Distributed Arithmetic Coding

Marco Grangetto, *Member, IEEE*, Enrico Magli, *Member, IEEE*, Gabriella Olmo, *Senior Member, IEEE*

Abstract

We propose a distributed binary arithmetic coder for Slepian-Wolf coding with decoder side information, along with a soft joint decoder. The proposed scheme provides several advantages over existing schemes, and its performance is equal to or better than that of an equivalent scheme based on turbo codes.

Index Terms

Distributed source coding, arithmetic coding, Slepian-Wolf coding, Wyner-Ziv coding, compression.

EDICS: CL1.3.1

The authors are with Dip. di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24 - 10129 Torino - Italy - Ph.: +39-011-5644195 - FAX: +39-011-5644099 - E-mail: `marco.grangetto(enrico.magli,gabriella.olmo)@polito.it`. Corresponding author: Enrico Magli.

Distributed Arithmetic Coding

I. Introduction and background

Distributed source coding (DSC) considers a situation in which two (or more) statistically dependent data sources must be encoded by separate encoders. DSC theory proves that separate encoding is optimal, provided that the sources are decoded jointly. For example, with two sources it is possible to perform “standard” encoding of the first source (side information), and “conditional” encoding of the second one at a rate lower than its entropy, with no information about the first source available at the second encoder. This has numerous applications in the fields of video coding and sensor networks [1].

DSC coders typically represent the source using its syndrome (or the parity bits) of a suitable channel code of given rate, e.g. trellis codes [2], turbo codes [3] and low-density parity-check (LDPC) codes [4]. Although near-capacity DSC coders have been designed for simple ideal sources, their application to realistic signals typically incurs several problems. Capacity-achieving channel codes require very large data blocks (typically in excess of 10^5 symbols), while in many practical applications the basic units to be encoded are much smaller. Real-world sources are not stationary and would benefit from context-based modeling, which is not provided by existing Slepian-Wolf (S-W) coders. Furthermore, upgrading an existing compression algorithm like JPEG 2000 or H.264/AVC in order to provide DSC functionalities requires at least to redesign the entropy coding stage, adopting one of the existing DSC schemes.

Among these issues, the block length is particularly important. While the performance of some practical DSC coders on ideal sources with very large block length can be as close as 0.09 bits per symbol (bps) to the theoretical limit [1], so far DSC of real-world data has fallen short of its expectations, one reason being the necessity to employ much smaller blocks. For example, the PRISM video coder [2] encodes each macroblock independently, with a block size of 256 samples; for the coder in [5], the block size is equal to the number of 8x8 blocks in one picture (1584 for the CIF format). The performance of both coders is rather far from optimal, highlighting the need of DSC coders addressing realistic block lengths.

In this letter we propose a novel S-W coder that attempts to overcoming the issues above. By recognizing that, besides being provably optimal source codes, arithmetic codes also allow sequential metric-based decoding [6], we design a distributed arithmetic coder (DAC) that encodes a source at a rate smaller than its entropy by allowing the intervals to overlap and generating an ambiguous codeword, and its soft joint decoder that exploits knowledge of the side information to decode the source.

II. Distributed arithmetic coding

A. Arithmetic coding

Let $\mathbf{X} = [X_0, X_1, \ldots, X_i, \ldots, X_{N-1}]$ be a length-N binary symbol sequence with probabilities $p_0 = P(X_i = 0)$ and $p_1 = P(X_i = 1)$, and $\mathbf{Y} = [Y_0, Y_1, \ldots, Y_i, \ldots, Y_{N-1}]$ be a side information binary symbol sequence. The arithmetic coding process is based on the estimation of p_0 and p_1, and the source symbols are mapped onto sub-intervals of the $[0, 1)$ interval. At initialization the "current" interval is set to $[0, 1)$. For each input symbol X_i, the current interval is partitioned into two adjacent sub-intervals whose lengths are proportional to p_0 and p_1. The sub-interval representing X_i is selected as the next current interval, and this procedure is repeated for the next symbol. After all N symbols have been processed, the sequence is represented by the final interval I. The codeword $\mathbf{C}_X$ can consist in any binary number lying in I, and requires approximately $-\log_2 |I|$ bits, where $|I|$ is the length of I.

B. DAC encoder

DAC is based on the principle of inserting an ambiguity in the encoding process. This is obtained using a modified interval subdivision strategy, where the interval length is proportional to the modified probabilities $\widetilde{p}_0$ and $\widetilde{p}_1$, such that $\widetilde{p}_0 \geq p_0$ and $\widetilde{p}_1 \geq p_1$. In order to fit the enlarged sub-intervals $\widetilde{p}_j$ into the $[0, 1)$ interval, the sub-intervals are allowed to partially overlap. The intuition is that, since the intervals are not disjoint, the decoder will typically be unable to decode the source unambiguously without knowledge of the side information. As for the encoding rate, the overlapping leads to a larger final interval, and hence a shorter codeword. It should be noted that in [7] a P-ary arithmetic coder has been proposed for Slepian-Wolf coding, in which the same interval can be associated to multiple symbols, and the design of the unit interval partition is based on the joint density of the alphabet in order to achieve negligible probability of decoding error. The approach in [7] requires optimization of the interval partition for a given joint density. This leads to an encoder with complex modeling stage and a classical arithmetic decoder, whereas the DAC has a simple encoder and a joint sequential decoder.

At initialization the "current" interval is set to $[0, 1)$. For each input symbol X_i, the current interval is subdivided into two partially overlapped sub-intervals whose lengths are proportional to $\widetilde{p}_0$ and $\widetilde{p}_1$. The interval representing X_i is selected as the next current interval. After all N symbols have been processed, the sequence is represented by the final interval I; its codeword $\mathbf{C}_X$ can consist in any binary number lying in that interval.

In practice, we define $\widetilde{p}_j = \alpha_j p_j$, where p_j are the source probabilities, $\widetilde{p}_j$ the probabilities used to set the interval lengths, and $\alpha_j \geq 1$, so that $\widetilde{p}_0 + \widetilde{p}_1 \geq 1$. The average coding rate is

$$R_X = \sum_{j=0}^{1} p_j \left(\log_2 \frac{1}{p_j} - \log_2 \alpha_j \right) = \sum_{j=0}^{1} p_j \left(r_j - \delta r_j \right)$$

where $r_j = -\log_2 p_j$, and $\delta r_j = \log_2 \alpha_j$ represents the number of bits saved in the binary representation of the j-th input symbol. We impose that $\delta r_j / r_j$ is equal to a constant k, i.e. that the ambiguity is uniformly distributed over all input bits, which leads to the formula $\alpha_j = p_j^{-k}$, with $0 \leq k \leq 1$. In this latter case, the expression of the average coding rate can be further simplified as

$$R_X = (1-k) \sum_{j=0}^{1} p_j \log_2 \frac{1}{p_j} = (1-k) H$$

where H is the entropy of the probability model, i.e. $H = -\sum_{j=0}^{1} p_j \log_2 p_j$. It is evident that the coding rate can be reduced to any desired value by means of a suitable selection of the parameter k.

C. DAC decoder

The decoding process can be formulated as a symbol-driven sequential search along a proper decoding tree, where each node represents a state of the sequential arithmetic decoder. For DAC, we employ the *Maximum A Posteriori* (MAP) metric $P(\mathbf{X}|\mathbf{C}_X, Y)$; the decoder operates as follows.

1) Initialize current interval to $[0, 1)$.
2) Subdivide the interval according to the probabilities $\widetilde{p}_j$.
3) While the number of decoded symbols is less than N
   - For all the nodes in the tree
     - Compare $\mathbf{C}_X$ with the current interval subdivisions.
     - If $\mathbf{C}_X$ belongs to the non overlapped portion of a certain interval, decode the corresponding symbol X_i without ambiguity and select the corresponding interval for next iteration of this node.
     - If $\mathbf{C}_X$ lies in the overlapped region between two intervals, perform a decoding branching. Store two alternative paths in memory, corresponding to the two alternative symbols X_i that can be decoded, update their MAP metrics and select the corresponding interval for next iteration.
   - Prune the decoding tree by keeping only the M most likely paths.
4) Output the sequence corresponding to the most likely path in the tree.

Given the correlation model in terms of the probability distribution $P(\mathbf{X}|\mathbf{Y})$, the MAP metric $P(X_0, \ldots, X_i | \mathbf{C}_X, \mathbf{Y})$ can be evaluated iteratively as

$$P(X_0 \ldots, X_i | \mathbf{C}_X, \mathbf{Y}) = P(X_0, \ldots, X_{i-1} | \mathbf{C}_X, \mathbf{Y}) P(X_i | \mathbf{C}_X, \mathbf{Y})$$

with

$$P(X_i|\mathbf{C}_X, \mathbf{Y}) = \begin{cases} 1 & \text{if non ambiguous branching} \\ P(X_i|\mathbf{Y}) & \text{otherwise} \end{cases}$$

We adopt a memoryless correlation model with $P(\mathbf{X}|\mathbf{Y}) = \prod_{i=0}^{N-1} P(X_i|Y_i)$. The last term of the MAP metric can be then simplified as:

$$P(X_i|\mathbf{C}_X, Y_i) = \begin{cases} 1 & \text{if non ambiguous branching} \\ P(X_i|Y_i) & \text{otherwise} \end{cases}$$

Decoding is based on the poor resync capabilities of arithmetic codes, since the metric of sequences other than $\mathbf{X}$ tends to decrease as more symbols are processed. This works if N is high enough that the metric estimates are stable; we have experimentally found that $N \geq 100$ provides adequate performance.

III. Results

We consider a communication system that employs a DAC and a soft joint decoder, with no feedback channel; at the decoder, pruning is performed using the M-algorithm [8], with M=2048. The side information is obtained by sending the sequence $\mathbf{X}$ through a binary symmetric channel with transition probability p, which measures the correlation between the two sources. The performance of the proposed system is compared with that of a system where the DAC and its decoder are replaced by a punctured turbo code similar to [3]; we use turbo codes with rate-$\frac{1}{2}$ generator (17,15) octal (8 states) and (31,27) octal (16 states), and employ random and S-random interleavers, and 15 decoder iterations. We consider the case of symmetric ($p_0 = p_1 = 0.5$) and asymmetric source ($p_0 = 0.9$ and $p_1 = 0.1$); in this case, as an improvement with respect to [5], the turbo decoder has been modified by adding to the decoder metric the *a priori* term as in [9]. Block sizes $N = 200$ and $N = 1000$ have been considered (with S-random interleaver spread of 11 and 25 respectively).

Fig. 1 reports the results for the symmetric source case, in the same format as [1]; the abscissa $H(X|Y)$ is such that the higher p, the less correlated the sources, and hence the higher $H(X|Y)$. The performance is measured in terms of the residual bit-error rate (BER) after decoding, which is akin to the distortion in the Wyner-Ziv binary coding problem with Hamming distortion metric. Both the DAC and the turbo code generate a description of $\mathbf{X}$ of size $N/2$ bits, achieving a 2:1 compression ratio. Hence, the reported results can be interpreted also in terms of the rate penalty of the DAC and the turbo code with respect to the S-W limit (0.5 bps) in order to achieve lossless decoding.

As can be seen, the performance of the DAC is little dependent on the block size. For $N = 200$, the proposed system outperforms all 8-states and 16-states turbo codes using either random or S-random interleaving. For block length $N = 1000$, the DAC performs better than all 8-states turbo codes, and is equivalent to the 16-states code. For $H(X|Y) \geq 0.35$, a result not reported in the figure above shows that the DAC with $N = 200$ and $N = 1000$ also outperform the 8-state turbo-coder with $N = 5000$. Interestingly, the DAC performance gain is larger for small $H(X|Y)$, i.e. very correlated sources; we have found this behavior to be consistent also at other compression ratios and degrees of correlation. In terms of rate penalty, setting a residual BER threshold of 10^{-4}, for $N = 200$ the DAC is 0.25 bps away from the S-W limit, while the best 16-state turbo code simulated in this paper is 0.35 bps away; for $N = 1000$ the best 8-state turbo code is 0.30 bps away.

A drawback of DAC is that the decoding process is a suboptimal sequential search; the value of M required for near-optimal decoding grows with N and p. Having fixed $M = 2048$ is the reason why in Fig. 1 the DAC with $N = 200$ outperforms $N = 1000$. Turbo codes can easily manage large block sizes, for which their performance is significantly better.

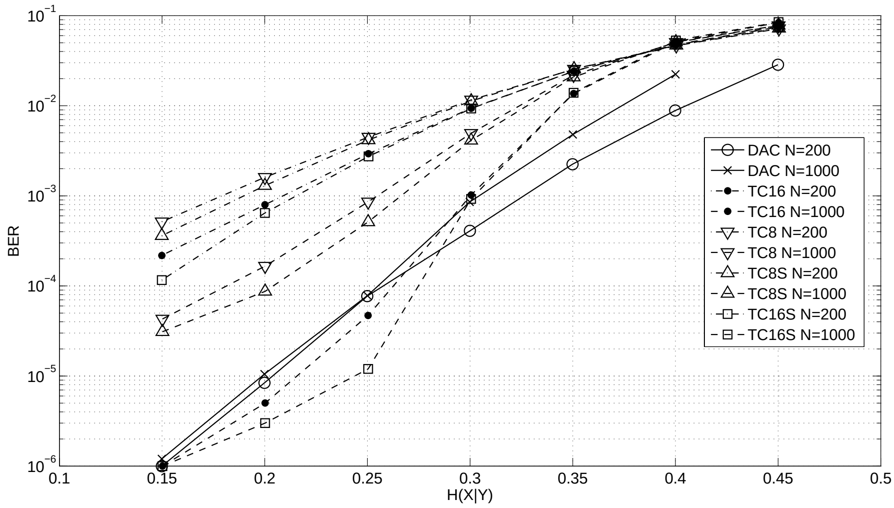

Fig. 1. Performance comparison of data communication systems: distributed arithmetic coding versus turbo coding, symmetric source ($p_0 = 0.5$). DAC: distributed arithmetic coding; TC8 and TC16: 8- and 16-state turbo code with uniform random interleaver; TC8S and TC16S: 8- and 16-state turbo code with S-random interleaver.

The results for an asymmetric source are reported in Fig. 2 for $p_0 = 0.9$. In this setting, we select various values of $H(X,Y)$, and we encode X at rate $1 - H(Y)$, so that the total rate budget is 1 bps. All turbo codes considered in this work perform rather poorly. This is consistent with the results in [10], where it is shown that the best turbo codes for symmetric sources do not perform well on asymmetric ones. The DAC has good performance, as it is designed to work with unbalanced distributions; for $N = 200$, with a threshold of 10^{-5}, it is only about 0.14 bps away from the S-W bound. For $N = 1000$, even more evidently than in Fig. 1, the DAC performs worse than for $N = 200$

when p is high, because the fixed value of M makes the decoder even more suboptimal. Turbo codes can be improved as in [10] for asymmetric distributions, by properly optimizing the generator code, and compressing the resulting S-W codewords. The results on symmetric and asymmetric sources for the DAC are expected to be even better if the sources are non stationary and a context-based probability model is employed.

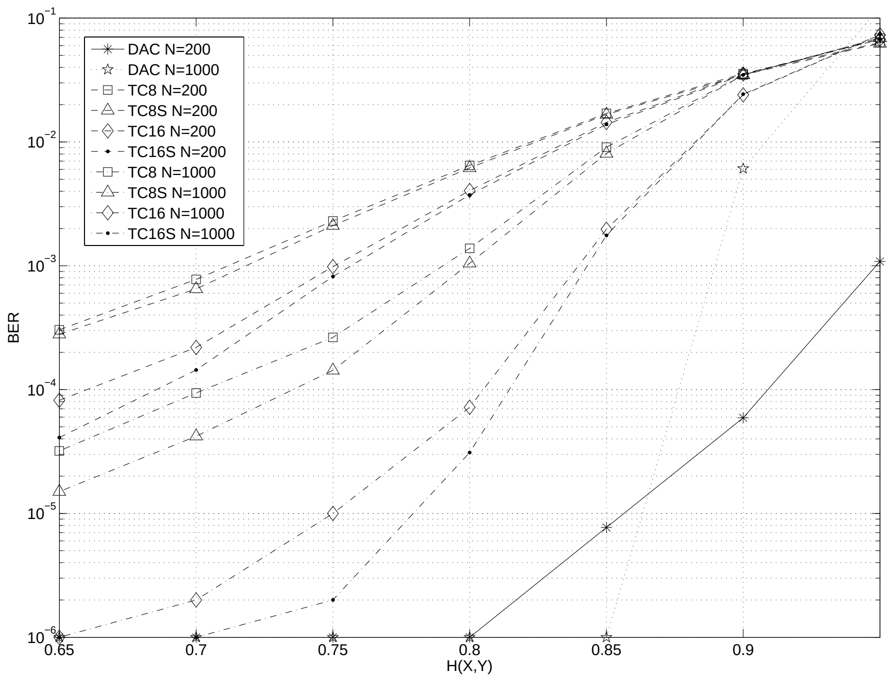

Fig. 2. Performance comparison of data communication systems: distributed arithmetic coding versus turbo coding, asymmetric source ($p_0 = 0.9$). DAC: distributed arithmetic coding; TC8 and TC16: 8- and 16-state turbo code with uniform random interleaver; TC8S and TC16S: 8- and 16-state turbo code with S-random interleaver.

We also considered the case of extremely correlated sources, for which high-rate channel codes are needed. These sources are a good model for the most significant bit-planes of several multimedia signals; due to the inefficiency of syndrome-based coders, practical schemes often assume that no DSC is carried out on those bit-planes, e.g. they are not transmitted, and at the decoder they are directly replaced by the side information [2].

The results in the case of 10:1 compression ratio, symmetric source and $N = 200$ are reported in Tab. I for the DAC and the 16-state turbo code. As can be seen, the DAC provides good performance, and for $H(X|Y) = 0.001$ in our simulation it corrected all errors; the turbo code would need to be optimized for high rate.

TABLE I

RESIDUAL BER IN CASE OF VERY CORRELATED SOURCES - $N = 200$, $p_0 = 0.5$ AND 10:1 COMPRESSION RATIO

$H(X\|Y)$	DAC	TC16	TC16S
0.1	$1.11 \cdot 10^{-2}$	$1.29 \cdot 10^{-2}$	$1.29 \cdot 10^{-2}$
0.01	$1.85 \cdot 10^{-5}$	$8.3 \cdot 10^{-4}$	$8.37 \cdot 10^{-4}$
0.001	$< 1 \cdot 10^{-6}$	$5.2 \cdot 10^{-5}$	$5.2 \cdot 10^{-5}$

IV. DISCUSSION AND CONCLUSIONS

We have proposed DAC as an alternative to existing DSC coders based on channel coding.

DAC achieves good compression performance, comparable with or better than that of turbo codes at small and medium block lengths; this is very important in many applications, e.g. in the multimedia field, where the encoder partitions the compressed file into small units (e.g. packets in JPEG 2000, slices and NALUs in H.264/AVC) that have to be coded independently. For an equal level of correlation and a given compression ratio, the DAC yields a smaller residual BER; this translates into less rate penalty for lossless compression. DAC encoding complexity is linear with the block size, and generally smaller than turbo codes, while decoding complexity is higher; hence, it is suitable for DSC applications. Interestingly, DAC works well for sources with unbalanced probability distributions; moreover, it can be extended to symmetric DSC, although this goes beyond the scope of this letter.

A major advantage of DAC is that it can exploit on-the-fly context-based probability estimation; this is a significant advantage over DSC coders based on channel codes. Turbo decoding can be modified to handle e.g. hidden markov sources [11]; however, this slightly complicates the decoder. Arithmetic coding provides a more natural and general framework for exploiting source prior probabilities.

Moreover, since the encoding is a simple extension of the arithmetic coding process, it is straightforward to extend an existing compression scheme in order to provide DSC functionalities.

REFERENCES

[1] Z. Xiong, A.D. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Processing Magazine*, vol. 21, no. 5, pp. 80–94, Sept. 2004.

[2] R. Puri and K. Ramchandran, "PRISM: a "reversed" multimedia coding paradigm," in *Proc. of IEEE International Conference on Image Processing*, 2003.

[3] J. Garcia-Frias and Y. Zhao, "Compression of correlated binary sources using turbo codes," *IEEE Communications Letters*, vol. 5, no. 10, pp. 417–419, Oct. 2001.

[4] A. 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, Oct. 2002.

[5] B. Girod, A. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proceedings of the IEEE*, vol. Special Issue on Advances in Video Coding and Delivery, no. 1, pp. 71–83, Jan. 2005.

[6] M. Grangetto, P. Cosman, and G. Olmo, "Joint source/channel coding and MAP decoding of arithmetic codes," *IEEE Transactions on Communications*, vol. 53, no. 6, pp. 1007–1016, June 2005.

[7] Q. Zhao and M. Effros, "Lossless and near-lossless source coding for multiple access networks," *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 112–128, Jan. 2003.

[8] J.B. Anderson and S. Mohan, *Source and Channel Coding*, Kluwer, 1991.

[9] J. Garcia-Frias and Y. Zhao, "Compression of binary memoryless sources using punctured turbo codes," *IEEE Communications Letters*, vol. 6, no. 9, pp. 394–396, Sept. 2002.

[10] J. Li, Z. Tu, and R.S. Blum, "Slepian-Wolf coding for nonuniform sources using turbo codes," in *Proceedings of IEEE Data Compression Conference*, 2004.

[11] J. Garcia-Frias and J.D. Villasenor, "Joint turbo decoding and estimation of hidden Markov sources," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 9, pp. 1671–1679, Sept. 2001.