Distributed Source Coding Using Serially-Concatenated-Accumulate Codes

Johnny Chen
DEAS
Harvard University
Cambridge, MA 02138
johnnyc@eecs.harvard.edu

Ashish Khisti
RLE
MIT
Cambridge, MA 02139
khisti@mit.edu

Dmitri M. Malioutov
LIDS
MIT
Cambridge, MA 02139
dmm@mit.edu

Jonathan S. Yedidia
MERL
201 Broadway, 8th Floor
Cambridge, MA 02139
yedidia@merl.com

Abstract

We describe a practical method for distributed compression of q-ary sources using multi-level serially-concatenated-accumulate codes. Our approach works well at high compression rates, and allows for graceful and incremental rate-adaptivity. Simulations show that the compression efficiency is near the information-theoretic limits for correlations between sources that obey a Gaussian or Laplacian distribution.

I Introduction

In 1973, Slepian and Wolf proved that correlated sources can be compressed without loss using just as few bits, whether or not the encoders of the separate sources are permitted to communicate [1]. Wyner and Ziv soon thereafter extended the Slepian-Wolf theorem to the case of lossy compression [2]. A first clue about how to construct a practical system that achieved these surprising results was provided by Wyner [3], who suggested that the encoders should transmit syndrome bits using error-correcting channel codes. In Wyner's approach, a decoder would decode a particular source X, using the syndrome bits provided by the encoder of X combined with the side information about a correlated source Y provided by another encoder.

There followed a period of roughly twenty-five years during which few if any attempts were made to construct practical systems that achieved the performance promised by Slepian-Wolf and Wyner-Ziv [4]. Recently, however, a variety of interesting proposals have been made, see e.g. [5, 6], and references therein. One reason for the new interest in practical distributed source coding schemes has been the generally increasing attention given to distributed sensor networks. Another important motivation is the potential of systems that borrow the idea of syndrome-based encoding and decoding to compress video in a way that shifts the complexity burden from the encoder to the decoder [7, 8].

In our view, a good distributed source coding method should satisfy the following requirements. First, it should be capable of compressing integer-valued sources with wide dynamic range. (Real-valued sources will normally quantized to integer values in any case.) Second, it should work well at very high compression ratios. Third, it should be gracefully rate-adaptive. That is, it should be possible to use different rate codes as the source correlations change, with minimal complexity, and ideally without even any need to explicitly transmit which code is being used. Fourth, it should be incremental, in the sense that one can send some bits to the decoder, and then if the decoder needs more bits, send those additional bits without wasting any information previously sent. Fifth, it should achieve compression efficiencies near the information-theoretic bounds. Finally, the complexity should scale linearly or nearly linearly with the size of the sources.

These requirements all arise naturally in video compression applications. For example, because of the large temporal redundancy, high compression rates are possible; but the level of redundancy changes constantly throughout a video stream, so rate-adaptivity is also essential.

We have developed new syndrome-based distributed source coding methods that satisfy all these requirements. They use, in an appropriately modified form, recently proposed error-correcting channel codes that we call "serially-concatenated-accumulate" (SCA) codes. SCA codes generalize the well-known repeat-accumulate (RA) codes [9], by replacing the repetition codes in RA codes with other "base codes." If the new base codes are products of single parity checks, the SCA code is a product-accumulate (PA) code [10]. If the new base codes are extended Hamming codes, the resulting SCA code is what we call an "extended-Hamming-accumulate" (EHA) code [11, 12]. Low complexity decoders of PA and EHA codes have both been shown to perform close to optimally at high rates for channel coding, which make them promising candidates for syndrome compression methods that need to operate at high compression rates.

The outline for the rest of the paper is as follows. In Section II, we will set up more precisely the problem that we are interested in solving. In Section III, we describe how bit-level syndrome encoders and decoders can be constructed using SCA codes. In section IV, we describe how PA and EHA codes can be conveniently divided and sub-divided into other codes of the same type, a property that we exploit to obtain graceful and incremental rate-adaptivity. In section V, we describe how multi-level codes can be constructed using SCA codes, and how to decode using a multi-stage decoder, to allow for encoding and decoding of q-ary sources. Finally, in

section VI, we present simulation results that show that our method gives performance that is quite close to the information theoretic bounds, at least for Gaussian and Laplacian probability distributions.

II Problem Set-up

We suppose that the sources Z, Y, X, W, etc., each consist of n integers ranging over $q = 2^m$ values. The sources are assumed to be statistically correlated, with a joint probability distribution that has the form of a Markov chain:

$$p(z, y, x, w, ...) = p(z)p(y|z)p(x|y)p(w|x).... \quad (1)$$

One source (say, Z) will be transmitted directly, and syndrome bits will be sent for each of the other sources. We can then decode by solving a series of "source coding with side information" (SCSI) problems. That is, we can decode the source Y using Z as side information, then decode X using Y as side information, and so on.

This set-up naturally describes the video compression problem, where the different sources are the different frames of the same video. Of course, for that problem, the encoders have no problem in communicating, but there are other reasons (e.g. complexity at the encoder and error-resilience) why one may want to use a syndrome-based scheme. [7, 8].

Each syndrome encoder needs to make a decision about how many syndrome bits to send. There are a variety of different scenarios for how this decision might be made. In the first scenario, which is the classical scenario for the SCSI problem, the encoder for X is given access to the conditional probability $p(x|y)$ (but not samples from Y itself), and can therefore estimate the entropy $H(X|Y)$, and send the appropriate number of bits.

In a second possible scenario, there is a a small feedback channel from the decoder telling an encoder whether it can decode. To exploit such a feedback channel, one needs to be able to transmit syndrome bits incrementally. The advantages of this scenario are that one does not need to make accurate estimates of the entropy, and one can eventually send enough syndrome bits to ensure successful decoding.

Finally, in a third scenario, the encoders might actually communicate, and also run a version of the decoder to ensure that enough syndrome bits are sent. In this scenario the normal reasons given for using a syndrome-based approach are not valid (since the encoders communicate, and there is no complexity savings, even at the encoder), but on the other hand, the compression performance can be very good, and syndrome-based methods can potentially out-perform conventional compression methods. Within the context of this scenario, methods for text compression using low-density parity check codes have recently been proposed and shown to have excellent performance [13].

Throughout, we will assume that the conditional probability function $p(x|y)$ factors over the integers x_i and y_i like

$$p(x|y) = \prod_{i=1}^{n} p(x_i|y_i) \quad (2)$$

The exact form of the distributions $p(x_i|y_i)$ that we consider will be detailed in section VI.

III Syndromes from SCA Codes

SCA codes are formed by a serial concatenation of a rate-1 accumulator, an interleaver, and a set of base codes. We focus here on PA codes, for which the base codes are product codes constructed from single-parity-check (SPC) codes, and EHA codes, for which the base codes are extended Hamming codes.

In this section, we will begin the process of describing our full system by showing how to construct bit-level syndrome encoders and decoders using PA and EHA codes.

A Encoding

To encode, i.e. compress, a source $[x_1, \ldots, x_n]$, we first apply an inner encoder, which is actually an inverse accumulator. For now, we assume that the source is binary. The inverse accumulator simply concatenates the first source bit with the modulo-2 sum of consecutive pairs of following source bits. The output of the accumulator is then interleaved (according to a fixed permutation) and divided into p blocks. For now, we take the p blocks to have equal size; we will lift this assumption later. For PA codes, the blocks are $s \times t$ arrays (so that $n = pst$), while for EHA codes these blocks have length 2^r, (so that $n = p2^r$).

The syndrome bits are then computed according to the appropriate base code. For PA codes, the syndrome bits are computed as the modulo-2 sums of single parity checks defined using the rows and columns of the arrays. Since the $s + t$ syndromes corresponding to each array have total parity zero, exactly one is redundant and does not need to be sent. Therefore a PA code with an outer code using p equal-sized $s \times t$ arrays gives a compression ratio (defined as the number of source bits divided by the number of syndrome bits) of $st/(s + t - 1)$. For EHA codes, the syndrome bits are computed using the syndromes of the $[N = 2^r, k = 2^r - (r + 1), d = 4]$ extended Hamming codes. Therefore an EHA code with an outer code using p extended Hamming codes of length 2^r will have a compression ratio of $2^r/(r + 1)$.

A factor graph [14] for an SCA code, including the syndrome bits, is shown in figure 1. Of course, the number of syndrome bits used will depend on the base codes–the SCA code shown in the figure is a toy code that uses three $[N = 3, k = 1, d = 3]$ repetition codes as base codes. The encoding operation that we have described starts with the source bits at the bottom of the factor graph, and

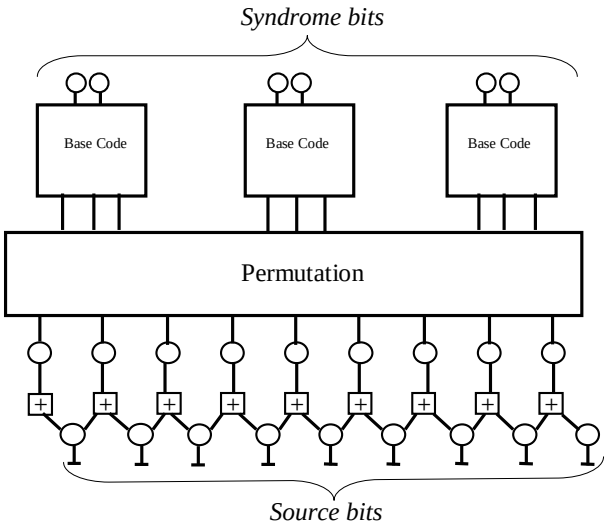

Figure 1: A factor graph for a syndrome encoder/decoder using SCA codes.

proceeds deterministically up to the syndrome bits at the top of the factor graph.

B Decoding

The decoder uses received syndrome bits and the correlated source Y to estimate the original message. Before decoding, we first compute the values of any redundant syndrome bits. The decoder then uses a turbo-like algorithm which passes soft extrinsic information between decoders of the base codes and the accumulator code.

The accumulator code is decoded using the BCJR algorithm, which generates extrinsic information for each bit for the decoders of the base codes. Side information from the source Y can be treated just like soft evidence from the channel in channel coding. This side information is represented by the "dongles" attached to the source bit variable nodes in figure 1. We will say more about how this evidence is generated in section V.

Different algorithms can be used for decoding the base codes. For EHA codes, we used symbol-MAP decoders of extended Hamming codes [15] that return soft extrinsic information. Because the complexity of these decoders scales with the block-length N of the extended Hamming code as $N \log N$, very high rate EHA codes can be used efficiently. For product codes, we followed [10] and used belief propagation-based decoders.

The over-all decoder uses a turbo-decoding algorithm to iterate between the decoder of the accumulator code and the decoders of the base codes. After a fixed number of turbo iterations were reached, we check whether a coset codeword (i.e., a word that satisfies all the syndrome bits, and all the constraints in the accumulator code) is reached.

IV Variable-Rate Compression

In this section, we lift our assumption that all the base codes in the SCA code have equal block-length. By permitting the different base codes in an SCA code to have different sizes, we will be able to adjust the rate of the overall code to have nearly any possible desired value.

We first describe the approach using PA codes. Suppose that our PA code has p base codes, and the ith base code is a product of SPC codes of size s_i by t_i, where s_i and t_i are both powers of two. That means that the ith base code contributes $s_i + t_i - 1$ syndrome bits using the encoding described in the previous section.

Suppose that we want to send more syndrome bits than the $\sum_i (s_i + t_i - 1)$ that are currently being sent. We can easily do this by "splitting" one of the base codes into two. For example, if the ith base code is a product code of size 8 by 8, we can split it into two 4 by 8 product codes. Incidentally, in our implementations, we always tried to keep the number of rows and columns in our product codes as equal as possible.

Many of the syndrome bits for the new, smaller, product codes will not need to be transmitted, because they already have been sent or are redundant. In the example just described, the syndrome bits corresponding to the columns of the 4 by 8 product code need not be sent, because they can be obtained from the corresponding syndrome bits for the 8 by 8 product code. In this example, the syndrome bits for the rows of one of the new, smaller, product codes will need to be transmitted, but the syndrome bits for the rows of the other smaller code can be determined using the new syndrome bits sent, in combination with the syndrome bits that had previously been sent for the rows of the 8 by 8 product code.

It is not difficult to verify that by splitting the base codes recursively, we can incrementally vary the number of syndrome bits sent, without ever wasting any information from previously sent syndrome bits.

A similar approach works for EHA codes, based on the fact that any extended Hamming code can be elegantly "split" into two smaller extended Hamming codes. Suppose that our EHA code has p base codes, and the ith base code is an extended Hamming code of block-length 2^{r_i}. Therefore, the ith base code will contribute $r_i + 1$ syndrome bits, and the total number of syndrome bits sent will be $\sum_i (r_i + 1)$.

Now suppose that we want to send more syndrome bits–the idea is once again to "split" one of the extended Hamming codes into two smaller extended Hamming codes, which can always be done. For example, suppose that the base code to be split is an $[N = 8, k = 4]$ extended Hamming code with parity check matrix

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}, \qquad (3)$$

and we want to split this code into two $[N = 4, k = 1]$ extended Hamming codes with parity check matrices

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}. \quad (4)$$

The $[N = 8, k = 4]$ code would provide four syndrome bits, and we would need six syndrome bits for the two $[N = 4, k = 1]$ extended Hamming codes. The syndrome bits for the first row of one of the smaller $[N = 4, k = 1]$ codes can be obtained from the second row of the parity check matrix of the $[N = 8, k = 4]$ code. Suppose that we send the syndrome bits of the second and third row for that smaller code. Then all the syndrome bits for the other smaller code can be obtained using some combination of the syndrome bits from the $[N = 8, k = 4]$ code, and the syndrome bits transmitted for the first $[N = 4, k = 1]$ code. Thus, if we already transmitted the syndromes for the $[N = 8, k = 4]$ code, we can obtain all the necessary syndrome bits for the two smaller codes by sending just two more syndrome bits, and no syndrome bits will be wasted.

Although we have just explained how to split extended Hamming codes into two using a small example, the regular structure of these codes makes it very easy to generalize the example to larger sizes.

To reiterate the essential point of this section: we can gracefully and incrementally vary the compression ratio in our approach by recursively splitting the base codes in the SCA code. A very important point is that the encoders and the decoder must agree on a pre-determined splitting schedule. Given such a schedule, the structure of the code can be directly inferred at the decoder from the number of transmitted syndrome bits.

V Multilevel Codes for q-ary Sources

We now lift the assumption that the source is binary, and consider instead a q-ary source, where $q = 2^m$. To encode such a source, we will use a multi-level code, with m levels or "bit-planes." We will decode this multi-level code using a multi-stage decoder, analogous to the decoders for block coded modulation multi-level codes pioneered by Imai and Hirakawa in 1977 [16].

To be more precise, let the q-ary source X consist of n integers, each ranging from 0 to $2^m - 1$, and let

$$X_i = X_i^1 X_i^2 \ldots X_i^m, \; X_i^k \in \{0, 1\}$$

be a binary representation of the ith integer X_i. We considered a variety of binary representations of the integers, including the standard binary representation, a Gray code, and a few other possibilities, but will present results for the standard representation, because the performance using it was at least as good as that of other representations.

After converting the source into a set of m bit-planes, each bit-plane is separately encoded into syndrome bits using its own binary SCA code.

The decoding proceeds by decoding one bit-plane after the other. We found by simulation that the minimal overhead was achieved when one begins with the least significant bit plane, and then proceeds to the next most significant bit plane, and so on until one decodes the most significant bit plane.

Each bit-plane decoder needs, as input, a set of *a priori* probabilities for each of the n bits. These *a priori* probabilities for x_i, the ith bit in X, are computed based on the conditional probability distribution $p(x_i|y_i)$, conditioned further on the previously decoded bit-planes of X. Thus, as bit-planes of X are decoded, the results are used to help determine the a priori bit probabilities for subsequent bit-planes. The idea behind these computations is illustrated in figure 2.

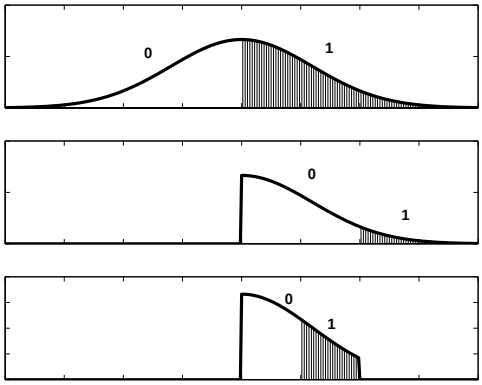

Figure 2: An illustration of how the conditional probability distribution will be partitioned based on previously decoded bit-planes. For simplicity, we assume here that the most significant bit is decoded first, followed by the next most significant bit, and so on. In this example, the most significant bit is decoded as a one, and the second most significant bit is decoded as a zero, so the third bit's distribution is in the region near to zero, but to the right. The *a priori* probability for a bit to be a zero or one is obtained by integrating the appropriate part of the distribution, and normalizing.

VI Simulation Results

We now present simulation results for PA and EHA codes using synthetic sources. We consider two possible models for the conditional probability: the Gaussian case, for which

$$p(x_i|y_i) \propto \exp\{\frac{(x_i - y_i)^2}{2\sigma^2}\}, \quad (5)$$

and the Laplacian case, for which

$$p(x_i|y_i) \propto \exp\{\lambda|x_i - y_i|\}. \quad (6)$$

The importance of the Gaussian distribution needs no special explanation, while the Laplacian model provides a very good fit for many image processing tasks, such as the distribution of the residuals of two consecutive frames in a video sequence. Since we only consider q-ary source symbols, we sample the Gaussian and Laplacian distributions on the grid of integers between 0 and $q-1$. In all our simulations we let $q = 256$. We set all the symbols in the side information sequence y to equal 128 and hence the source symbols x_i are generated from the corresponding discrete distribution centered around 128. This set-up is not as artificial as it may appear; in image or video compression, one typically compresses the residual error compared with a predicted value, and the residual error will have zero mean, which we have simply re-centered to 128 for convenience.

In the simulations we use the incremental rate adaptivity feature of our system as discussed in secton IV. For system based on PA codes, we used a splitting schedule that first split the columns (from left to right) and then the rows from (top to bottom) at any given level. For the EHA codes, we always split the first available largest block into two smaller blocks of half the size. While the PA codes can send syndromes in increments of one, the EHA codes require slightly larger increments.

The overall protocol used was to send syndrome bits incrementally until the decoder was able to successfully decode to a coset codeword. In our experiments, the decoder never failed by decoding to an incorrect coset codeword, but instead would fail by reaching a word that did not satisfy all the constraints. If decoding failed, more syndrome bits were sent, until decoding ultimately succeeded.

The simulations were performed on synthetic sources of length size 4096 (4K) and 16384 (16K). The number of turbo iterations for the EHA codes was set to 30 while that of the PA codes was set to 20. The number of trials for sources of length 4K was chosen to be 1000, and for length 16K it was 100. For all the figures shown here, the error-bars will be smaller than the size of points used to plot the data.

In figure 3, we plot, for both the system that uses PA codes and the one that uses EHA codes, the overall number of syndrome bits used in our method, compared with the computed conditional entropy $H(X|Y)$, which sets a theoretical lower bound. In this plot, we use sources of length 4K and a Gaussian distribution. Note that both systems closely approach the entropic limit, although the system based on EHA codes performs somewhat better than that based on PA codes.

Figure 4 plots the same data, along with the data for the sources of 16K, in a different way, as the "overhead" required by our method. Note that the overhead required by the system that uses EHA codes is roughly 2% over a wide range of the standard deviation σ, and it is not strongly dependent on the blocklength. The non-monotonic relation between the overhead and the standard deviation for the system based on PA codes is probably a result of complicated details concerning how the entropy is shared between different bitplanes, and the relative effectiveness of PA codes at different rates.

Figure 5 helps fill in some of these details and explain why our systems perform so well. We plot the number of syndrome bits used and the conditional entropy for each bit-plane in our system when $\sigma = 1$. For bitplanes with conditional entropy close to 1 the overhead is close to zero since we are sending a number of syndrome bits equal to the number of source bits. This happens for the few least significant bitplanes which are decoded first in our simulations. When the conditional entropy is very near zero the source can be recovered from their a priori probabilities wihout sending any syndromes. This is the case for the few most significant bitplanes, which we decode last. Finally, for the intermediate bit planes EHA codes were observed to incur lower overhead than the PA codes.

Figure 6 compares the overhead incurred with Laplacian and Gaussian models. We note that the Laplacian model requires somewhat higher overhead than the Gaussian model, although the overall overhead is still quite small. This can be explained by noting that the Laplacian distribution has more bit planes with intermediate values of the conditional entropy than the Gaussian model (cf. figure 5).

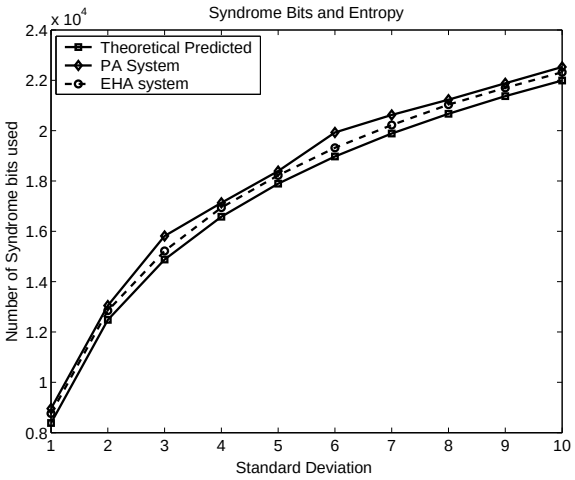

Figure 3: Syndrome bits used and entropy for systems based on EHA and PA codes using sources of length 4K and Gaussian correlations.

Acknowledgements

We thank Anthony Vetro, Emin Martinian, Min Wu, and Bill Freeman for helpful discussions, and Karunakar Pedagani for sharing his software for a symbol-MAP decoder for extended Hamming codes.

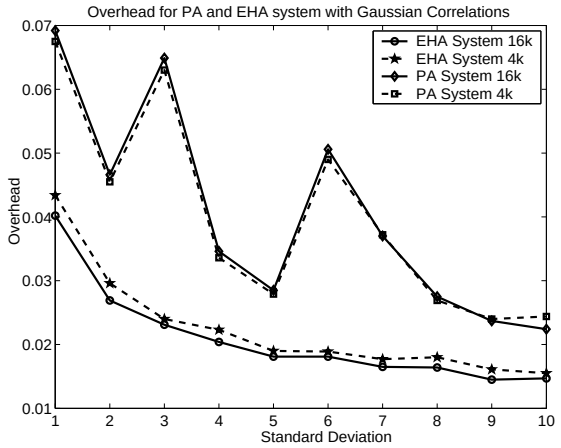

Figure 4: Overhead of EHA and PA codes, using Gaussian correlations.

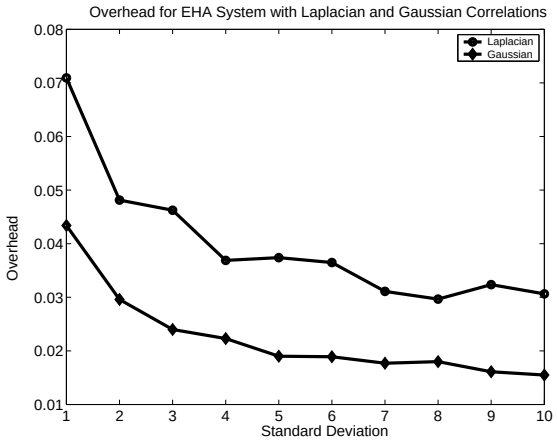

Figure 6: Overhead for EHA systems of length 4K– Laplacian vs. Gaussian correlations.

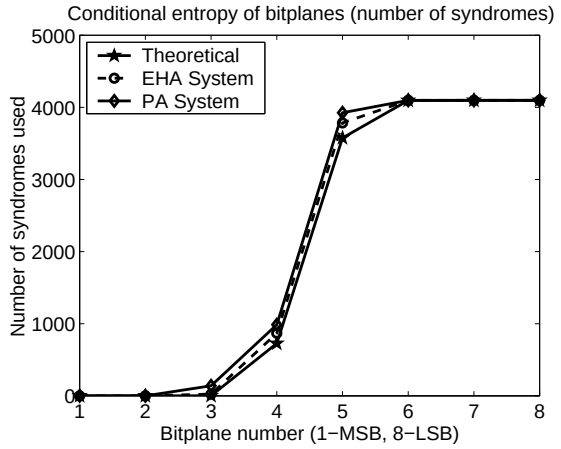

Figure 5: Overhead for each bit-plane for Gaussian correlations with $\sigma = 1$. (Length = 4K.)

References

[1] D. Slepian and J.K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Information Theory,* vol. 19, pp. 471-480, July 1973.

[2] A.D. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Trans. Information Theory* vol. 22, pp. 1-10, Jan. 1976.

[3] A.D. Wyner, "On source coding with side information at the decoder," *IEEE Trans. on Information Theory*, vol. 21, pp. 244-300, May 1975.

[4] S. Verdu, "Fifty years of Shannon theory," *IEEE Trans. on Information Theory*, vol. 44, pp. 2047-2078, Oct. 1998 (see section II.F).

[5] S.S. Pradhan and K. Ramchandran, "Distributed Source Coding Using Syndromes (DISCUS): Design and Construction,," *IEEE Trans. Information Theory*, vol. 49, No. 3, pp. 626-643, March 2003.

[6] Z. Xiong, A. Liveris, and S. Cheng, "Distributed Source Coding for Sensor Networks," *IEEE Signal Processing Magazine* September 2004, to appear.

[7] B. Girod, A. Aaron, S. Rane, and D. Rebollo-Monedero "Distributed Video Coding", to appear in *Proc. of the IEEE*, Special Issue on Video Coding and Delivery, 2004.

[8] R. Puri and K. Ramchandran, "PRISM: A Video Coding Paradigm Based on Motion-Compensated Prediction at the Decoder", submitted to *IEEE Trans. on Image Processing*, 2003.

[9] D. Divsalar, H. Jin, and R.J. McEliece, "Coding Theorems for 'turbo-like' codes." *Proc. 36th Allerton Conf. on Communication, Control, and Computing*, pp. 201-210, Sept. 1998.

[10] J. Li, K.R. Narayanan, and C.N. Georghiades, "Product accumulate codes: a class of codes with near-capacity performance and low decoding complexity," *IEEE Trans. Information Theory*, vol. 50, No. 1, pp. 31-46, Jan. 2004.

[11] M. Isaka and M. Fossorier, "High rate serially concatenated coding with extended Hamming codes," *IEEE Communications Letters*, 2004, to appear.

[12] D. Divsalar and S. Dolinar, "Concatenation of Hamming codes and accumulator codes with high-order modulations for high-speed decoding," *IPN Progress Report 42-156*, Jet Propulsion Laboratory, Feb. 15, 2004.

[13] G. Caire, S. Shamai, and S. Verdu, "Lossless Data Compression with Error Correcting Codes," *Proc. International Symposium on Information Theory (ISIT 2003)*, p. 22, 2003.

[14] F.R. Kschsichang, B.J. Frey, and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. Information Theory*, vol. 47, No. 2, pp. 498-519, Feb. 2001.

[15] A. Ashikhmin and S. Litsyn, "Simple MAP decoding of first order Reed-Muller and Hamming Codes," *Proc. of IEEE Information Theory Workshop (ITW03)*, pp. 18-21. 2003.

[16] H. Imai, S. Hirakawa, "A New Multilevel Coding Method Using Error-Correcting Codes," *IEEE Trans. on Information Theory*, vol. 23, no. 3, 1977.