Reconstruction of Punctured Convolutional Encoders

Eric Filiol

Ecoles militaires de Coëtquidan
DGER/CRESCC/DSI
56381 Guer Cedex, FRANCE
efiliol@mailhost.esm-stcyr.terre.defense.gouv.fr

INRIA - Projet Codes
Domaine de Voluceau B.P. 105
78153 Le Chesnay cedex, FRANCE
Eric.Filiol@inria.fr

Abstract
This paper presents results on an operational reconstruction technique of punctured convolutional encoders. With only intercepted coded messages and the assumption that we deal with such kind of encoder, this technique recovers all the parameters of the encoder, allowing complete decoding. The technique works very well even for noisy channels. Experimental results are presented.

1. Introduction

Punctured convolutional codes were introduced by Cain *et al.* [2] as a means of greatly simplifying both Viterbi and sequential decoding of high rate convolutional codes at the expanse of a relatively small performance penalty.

A punctured convolutional code $\mathcal{C}$ is obtained by periodically deleting output symbols from a (base) (n, k, m)-convolutional code $\mathcal{C}_b$. Output symbols from $\mathcal{C}_b$ are deleted according to a periodic puncturing pattern (or perforation pattern) which can be described by its punctured matrix:

$$P = \begin{bmatrix} p_{1,1} & \cdots & p_{1,M} \\ \vdots & & \vdots \\ p_{n,1} & \cdots & p_{n,M} \end{bmatrix}$$

A very important problem is that of the reconstruction of such codes. In an attack context, a monitor wants to have access to the transmitted information (*the message*) without any knowledge on the encoder which produces the intercepted stream (*the coded sequence*). The only way is to reconstruct the encoder, that is to say to recover all its parameters. A simple decoding then gives access to the message.

This paper presents the implementation results of such a reconstruction for punctured convolutional codes of any rate, using the technique presented in [4]. To be close to a real context, we fixed the following critical limitations:

- the available coded sequence can be short (about few kbits).
- the beginning or/and the end of the coded sequence can be missing.
- We cannot use in any way the message.

In Section 2, useful notation for convolutional codes are given as well as a short description of the basic reconstruction technique of [4] we use for the punctured case. Section 3 presents the reconstruction results for punctured codes in noisy channels.

2. Basic Notation

2.1. Convolutional codes

A convolutional encoder can be seen as an encoding system (based on a set of k shift-registers without feedback) such that, at each time instant, k information digits enter the encoder (one per register). Each information digit remains in the encoder for K time units and may affect each output during that time. The constant K is the constraint length or the *memory* of the encoder.

At each time instant, n information digits are output, each of them resulting from the xor of k digits produced by the action of n polynomials on each register. The encoder is thus said to be of rate $\frac{k}{n}$. The action of the kn polynomials and shifts are easily described by polynomial multiplications. So polynomials will be used to represent the different streams.

A message will be composed of k interlaced input streams, each of them represented by a polynomial of degree $N + t$ noted $a_i(x)$, $i = 1, \ldots, k$. The kn polynomials are of degree N (hence $N = K - 1$) and will be noted $f_{i,j}(x)$. Then the encoder produces n output streams (of length t) represented as polynomials of degree t, $c_j(x)$, $j = 1, \dots, n$ and we have:

$$\sum_{i=1}^{k} a_i(x) f_{i,j}(x) = u_{j,1}(x) + x^N c_j(x) + x^{N+t} u_{j,2}(x) \tag{1}$$

The polynomials $u_{j,1}(x)$ (resp. $u_{j,2}$) (the filling (resp. the emptying) of the registers) are of degree at most $N-1$. Then the coded sequence is composed of the n interlaced output streams. Finally the puncturing pattern P is applied on the output streams.

Thus the parameters (to be recovered) of a convolutionnal encoder are: k and n defining the rate and the number of polynomials, K the constraint length, and the kn polynomials $f_{i,j}(x)$ of degree $N = K - 1$.

Generally, n and k are small integers with $k < n$. The most frequent case is $k = n - 1$. On the contrary, N must be made large enough to achieve low error probabilities. The symbols are usually elements of $GF(2)$ but generalization to $GF(q)$ where q is some prime power ($q = p^m$ for some positive integer m) can be easily done. Without loss of generality, we will only consider the case $q = 2$ (for details and generalization see [4, 5]).

2.2. The basic reconstruction technique

We now present a short description of the basic reconstruction technique of (nonpunctured) convolutional codes for $n = 2$ and $k = 1$ (for details see [4]). The encoding scheme can be described by the following polynomial equations:

$$\begin{aligned} a(x)f_1(x) &= u_{1,1}(x) + x^N c_1(x) + x^{N+t} u_{1,2}(x) \\ a(x)f_2(x) &= u_{2,1}(x) + x^N c_2(x) + x^{N+t} u_{2,2}(x) \end{aligned}$$

We don't use in any way $a(x)$ (the message), it must be eliminated. So:

$$\begin{aligned} a(x)f_1(x)f_2(x) &= u_{1,1}(x)f_2(x) + x^N c_1(x)f_2(x) \\ &\quad + x^{N+t} u_{1,2}(x) f_2(x) \\ a(x)f_2(x)f_1(x) &= u_{2,1}(x)f_1(x) + x^N c_2(x)f_1(x) \\ &\quad + x^{N+t} u_{2,2}(x).f_1(x) \end{aligned}$$

or equivalently:

$$\begin{aligned} & u_{1,1}(x)f_2(x) + x^N(c_1(x)f_2(x) + c_2(x)f_1(x)) \\ & + u_{2,1}(x)f_1(x) + x^N(c_1(x)f_2(x) + c_2(x)f_1(x)) \\ & \qquad + x^{N+t}(u_{1,2}(x)f_2(x) + u_{2,2}(x)f_1(x)) = 0 \end{aligned}$$

By identification, all the coefficients of power of x between N and $N + t$ in the polynomial $c_1(x)f_2(x) + c_2(x)f_1(x)$ are all equal to zero. In other words, we have to solve the homogenous linear system: $C\underline{f} = \underline{0}$ where $\underline{0}$ is the $(t - N)$-null vector and $\underline{f}$ is the $(2K)$-vector:

$$(f_{1,0}, f_{1,1}, f_{1,2}, \dots, f_{1,N}, f_{2,0}, f_{2,1}, f_{2,2}, \dots, f_{2,N})$$

The unknowns $f_{1,i}$ and $f_{2,i}$ for $i = 0, \dots, N$ represent the coefficients of the polynomials to be recovered. The $(t-N, 2K)$-matrix C is constructed from the $c_1(x)$ and $c_2(x)$ which are known (coded sequence) so we can solve the system and retrieve the two polynomials.

For a non-trivial solution ($f_i(x) \neq 0$) the rank of C must be at most $2K - 1$ that is to say we must have (noiseless channel):

$$t \geq 3N + 1 = 3(K-1) + 1 = 3K - 2 = L_{\text{min}}$$

For any other encoder of rate $\frac{k}{n}$, a recurring method generalizes the previous technique. First we limit ourself to the rate $\frac{k-1}{k}$ since it is always possible to consider this case. We thus will deal with $\binom{n}{k+1}$ "subencoders". A simple coincidence checking on the common polynomials allows to test the consistency of the final solutions.

So let us consider the rate $\frac{n-1}{n}$ case. Such an encoder is defined by the n equations (1):

$$\sum_{i=1}^{k} a_i(x) f_{i,j}(x) = u_{j,1}(x) + x^N c_j(x) + x^{N+t} u_{j,2}(x)$$

for $i = 1, \dots, n-1$ and $j = 1, \dots, n$. Since we cannot use in any way the different $a_i(x)$, we must eliminate them successively. As described in section 2.2, we (arbitrarily) choose first to eliminate $a_1(x)$.

Then we obtain $(n-1)$ new equations involving only $(n-2)$ message polynomials $a_i(x)$ for $i \neq 1$. That is to say we are now facing the rate $\frac{n-2}{n-1}$ case. By repeating this, we finally deal with the rate $\frac{1}{2}$ case which is easily solved. Simple polynomials operations allows to recover very easily the constituent polynomials. Detailed example can be found in [5]. Finally we need $L_{\text{min}} = (n+1)((2^{n-1} - 1)N + 1) - 2$ bits [4].

3. The Reconstruction of Punctured Codes

Let us consider a (n, k, K)-(base) convolutional code $\mathcal{C}_b$. A given puncturing pattern P is a $n \times M$ $0-1$ matrix with a total of I 1's and $nM - I$ 0's where $p_{i,j} = 0$ indicates that the i-th symbol of every branch in the j-th treillis section (of the treillis diagram of $\mathcal{C}_b$) is to be deleted.

Then the original code $\mathcal{C}_b$, after being punctured with pattern P, has become a (I, kM, m)-(punctured) code [1] $\mathcal{C}$ [7, 8]. Appendix A presents an illustrative

[1]In fact, the degree of the punctured code may be less than K, but for most interesting punctured codes no degree reduction will take place

simple example. That means that we have $I \times kM$ polynomials to recover. We then generalized the reconstruction technique of [5] by recursively eliminating the kM input (message) streams as in Section 2.2. We obtain as a result a product of polynomials that we easily factored to get the $I \times kM$ polynomials.

In communications corrupted by noise, the previous technique does no longer directly work. Two approaches have been considered. The first one is to work on noiseless parts of the intercepted sequence of required minimal necessary length. So the technique is repeated on successive subsequences of length L_{min}. The problem then is to deal with an interception long enough to be sure that at least one such noiseless subsequence is present.

In [4] the theory of recurrent events [3] has been used. Whilst giving acceptable results for non punctured encoders, this approach is less interesting for punctured codes. So we used the following result to get far better approximation:

Proposition 1 *[5] A sequence of length*

$$N = \frac{L_{min}}{(1-p)^{L_{min}}}$$

contains with a high probability a noiseless subsequence of length L_{min}, where p is the channel bit error probability.

The other approach was to try all the possible error patterns of a maximum given weight, according to the channel noise level. It is more time consumming but in return requires a far shorter intercepted sequence.

In a real context, noise level quite never exceeds 5 % (*bit error probability* P_b). Beyond that limit, other kind of codes are generally preferred (such as block codes). In very few cases the noise level may nearly reach 10 %.

We first worked with a bit error probability of 0.01 ($\frac{E_b}{N_0} = 2.71\ dB$ for an uncoded BPSK) to validate the approach. In a first time, we looked for a noiseless part of coded sequence S. Experimental results were better than expected by the theory since we consider a very pessimistic noise model (uniformly distributed rather than burst-error model).

In a second time, we try all the possible error patterns of a maximum given weight, according to the noise level. It allowed to work with a far shorter intercepted sequence S^*. Once again results were far better than expected. It is interesting to note that the reconstruction is independent of the perforation pattern. Hence, no assumption need to done on this part of the encoder.

We give here parameters (Table 1) we choose for experiments on punctured codes of rate $\frac{2}{3}$ (punctured $(2, 1, K)$ base code $\mathcal{C}_b$) and puncturing pattern

$$P = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

All computations have been performed on a 233 Mhz Linux D.E.C. Alpha Workstation with gcc. In each case the complete code was reconstructed. Experiments show that in fact shorter intercepted sequence could often be used in many practical cases.

Extended results for any other encoder of any rate can be found in [5] as well as for noise level up to 10 %.

K	S	S^*
5	307	106
10	3,084	250
15	15,246	370
20	67,445	490

Table 1: Necessary minimal length sequence for rate $\frac{2}{3}$.

Computation time required few minutes for case S and were far more important for S^*, ranging from few minutes ($K = 5$) to 3 months ($K = 20$). This is essentially not a problem since the reconstruction is done once and for all and as the expected life of the real encoder is generally greater than this computation time. More powerful computers and parallelization are bound to drastically reduce the computation time for critical applications. These latter generally use encoders with $K < 23$. Tables of the most used encoders can be found in [1, 6, 9].

Acknowledgments

I would like to thank Mr Guidoux and Captain Oddo from the DGA for their valuable comments on this work and their friendly support.

References

[1] G. Bégin, D. Haccoun, C. Paquin *Further results on high-rate punctured convolutional codes for Viterbi and sequential decoding*, IEEE Transactions on Communications, Vol. 38, No. 11, November 1990.

[2] J.B. Cain, G.C. Clark Jr., J.M. Geist, *Punctured convolutional codes of rate $\frac{n-1}{n}$ and simplified maximum likelihood decoding*, IEEE Transactions on Information Theory, vol. IT-25, No.1, pp. 97-100, January 1979.

[3] W. Feller, *An Introduction to Probability Theory*, Vol. 1, Wiley, 1966.

[4] E. Filiol, *Reconstruction of convolutional encoders over GF(q)*, Proceedings of the 6th IMA on Cryptography and Coding, M. Darnell editor, Lectures Notes in Computer Sciences Vol. 1355, Springer Verlag, 1997.

[5] E. Filiol, *Reconstruction techniques in coding theory and cryptology*, Ph. D Thesis, Ecole Polytechnique, France, 2000.

[6] D. Haccoun, G. Bégin *High-rate punctured convolutional codes for Viterbi and sequential decoding*, IEEE Transactions on Communications, Vol. 37, No. 11, November 1989.

[7] R. Johannesson, K. Sh. Zigangirov, *Fundamentals of convolutional coding*, IEEE Series on Digital and Mobile Communication, IEEE Press, 1999.

[8] R. J. McEliece, *The algebraic theory of convolutional codes*, in Handbook of Coding Theory, V. S. Pless and W. C. Huffman editors, North-Holland, 1998.

[9] Y. Yasuda, K. Kashiki, Y. Hirata *High-rate punctured codes for soft decision Viterbi decoding*, IEEE Transactions on Communications, Vol. 32, March 1984.

A. Illustrative Example

Let us take the $(2, 1, 3)$ code with polynomials

$$(1 + x^2, 1 + x + x^2)$$

The two output streams can be denoted as follows:

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & \dots \\ y_0 & y_1 & y_2 & y_3 & y_4 & y_5 & \dots \end{pmatrix}$$

By using the following puncturing pattern:

$$P = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

we then obtain the two following output streams:

$$\begin{pmatrix} x_0 & & x_2 & & x_4 & & \dots \\ y_0 & y_1 & y_2 & y_3 & y_4 & y_5 & \dots \end{pmatrix}$$

that we can rearrange as follows:

$$\begin{pmatrix} x_0 & x_2 & x_4 & \dots \\ y_0 & y_2 & y_4 & \dots \\ y_1 & y_3 & y_5 & \dots \end{pmatrix}$$

It becomes then obvious that this puncturing produces a new encoder producing three output streams.

By use of polycyclic pseudo-circulant matrices [8], the new parameters are easily defined and we have the 6 following polynomials

$$f_{1,1}(x) = 1 + x \quad f_{1,2}(x) = 1 + x \quad f_{1,3}(x) = 1$$

$$f_{2,1}(x) = 0 \quad f_{2,2}(x) = x \quad f_{2,3}(x) = 1 + x$$

where $f_{i,j}$ denotes the j-th parity-check polynomial applied on input message stream i.