages along edges adjacent to variable node j_v, and all the incoming messages along edges adjacent to j_u (except the edge along which the outgoing message is passed). The next step is to marginalize this joint distribution along G_u since its messages convey information about only u:

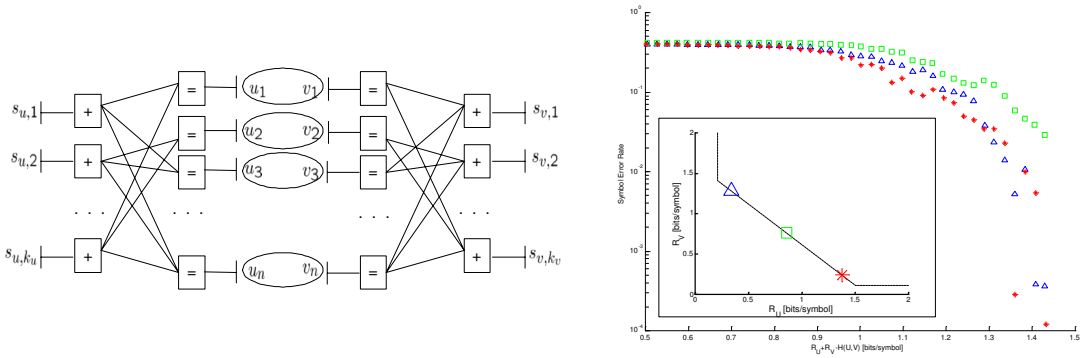

Figure 1: (L) Normal Joint Decoding Graph; (R) SER for the Joint Decoding

$$P_{j_u \to i_u}(u) \propto \prod_{k_u \in N(j_u)\setminus\{i_u\}} P_{k_u \to j_u}(u) \sum_{v=0}^{1} \left(P_{U,V}(u,v) \prod_{k_v \in N(j_v)} P_{k_v \to j_v}(v) \right) \quad (3)$$

$$= P'_{U_{\text{joint}}}(u) \prod_{k_u \in N(j_u)\setminus\{i_u\}} P_{k_u \to j_u}(u) \quad (4)$$

$$\text{where } P'_{U_{\text{joint}}}(u) \propto \sum_{v=0}^{1} P_{U,V}(u,v) \prod_{k_v \in N(j_v)} P_{k_v \to j_v}(v) \quad (5)$$

is the *updated joint information* and has the property that it does not depend on the edge along which the message is to be passed. In this way, the updated joint information is similar to channel information in a single user LDPC channel code. By exchanging the roles of U and V, we obtain analogous results for the variable update equations for G_v.

We iteratively recalculate these values according to some schedule, until a criterion for stopping is met. For example, the algorithm can be stopped when all parity checks are satisfied or after a fixed number of iterations. Thus the decoding algorithm for the Slepian-Wolf problem is the same as using two single-user LDPC decoders with each decoder *updating* the effective prior distribution for the *other* code at each iteration. In a sense, the two single-user LDPC decoders cooperate in 'turbo-like' fashion to work on the Slepian-Wolf problem.

The rightmost plot of figure 1 shows the achievability of non-corner points using the proposed joint decoding process. Regular LDPCs over $GF(4)$ with blocklength $n = 1000$ were used to code two correlated memoryless sources U, V where $|U| = |V| = 4$. The symbol error rate (SER) plots are shown for three non-vertex rate pairs, as a function of the sum rate difference from the entropy boundary. The legend shows the location of these rate pairs on the boundary of the Slepian-Wolf region. We note that this approach appears to suffer as rate tuples deviate far away from vertices; this is in agreement with analogous observations for the multiple access problem [12].

3 Source-Splitting

Let us now consider taking each symbol of a discrete memoryless source U of cardinality Q and splitting it into a collection of random variables of smaller cardinality. We say that $U_i \leftrightarrow (U_i^1, \ldots, U_i^k)$ if there is a bijection between the random variables U_i and $(U_i^1, \ldots, U_i^k)$.

Without loss of generality, we may assume that U_i takes on values in $\{0, 1, \ldots, Q-1\}$. Examples of how we may construct splits go as follows:

$$U_i \in \{0,1,\ldots,Q-1\} \quad \mapsto \quad \left(\begin{array}{l} U_i^1 = \min(U_i, T) \\ U_i^2 = \max(U_i, T) - T \end{array} \right) \mapsto U_i = U_i^1 + U_i^2 \tag{6}$$

$$U_i \in \{0,1,\ldots,Q-1\} \quad \mapsto \quad \left(\begin{array}{l} U_i^1 = 1_{\{U_i=1\}} \\ U_i^2 = 1_{\{U_i=2\}} \\ \vdots \\ U_i^{Q-1} = 1_{\{U_i=Q-1\}} \end{array} \right) \mapsto U_i = \sum_{k=1}^{Q-1} k U_i^k \tag{7}$$

where in (6), $T \in \{0, \ldots, Q-1\}$ and in (7), $1_{\{A\}} = 1$ if event A occurs and 0 otherwise.

We note that definition (6) gives many possible splits, especially when we consider re-ordering the alphabet. For nontrivial splits ($T \in \{1, \ldots, Q-2\}$), $U_i^1 \in \{0, \ldots, T\}$ and $U_i^2 \in \{0, \ldots, Q-1-T\}$ and there are $\binom{Q}{T}$ distinct ways to map the Q symbols to the splitting sets in (6). This provides a total of

$$\sum_{i=1}^{Q-2} \binom{Q}{i} = 2^Q - \binom{Q}{0} - \binom{Q}{Q-1} - \binom{Q}{Q} = 2^Q - Q - 2$$

distinct ways to perform the splitting mechanism and form the bijection $U_i \leftrightarrow (U_i^1, U_i^2)$.

The definition in (7) might, for example, be used to perform fixed-to-fixed block coding of a single non-binary source by transforming it to a set of binary sources. What makes this attractive is not the encoding, but the benefits in decoding, which is simplified, as we shall see. Each permutation of the indices of U yields new splits for the case of (7) and thus there are $Q!$ splits.

We call these operations *source-splitting*, in analogy with the rate-splitting ideas for multiple access channels [13]. We note that there are other possible ways to perform source-splitting, such as simple binary representation. As we shall see in section 3.1, the two examples we provide have nice implications that lead to simple MAP decoders.

We note that an approach for source-splitting was discussed in [14], but it made the assumption that i.i.d binary sources of common randomness were shared between the decoder and each encoder that performed splitting. Furthermore, it assumed that the splitting operation's output variables had cardinalities strictly larger than the cardinality of the original variable. Our approach gets around both of those issues.

3.1 Source-Splitting for Slepian-Wolf Coding

We note that, if we have two sources U and V, then we can split U to form (U^1, U^2) as shown in (6). At this point, we have three sources, all of which can be encoded separately using linear codes, as displayed in figure 2. We note that because $U \leftrightarrow (U^1, U^2)$, we have that $H(U) = H(U^1, U^2)$ and furthermore, $H(U, V) = H(U^1, U^2, V)$. As a consequence, the rate triple $(R_{U^1}, R_V, R_{U^2}) = \big(H(U^1), H(V|U^1), H(U^2|V, U^1)\big)$ where $R_U = R_{U^1} + R_{U^2}$ is not only achievable, but is also single-user decodable. What this means is that the decoding operation defined in section 2.2 can be applied here in a successive fashion using a schedule where first messages are passed only along one graph, then passed to the next given by (4), and so forth. The order in which they perform this successive decoding corresponds to the order in which the chain rule is expanded to arrive at the single-user decodable rate-triple. This scheme yields a pipelined approach that in real time operates at the rate of a lower-alphabet decoder (although decoders with side information need to be reconfigurable). Figure 2 illustrates the encoding and decoding process. The MAP

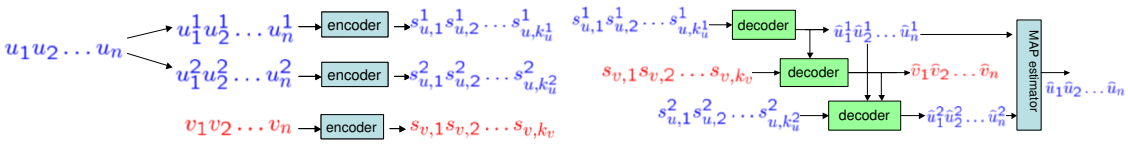

Figure 2: Source Splitting and Decoding for a Two-Source Slepian-Wolf Problem

estimator for symbol i of $\underline{U}$ is given by

$$\hat{U}_i = \arg \max_{u \in \{0,1,\ldots Q-1\}} P\left(U_i = u | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right)$$

We note that the sum-product algorithm for iterative decoding produces approximate a posteriori probabilities (APPs) for U_i^1, U_i^2 and V_i:

$$\begin{aligned} P\left(U_i^1 = u | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) &\triangleq \text{app}_i^{U^1}(u), \\ P\left(U_i^2 = u | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) &\triangleq \text{app}_i^{U^2}(u), \\ P\left(V_i = u | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) &\triangleq \text{app}_i^{V}(u). \end{aligned}$$

We wish to use these probabilities to form an efficient MAP decoder for U_i. By noting that in (6), the parameter T is fixed, we have that

$$j \neq T : U_i^1 = j \quad \Rightarrow \quad U^2 = 0 \tag{8}$$

$$j \neq 0 : U_i^2 = j \quad \Rightarrow \quad U^1 = T. \tag{9}$$

Thus we can arrive at the following for MAP decoding:

$$\begin{aligned} u < T : P\left(U_i = u | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) &= P\left(U_i^1 = u, U_i^2 = 0 | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) \\ &= P\left(U_i^2 = 0 | U_i^1 = u, \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) P\left(U_i^1 = u | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) \\ &= P\left(U_i^1 = u | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) \\ &= \text{app}_i^{U^1}(u) \text{ owing to (8)} \\ u > T : P\left(U_i = u | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) &= P\left(U_i^1 = T, U_i^2 = u - T | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) \\ &= P\left(U_i^1 = T | U_i^2 = u - T, \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) \\ & \quad P\left(U_i^2 = u - T | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) \\ &= P\left(U_i^2 = u - T | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) \\ &= \text{app}_i^{U^2}(u - T) \text{ owing to (9)} \\ P\left(U_i = T | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) &= 1 - \sum_{u \neq T} P\left(U_i = u | \underline{s}_u^1, \underline{s}_u^2, \underline{s}_v\right) \\ &= 1 - \left(\sum_{u=0}^{T-1} \text{app}_i^{U^1}(u)\right) - \left(\sum_{u=T+1}^{Q-1} \text{app}_i^{U^2}(u - T)\right) \end{aligned}$$

Since the APP outputs of the sum-product decoder are approximate, this MAP decoder that uses the APP outputs itself is approximate. In particular, we see that the final equation above can be negative. In the event of it being negative, we may simply set it to a small $\epsilon > 0$ or 0. Nonetheless, the MAP estimator is simple.

3.2 Source Splitting a Single Source

For splits defined in terms of (7), we may transform the source coding of a single source into a binary multiple source Slepian-Wolf problem corresponding to decoding at a vertex

point. The benefit of this approach lies in its low-complexity decoding (which is analogous to figure 2).

The MAP estimator for symbol i of $\underline{U}$ is given by

$$\hat{U}_i = \arg \max_{u \in \{0,1,\ldots Q-1\}} P\left(U_i = u | \underline{s}_u^1, \underline{s}_u^2, \ldots, \underline{s}_u^{Q-1}\right)$$

We note that the sum-product algorithm for iterative decoding produces approximate a posteriori posteriori probabilities (APPs) $U_i^1, U_i^2, \ldots, U_i^{Q-1}$:

$$\begin{aligned} P\left(U_i^1 = u | \underline{s}_u^1, \underline{s}_u^2, \ldots, \underline{s}_u^{Q-1}\right) &\triangleq \text{app}_i^{U^1}(u), \\ &\vdots \quad \vdots \\ P\left(U_i^{Q-1} = u | \underline{s}_u^1, \underline{s}_u^2, \ldots, \underline{s}_u^{Q-1}\right) &\triangleq \text{app}_i^{U_i^{Q-1}}(u). \end{aligned}$$

For splits defined in terms of (7) we notice that

$$k \in \{1, \ldots, Q-1\} \text{ and } U_i^k = 1 \Rightarrow U_i^r = 0, r \neq k \tag{10}$$

Thus we can arrive at the following for MAP decoding:

$$\begin{aligned} k \neq 0 : P\left(U_i = k | \underline{s}_u^1, \underline{s}_u^2, \ldots \underline{s}_u^{Q-1}\right) &= P\left(U_i^k = 1, U_i^r = 0, r \neq k | \underline{s}_u^1, \underline{s}_u^2, \ldots \underline{s}_u^{Q-1}\right) \\ &= P\left(U_i^r = 0, r \neq k | U_i^k = 1, \underline{s}_u^1, \underline{s}_u^2, \ldots \underline{s}_u^{Q-1}\right) \\ &\quad P\left(U_i^k = 1 | \underline{s}_u^1, \underline{s}_u^2, \ldots \underline{s}_u^{Q-1}\right) \\ &= P\left(U_i^k = 1 | \underline{s}_u^1, \underline{s}_u^2, \ldots \underline{s}_u^{Q-1}\right) \\ &= \text{app}_i^{U^k}(1) \text{ owing to (10)} \\ P\left(U_i = 0 | \underline{s}_u^1, \underline{s}_u^2, \ldots \underline{s}_u^{Q-1}\right) &= 1 - \sum_{k=1}^{Q-1} P\left(U_i = k | \underline{s}_u^1, \underline{s}_u^2, \ldots \underline{s}_u^{Q-1}\right) \\ &= 1 - \sum_{k=1}^{Q-1} \text{app}_i^{U^k}(1), \end{aligned}$$

giving another very simple MAP estimator.

Figure 3 illustrates the performance of the splitting approach using iterative sum-product decoding. Regular LDPCs of length $n = 500$ over $GF(4)$ were used to code memoryless sources. The leftmost plot shows iterative decoding results for coding two correlated sources U and V, where $|U| = |V| = 4$. The symbol error rate (SER) plots are shown for four non-vertex rate pairs, as a function of the sum rate difference from the entropy boundary. We note that in comparison to the joint decoding scheme proposed in section 2 under a turbo-like decoding schedule, this scheme achieves lower SER for the same rate. The rightmost plot shows iterative decoding results for splitting a single source U, where $|U| = 4$, into binary sources and performing iterative decoding with the piplelined approach. This is compared to block-compressing U and applying iterative decoding. We note that these results are merely proof-of-concept illustrations: significant gains can be acquired using irregular LDPCs, using larger block lengths, and applying density evolution.

4 Approximate ML Decoding Using Linear Programming

The rationale for this section is provided by the recent work of Feldman et al [15, 16]. We first show how ML decoding for block source coding using LDPCs can be approximated

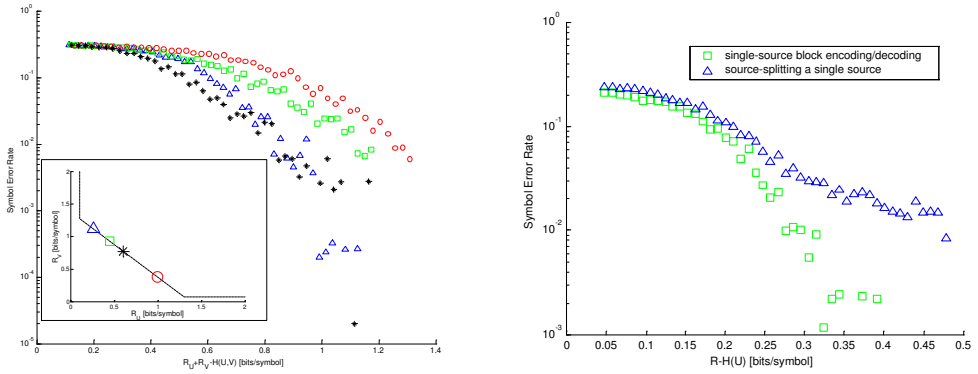

Figure 3: SER for source-splitting (L) Slepian-Wolf; (R) a single source.

using a linear program with the ML-certificate property. We then extend this to the binary Slepian-Wolf case for encoding at a vertex point. Suppose $u_i \in \{0,1\}, i = 1, \ldots, n$ and we use a linear code with syndrome former matrix H to compress $\underline{u}$ into a length k syndrome $\underline{s}$. If we have that $\gamma \triangleq \ln\left(\frac{P(U_i=0)}{P(U_i=1)}\right)$, then the ML estimate of u is given by:

$$\begin{aligned}
\underline{u}^* &= \arg \max_{\underline{u}\in\{0,1\}^n} P(\underline{U}=\underline{u}|\underline{S}=\underline{s}) = \arg \max_{\underline{u}\in\{0,1\}^n} P(\underline{S}=\underline{s}|\underline{U}=\underline{u})P(\underline{U}=\underline{u}) \\
&= \arg \max_{\underline{u}\in\{0,1\}^n} 1_{\{H\underline{u}=\underline{s}\}} \prod_{i=1}^{n} P(U_i=u_i) = \arg \min_{\underline{u}\,:\,H\underline{u}=\underline{s}} \sum_{i=1}^{n} -\ln P(U_i=u_i) \\
&= \arg \min_{\underline{u}\,:\,H\underline{u}=\underline{s}} \sum_{i=1}^{n} \Big(\ln P(U_i=0) - \ln P(U_i=u_i)\Big) \\
&= \arg \min_{\underline{u}\,:\,H\underline{u}=\underline{s}} \sum_{i:u_i=1} \ln\left(\frac{P(U_i=0)}{P(U_i=1)}\right) \\
&= \arg \min_{\underline{u}\,:\,H\underline{u}=\underline{s}} \sum_{i=1}^{n} \ln\left(\frac{P(U_i=0)}{P(U_i=1)}\right) u_i = \arg \min_{\underline{u}\,:\,H\underline{u}=\underline{s}} \sum_{i=1}^{n} \gamma u_i
\end{aligned}$$

By defining $\mathcal{P}(\underline{s}) = CH\left(\{\underline{u} : H\underline{u} = \underline{s}\}\right)$, where CH denotes the convex hull, the above problem can be cast as a linear program:

$$\underline{u}^* = \arg \min_{u\in\mathcal{P}(\underline{s})} \sum_{i=1}^{n} \gamma u_i.$$

The polyhedron however is in general too complex to represent explicitly and thus the problem is NP-hard. We propose using a relaxed polytope $\tilde{\mathcal{P}}(\underline{s}) \supseteq \mathcal{P}(\underline{s})$ that includes not only valid codewords as vertices, but also fractional vertices. This relaxation exhibits the ML-certificate property: if an integral LP solution is found, it is the ML-codeword.

We construct $\tilde{\mathcal{P}}(\underline{s})$ as the intersection of k polyhedra $\tilde{\mathcal{P}}_j(s_j)$, where each $\tilde{\mathcal{P}}_j(s_j)$ is the convex hull of all code symbols consistent with the local parity check j and syndrome s_j. Note that (as in the left-hand side of figure 1), parity-check j is connected to one syndrome symbol s_j and a set of δ_j adjacent variable nodes, given by $\bar{N}(j)$. If $s_j = 0$, the valid configurations the δ_j adjacent variables form is a $(\delta_j, \delta_j - 1, 2)$ binary linear code $\mathcal{C}_j(0)$. We define $[\mathcal{C}_j(0)]$ to be the matrix with each codeword of $\mathcal{C}_j(0)$ as a column vector.

In the event that $s_j = 1$, we see that $\mathcal{C}_j(1) = \{0,1\}^{\delta_j} \setminus \mathcal{C}_j(0)$ and $[\mathcal{C}_j(1)]$ is defined similarly. Then we have that

$$\begin{aligned}\tilde{\mathcal{P}}_j(s_j) &= \left\{ \underline{u} \in \mathbb{R}^{\delta_j} \; s.t. \; \underline{u} = [\mathcal{C}_j(s_j)]\,\underline{x}, \; \underline{x} \in \mathbb{R}^{2^{\delta_j - 1}}, 0 \le x_i \le 1, \sum_i x_i = 1 \right\} \\ \tilde{\mathcal{P}}(\underline{s}) &= \left\{ \underline{u} \in \mathbb{R}^n \; s.t. \; \underline{u}_{|\bar{N}(j)} \in \tilde{\mathcal{P}}_j(s_j), \; j = 1, \ldots, k \right\},\end{aligned}$$

where $\underline{u}_{|V}$ is defined to be the restriction of $\underline{u}$ to the coordinates in V.

Interestingly, in the context of LP channel decoding, the input log likelihood ratios change (and thus the objective function changes) as a function of the channel output, but the polyhedron is fixed. In the source coding case, the log-likelihood ratio γ (and thus the objective function) is fixed, but the polyhedron changes depending upon the syndrome. Another interesting aspect to the source coding case is that it is essentially *universal*: detailed information about γ need not be known - the optimization algorithm is invariant to scaling γ by a positive constant.

We next show how this extends to decoding at vertex points of the Slepian-Wolf problem. Suppose that we have encoded at U at rate $R_u > H(U)$ and V at rate $R_v > H(V|U)$. Suppose we decoded U correctly and must now decode binary V using U as side information. Suppose U lies in $\{0, 1, \ldots, Q-1\}$ and by assumption $V \in \{0,1\}$. Then we define the following likelihood ratios:

$$\gamma_0 \triangleq \ln\left(\frac{P(V_i = 0|U_i = 0)}{P(V_i = 1|U_i = 0)}\right), \; \ldots, \; \gamma_{Q-1} \triangleq \ln\left(\frac{P(V_i = 0|U_i = Q-1)}{P(V_i = 1|U_i = Q-1)}\right).$$

By performing analysis similar to the above derivations, we arrive at the following Slepian-Wolf ML relaxation

$$\underline{v}^* = \arg \min_{\underline{v} \in \tilde{\mathcal{P}}(\underline{s}_v)} \sum_{k=0}^{Q-1} \sum_{i:U_i = k} \gamma_k v_i$$

and note that it also exhibits the ML-certificate property.

5 Conclusion

Much of this paper results from the duality between source and channel coding. We take ideas developed for channel coding and transform them appropriately to construct new source coding techniques. We can accomplish distributed source compression using codes on graphs with appropriate message-passing between all graphs that takes into account the joint distribution between the sources. The iterative decoding procedure has a 'turbo-like' interpretation. We also illustrated how to achieve Slepian-Wolf rates besides vertices with low-complexity iterative decoding by applying source-splitting. This operation was also shown to be applicable for single-source block coding to reduce decoder complexity. All of these multi-user source coding results were motivated by recent results in the multiple access channel coding literature. Finally, we constructed linear programming relaxations to the maximum-likelihood decoding problem for source coding. We illustrated these relaxations possess the ML-certificate property and showed relaxations for both the single-source binary block coding case as well as the binary Slepian-Wolf vertex case. This approach as well had its origins in the channel coding domain.

References

[1] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 471–480, 1973.

[2] I. Csiszár, "Linear codes for sources and source networks: Error exponents, universal coding," *IEEE Transactions on Information Theory*, vol. 28, no. 4, pp. 585–592, 1982.

[3] T. Tian, J. Garcia-Frias, and W. Zhong, "Compression of correlated sources using LDPC codes," in *IEEE Data Compression Conference*, 2003.

[4] D. Schonberg, S. S. Pradhan, and K. Ramchandran, "LDPC codes can approach the Slepian-Wolf bound for general binary sources," in *Proceedings of the 40th Allerton Conference on Communication, Control and Computing*, October 2002.

[5] A. D. Liveris, Z. Xiong, and C. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," *IEEE Communications Letters*, vol. 6, pp. 440–442, October 2003.

[6] J. Garcia-Frias and Y. Zhao, "Compression of correlated binary sources using turbo codes," *IEEE Communications Letters*, vol. 5, pp. 417–419, October 2001.

[7] A. Aaron and B. Girod, "Compression with side information using turbo codes," in *IEEE Data Compression Conference*, April 2002, pp. 252–261.

[8] J. Bajcsy and P. Mitran, "Coding for the Slepian-Wolf problem with turbo codes," in *IEEE GLOBECOM*, November 2001, pp. 1400–1404.

[9] A. Liveris, Z. Xiong, and C. Georghiades, "Distributed compression of binary sources using conventional parallel and serial concatenated convolutional codes," in *Proc. IEEE DCC*, Brest, France, March 2003, pp. 193–202.

[10] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): Design and construction," *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 626–643, 2003.

[11] G. D. Forney, "Codes on graphs: Normal realizations," *IEEE Transactions on Information Theory*, pp. 101–112, 2001.

[12] R. Palanki, A. Khandekar, and R. McEliece, "Graph-based codes for synchronous multiple access channels," in *Proceedings of the 39th Allerton Conference on Communication, Control and Computing*, October 2001.

[13] A. Grand, B. Rimoldi, R. Urbanke, and P. A. Whiting, "Rate-splitting multiple access for discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 47, no. 3, pp. 873–890, 2001.

[14] B. Rimoldi and R. Urbanke, "Asynchronous Slepian-Wolf coding via source-splitting," in *IEEE International Symposium on Information Theory*, Ulm, Germany, June 29–July 4 1997, p. 271.

[15] J. Feldman, M. Wainwright, and D. R. Karger, "Using linear programming to decode linear codes," *Proceedings of Conference on Information Sciences and Systems, The John Hopkins University*, March 2003.

[16] J. Feldman, *Decoding Error-Correcting Codes via Linear Programming*, PhD dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, September 2003.