INCREMENTAL AND DECREMENTAL REDUNDANCY IN TURBO SOURCE-CHANNEL CODING

Joachim Hagenauer, Nicolas Dütsch, Joao Barros and Andrew Schaefer

Institute for Communications Engineering (LNT)
Munich University of Technology (TUM)
D-80290 München, Germany
Email: {hag,nicolas,joao,andrew}@lnt.ei.tum.de

ABSTRACT

For joint source channel coding we extend the concept of incremental redundancy to a combination of decremental and incremental redundancy in a parallel concatenated Turbo scheme. Even for compression only the new scheme outperforms Lempel-Ziv compression, but has many advantages in adaptive type II FEC/ARQ systems for the transmission over noisy channels.

1. INTRODUCTION

A series of research breakthroughs in the field of coding of have rendered a large number of coding schemes which can achieve near-to-optimal performance with reasonable complexity. The secret of their success lies in the use of a "turbo" feedback in an iterative decoding scheme, an ingenious approach we call the "turbo principle" [1].

More recently, the turbo principle has been applied successfully to data compression of binary memoryless sources [3]. While in channel coding the objective is to improve the error correction after transmission over a channel and therefore all encoded bits are useful, in the dual source coding problem addressed in [2], [3] there is initially no channel involved and so the encoded bits can be punctured heavily depending on the desired compression rate. An alternative scheme based on LDPC codes and belief propagation decoding was presented in [4]. We use the punctured codes not only by the compression part, but also for the transmission over a noisy channel.

The approach we propose is dual to a hybrid automatic repeat request (ARQ Type II) scheme with forward error correction (FEC), in which parity bits are transmitted successively (*incremental redundancy*) to build up a code which is powerful enough to correct all errors introduced by the communications channel. The duality between incremental redundancy channel coding and our proposed decremental redundancy source coding scheme is illustrated in Table 1. For incremental redundancy one can use rate-compatible punctured convolutional (RCPC) codes, which result from periodic puncturing of low-rate convolutional codes. The encoder typically processes blocks of 100 to 1000 information bits and uses a puncturing pattern with period 8 and rates with incremental redundancy, e.g. $8/9, 8/10, \ldots, 8/24$.

For turbo source coding we require block lengths of about $N = 10^4 \ldots 10^6$ and finer puncturing steps are needed than those achieved with periodic puncturing with period 8. For example, if we puncture every 4th and then every 8th bit of a block, we get a compression rate of first $R = 0.875$ and then $R = 0.750$. This corresponds to a puncturing step of 0.125, which is too coarse for fine-tuning our turbo source encoder at compression rates close to entropy.

2. THE TURBO COMPRESSION ALGORITHM

Consider a block of source symbols U^N. Since the entropy of the source is $H(U)$, the source block can be perfectly reconstructed from a binary sequence X^K of length $K \simeq NH(U)$ for sufficiently large N (source coding theorem). Our task is to design a mapping from the source to the compressed sequence. We assume a binary source ($L = 2$).

The motivation for using "turbo codes" for source compression lies in the duality of source and channel coding. It follows that a good channel code is likely to be a good source code. Since "turbo codes" are well known to be good channel codes i.e. they provide a good solution to the channel coding problem an can approach the capacity limit by less than 0.5 dB. It is just a natural step to apply these to source coding problems. We now proceed to discuss the question of "how".

As stated above, we wish to generate a binary sequence X^K from the source U^N. It is possible to do this as shown in Fig. 1. We pass each of the sequences U^N and $\Pi(U^N)$ (Π defines the interleaver function), through rate 1 convolutional codes (or scramblers) with feedback to generate the parity sequences P_1^N and P_2^N respectively. As opposed to parallel concatenation in channel coding, we discard the

0-7803-8379-6/04/$20.00

Table 1. Duality between Incremental Redundancy (IR) Channel Coding (ARQ Type II) and Decremental Redundancy (DR) Source Coding

	IR-Channel Coding	DR-Source Coding
Redundancy Handling	Increment redundancy at each transmission, decoding trials at the receiver	Decrement redundancy at each compression, decoding trials at the transmitter
A priori information at both sides	Sequence of Puncturing Tables	Sequence of Puncturing Tables
Required State Information	Channel state information L_c for the channel decoder	Source state information L_p for the source decoder
Side Information	Successful decoding flag (ARQ) transmitted to sender	Successful encoding rate (parity index i) transmitted to decoder
Error Detection	CRC	Comparison to data sequence
Figure of Merit	Average throughput	Average compression
System Failure	if CRC fails	if decodability test at compressor side fails
Main Complexity	at the decoder	at the encoder
Codes	coarsely punctured RCPC codes or RCP Turbo Codes	finely punctured parallel concatenated systematic rate 1 convolutional codes
Decoder	A-priori Viterbi Algorithm or APP decoder	APP decoder
Extensions	less redundancy if source statistics are known	more redundancy if channel errors are expected

source bits.

We now have the sequences P_1^N and P_2^N, which when put together are of length $2N$ meaning that we do not yet have any compression. We achieve compression by puncturing the parity bits. If we opt for a fixed but randomly chosen puncturing scheme then the puncturing operation is analogous to transmitting the parity sequences over a binary erasure channel (BEC), whose "erasure probability" can be adjusted to achieve the required compression rate. We then rely on the turbo decoder to reconstruct the original message after transmission over the dual noisy channel (in this case our BEC). If ϵ represents the proportion of erased bits, then we have

$$\epsilon \approx 1 - \frac{H(U)}{2}. \tag{1}$$

Thus the sequence X will be of length approximately $K = NH(U)$ and we have compressed our sequence close to its entropy. We now address the problem of reconstructing U^N from X^K.

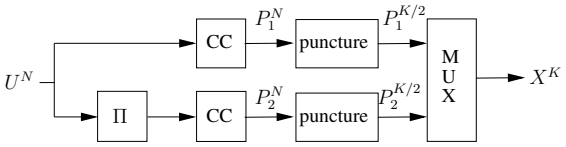

Fig. 1. A parallel concatenation of rate 1 convolutional codes (CC) with an interleaver between them followed by heavy puncturing can be used to perform data compression of binary sources.

At first glance, random puncturing of bits does not seem to be a very sophisticated way of performing source compression. However, just as in channel coding with turbo codes, the sophisticated part lies in the use of a turbo decoder. The turbo decoder for our system is depicted in Fig. 2.

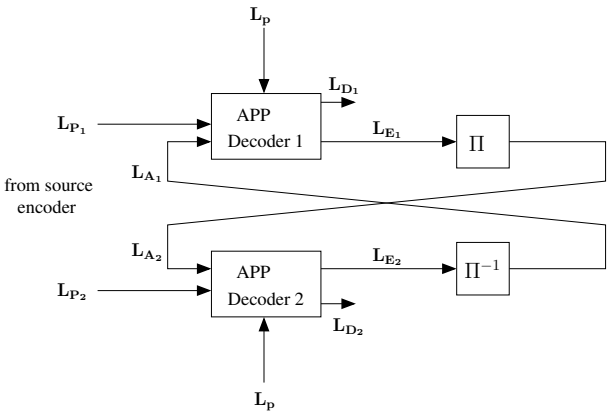

Fig. 2. Parallel turbo source decoder.

We will use L-values to describe the values input and passed around by the turbo decoder. Given a binary random variable U, where for convenience $u \in \{+1, -1\}$, then the

L-value is defined as,

$$L(U) = \log_e \frac{P(u=+1)}{P(u=-1)}. \quad (2)$$

In Fig. 2, $\mathbf{L_{P_1}}$, $\mathbf{L_{P_2}}$, $\mathbf{L_{D_1}}$, $\mathbf{L_{D_2}}$, $\mathbf{L_{A_1}}$, $\mathbf{L_{A_2}}$, $\mathbf{L_{E_1}}$ and $\mathbf{L_{E_2}}$ represent sequences of L-values. The letters A and E stand for "apriori" and "extrinsic" respectively. Since the parity bit sequences $P_1^{K/2}$ and $P_2^{K/2}$ can be thought of as being "transmitted" through a binary erasure channel (i.e. punctured), the corresponding input sequences $\mathbf{L_{P_1}}$ and $\mathbf{L_{P_2}}$ take on the L-values $\pm\infty$ (perfectly known bits) or 0 (erased).
In the case of a non-uniform binary source with $P(U = +1) = p$ and $P(U = -1) = 1 - p$, for which $H(U) = H_b(p)$ (i.e. the binary entropy function), each decoder has an additional input vector $\mathbf{L_p}$, where each element of the vector is equal to $L_p = \log_e\left(\frac{p}{1-p}\right)$. We call this 'source' apriori knowledge and distinguish between apriori knowledge which is learnt during the iterations of the decoder which we will call 'learnt' apriori knowledge. In the case of more sophisticated (Markov) source models our source apriori vector would contain different sets of L_p values. The decoding algorithm is carried out according to Fig. 2. APP decoder 1 uses $\mathbf{L_{P_1}}$ and $\mathbf{L_{A_1}}$ (initialized to zero in the first iteration) to calculate the extrinsic L-values $\mathbf{L_{E_1}} = \mathbf{L_{D_1}} - \mathbf{L_{A_1}} - \mathbf{L_p}$. The sequence $\mathbf{L_{E_1}}$ is interleaved and passed onto APP decoder 2 as "learnt" apriori knowledge $\mathbf{L_{A_2}}$. Decoder 2 performs analogous operations and the process is then "iterated" until convergence is achieved.

2.1. Decremental redundancy

To guarantee that the decoder is able to decode the input sequence without errors (lossless source coding), we can enable the encoder to test the decodability of its output. Since we want to achieve as much compression as possible, we puncture the parity bits on a step by step basis (*decremental redundancy*) as long as the compressed block can still be decoded error free by the test decoder at the source compression side. We propose the decremental redundancy scheme illustrated in Fig. 3. In a first step the parity bits in each block P_i^N with $i = 1, 2$ are written line by line in a matrix $N_c \times N_c$, such that $N = N_c^2$. We call each column of the matrix a parity segment and index it by $i \in \{1, \ldots, N_c\}$. To gradually remove the redundancy in the encoded stream we can eliminate one such parity segment at a time (in parallel for P_1^N and P_2^N), yielding a puncturing step of $2N_c/N$ that allows us to fine tune the compression rate. To make sure that the erased bits are spread out in the block we interleave the parity bits before puncturing them. Notice that we can use the same interleaver as for the turbo code, i.e.

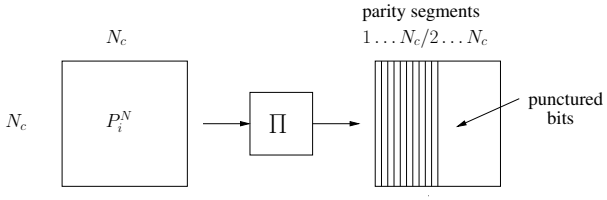

Fig. 3. Puncturing scheme for decremental redundancy.

Π and Π^{-1} for P_1^N and P_2^N, respectively, thus avoiding the storage of extra interleavers for the parity bits.

Since the decoder on the receive side knows the block length, the random puncturing matrix, and the interleaver, all it requires is the index of the last parity segment that has been erased. Since at least half the parity bits are punctured (otherwise we would have no compression at all), it suffices to use $\lceil \log_2(N_c/2) \rceil$ bits as side information (e.g. if $N = 10^4$ bits, then we get a puncturing rate step of 0.02 and require only 7 bits of rate side information per encoded block).

2.2. Incremental redundancy

If the channel which transmits the compressed bits is error free our source bits can be exactly (lossless) recovered, because the receiving side works with the same turbo decoding algorithm as the transmitting side and has the same side information . This is not true if the channel is noiseless. Then we propose the following scheme:

1) Compress the source as described in section 2.1.

2) Add a CRC to the compressed data.

3) Transmit compressed data with CRC, rate and L_P side information over the noisy channel.

4) Decode with the turbo decoder.

5) Check the integrity of the data with the CRC. If OK go to the end. If not transmit incremental redundancy which was previously punctured at the transmit size and goto 4).

6) End with error free decoded data.

3. EXAMPLES

To underline the effectiveness of the proposed source coding tools, we present a set of numerical results, which further illustrate the concept of decremental redundancy and give an idea of the performance one can expect from parallel concatenated turbo source codes. As outlined in the decremental redundancy algorithm we successively remove

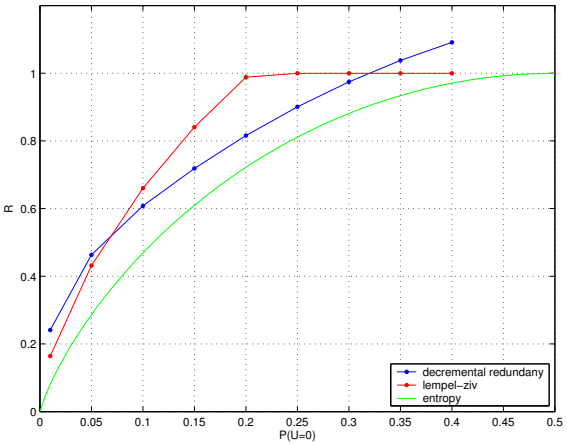

Fig. 4. Average compression rate for the decremental redundancy algorithm compared to Lempel-Ziv and entropy, block size 10000 (100×100)

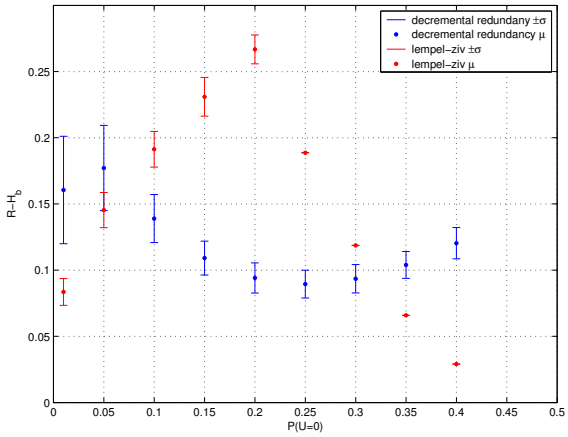

Fig. 5. Compression rate distances to entropy for the decremental redundancy algorithm compared to Lempel-Ziv, shown are average and std. deviations of the rate distances, block size 10000 (100×100)

parity segments, allowing the compression rate to gradually approach the entropy.

The performance of the of the proposed turbo source coding scheme is shown in Figures 4 and 5. A comparison with the wide spread Lempel-Ziv 1977 algorithm (universal source coding) shows that for the same block length turbo source coding yields lower compression rates, i.e. closer to the entropy of the source in the entropy range of 0.4 to 0.9 corresponding to a source probability of 0.06 to 0.32. At source probabilities around 0.2 the distance to entropy is more than halved. This gain of our scheme vs Lempel-Ziv is more pronounced at smaller block sizes.

4. CONCLUSIONS

We have presented a turbo source coding scheme, which guarantees lossless recovery of the source information through a decremental redundancy loop in the transmitter. The convergence of the turbo source decoder can be analyzed using a modified EXIT chart as shown in [5] and [6], which becomes a powerful tool to optimize the component codes and the puncturing rates near the entropy of the source. As indicated above, we can extend the work to Markov sources and noisy channels with incremental redundancy and compare it with other compression and FEC algorithm.

5. REFERENCES

[1] J. Hagenauer, "The turbo principle - tutorial introduction and state of the art," in *Proc. Int. Symposium on Turbo Codes & Related Topics*, (Brest, France), pp. 1–11, 1997.

[2] J. Garcia-Frias and Y. Zhao, "Compression of Correlated Binary Sources Using Turbo Codes," *IEEE Communications Letters*, pp. 417–419, 2001.

[3] J. Garcia-Frias and Y. Zhao, "Compression of Binary Memoryles Sources Using Punctured Turbo Codes," *IEEE Communications Letters*, pp. 394–396, 2002.

[4] G. Caire, S. Shamai, and S. Verdu, "A new data compression algorithm for sources with memory based on error correcting codes," in *Proc. IEEE Information Theory Workshop*, (La Sorbonne, Paris, France), pp. 291–295, March 31 - April 4 2003.

[5] S. Brink, "Design of serially concatenated codes based on iterative decoding convergence," in *Proc. 2nd Int'l Symp. on Turbo codes & Related topics*, (Brest, France), pp. 319–322, September 2000.

[6] Hagenauer, J., Barros, J, Schaefer, A: Lossless Turbo Source coding with decremental redundancy. - In: *5th ITG Conference Source and Channel Coding*, Erlangen, 14.-16.01.2004, ITG-Fachbericht 181, VDE-Verlag, Berlin, p. 333 - 339.

.