Heuristics for Optimizing in Complexity SISO Trellis-Based Decoding of Linear Block Codes

A. BERTHET[(1)], J. FANG[(2)], P. TORTELIER[(1)]
[(1)] FRANCE TELECOM, CNET/DMR/IIM
38-40 rue du Général Leclerc, 92794 Issy Moulineaux, France
[(2)] ALCATEL, Space Division
5 rue Noël Pons, 92743 Nanterre, France

Résumé- Un problème important qui se pose lors du décodage des codes linéaires en blocs par l'algorithme Forward-Backward (FB) est de trouver une permutation des coordonnées qui minimise la complexité de branches des treillis associés aux codes. Dans cet article, des heuristiques sont proposées. Nous fournissons une table récapitulative des résultats obtenus pour de nombreux codes et les comparons à ceux publiés dans la littérature. L'optimisation en complexité des treillis et donc de l'algorithme FB permet d'envisager de nouveaux schémas de concaténation parallèle de codes en blocs, basés sur des codes composants plus puissants que de simples codes de Hamming (étendus) et de comparer leurs performances à celles des turbo-codes.

Abstract- An important problem in the decoding of linear block codes via the Forward-Backward (FB) algorithm is to find a coordinate permutation that minimizes trellis code complexity. In this paper, heuristics for reducing codes trellises are proposed. We give a recapitulating table of the results we have found for many codes and compare them to those published in litterature up to now. This complexity optimization allows us to introduce new schemes of parallel concatenated block codes (PCBC), based on more powerful constituent codes than simple (extended) Hamming codes and to compare their performance with turbo-codes ones.

1. Introduction

Quite by essence, iterative decoding of any combination of convolutional or block codes needs some basic decoding device, called Soft-In-Soft-Out (SISO) decoder, able to accept and to deliver A Posterior Probablities (APP) on symbols of constituent codes codewords. Indeed, iterative decoding consists in re-evaluating APP on symbols.

The Forward-Backward (FB) algorithm has been first introduced in [BAU-66]. This algorithm computes exact APP on states and transitions of any markovian process, whose all state sequences can be represented by a *trellis graph*. As pointed out by [BAH-74], convolutional codes are such time-independent markovian processes. From APP on states or transitions, APP on symbols of a given information or coded sequence can be derived respectively. As a consequence, the FB algorithm is a SISO decoder and realizes an optimal symbol by symbol MAP decoding of convolutional codes. Since linear block codes can be seen as time-dependent markovian processes and since they also accept a dynamic (non-regular) trellis representation, the FB algorithm can again be used to decode them in MAP sense.

Because of numerical representation problems, the FB algorithm which normally operates in the real domain, has to be implemented in the log-domain, thus becoming the log-FB algorithm. A simplified sub-optimal version, called Min-Log-FB algorithm also exists [ROB-95]. These three basic SISO algorithms are all trellis-oriented. It can be shown that their complexity linearly depends on the *number of edges* of the trellis on which they are applied. Hence, reducing the trellis edge complexity leads to a significant reduction in complexity of the SISO decoding of linear constituent block codes, involved in an iterative process.

2. Trellises for linear block codes

2.1 Elements of minimal trellis theory

In 1974, [BAH-74] first propose some way to represent a linear block code via a trellis graph, called *BCJR or syndrome trellis*. In 1978, [WOL-78] rediscovers the BCJR trellis. Later, [MUD-88] introduces the concept of *minimal trellis* for any linear block code, generalizing Forney's previous results on trellis diagrams [FOR-88].

Definition 1: We denote by S_i the vertex subspace at depth i (complexity $|S_i|$) and $B_{i-1,i}$ the edge subspace between depths $i-1$ and i (complexity $|B_{i-1,i}|$).

Definition 2: A trellis is minimal if and only if it simultaneously minimizes the number of vertices and the number of edges, at each depth.

Theorem 1: The minimal trellis exists and is unique for any linear block code.

Theorem 2: The BCJR trellis is minimal.

Several other methods for constructing trellises are known [FOR-88], KSC[95], which substantially differ from [BAH-74] approach. All these methods produce minimal trellises, i.e. trellises isomorphic to the BCJR trellis. In the present paper, we focus on *Kschichang-Sorokine trellises.*

Definition 3: We also define:

- the trellis vertex complexity as $|S| = \sum_{i=0}^{n} |S_i|$;
- the trellis edge complexity as $|B| = \sum_{i=1}^{n} |B_{i-1,i}|$;
- the trellis state complexity as $|E| = \max_{i\in[0,n]} |S_i|$.

2.2 The permutation problem

It turns out that a permutation of the trellis time axis (equivalent to a permutation of code symbols) can drastically change the number of vertices or edges in the minimal representation of some given code C. In fact, this permutation problem is far to be as well understood as the theory of minimal trellises and still leads to very interesting and challenging questions. Some authors strongly believe that the problem of finding an optimal trellis in sense of width (state complexity) among all possible permutations of the time axis is NP-complete. [KSC-94] have proved the NP-completeness of some strongly related problem (state complexity profile problem). Since the cardinality of the search space, i.e. the full symmetric group Ω on n elements, increases as $n!$, heuristics are the single solution to find « good » trellises.

Proposition 1: Optimization attempt under some specific trellis complexity criterion does not necessarily produce a trellis which also minimizes other trellis complexity measures.

Proof : by a counter-example. Consider the binary BCH code $(15,7,5,4)$. The optimal trellis in sense of edge complexity has parameters $|B|_{opt} = 284$ $|S| = 222$, whereas the optimal trellis in sense of vertex complexity has parameters $|S|_{opt} = 206$ $|B| = 300$ q.e.d.

Hence, in the context of minimizing the complexity of a trellis via permutations of the time axis, it is fundamental to specify which measure of complexity we attempt to optimize.

3. Heuristic search of optimal trellises

In this section, we briefly describe several heuristics we have implemented to find « good » time axis permutations. The fitness function to minimize is:

$$f(\pi C) = |E(\pi C)| = \sum_{i=1}^{n} |E_i(\pi C)|$$

over all equivalent codes πC of C.

For some given permutation pattern π, the fitness function proceeds in two steps:

Step 1: compute the Minimal Span Generator Matrix (MSGM) associated with equivalent tested code πC, using for example the « greedy algorithm II » proposed by [McE-96].

Step 2: read through the MSGM and compute the trellis edge complexity.

Clearly, step 1, whose complexity is roughly in $O(k^2)$ is the bottleneck of the heuristic. Each time code parameters satisfy:

$$n - k < k$$

heuristic is applied on dual code. Using past/future decomposition, many relationships can be exhibited between direct and dual trellis code complexity measures [McE-96], [FOR-94].

3.1 Basic descent procedure

Any basic descent procedure operates as follows:

Step 1: *(initialization)* Choose a starting point $\pi_0 C$ in the search space. Compute the fitness function for this point, and store it as the current best solution $f_{\min}$. Let N be the maximum number of permutations to test. Set $i \leftarrow 0$.

Step 2: while $(i < N)$

```
• « Permutor » procedure selects a permutation scheme $\pi_i$ ;
• $f(\pi_i C)$ is computed:
if $f(\pi_i C) < f_{\min}$
        $f_{\min} \leftarrow f(\pi_i C)$
        $\pi_{best} \leftarrow \pi_i$
$i \leftarrow i + 1$
```

Different strategies are possible for the « Permutor » process. Following [KSC-94], the permutation schemes can be chosen within a neighborhood, restricted to the set of all codes that can be reached from the given one by transposing only *two* symbol indices (or columns). If at a time, no improvement is found within the neighborhood, the heuristic search algorithm:

- can return the current best permutation scheme as a tentative local optimum;
- extend the « local » search to a bigger subspace, or the full space.

The basic descent procedure can be started from randomly chosen points and the best overall solution chosen.

3.2 Improved descent procedures

As noticed before, direct evaluation of the fitness function for each tested permutation scheme, although polynomial in complexity, is costly. We propose at least two suboptimal solutions to solve the problem, which lead to several heuristics:

- compute a simpler related, but not exact objective function;
- attempt to « learn » common charateristics associated with « good » permutations and use them to improve further permutation schemes. We introduce two parameters:

σ the number of transpositions after which heuristic begins to learn common characteristics either on permutation patterns or directly on MSGM structure;

δ the number of unavailing transpositions after which learning based constraints are released (according to a rule).

3.3 Simulated annealing method

The simulated annealing method aims at avoiding local minima of $f(\pi C)$. Instead of discarding a transposition leading to an increase of the fitness function, as in basic descent procedure, the method accepts it with some given probability, which depends on the increase amplitude. A control parameter θ, called temperature, makes unavailing transpositions less and less likely. Our simulated annealing procedure proceeds as follows:

Step 1: *(initialization)* Choose a starting point πC in the search space. $\pi_{best} \leftarrow \pi$ $f_{\min} \leftarrow f(\pi C)$. Initialize the temperature θ (several strategies).

Step 2: while (temperature θ)

Step 3: Compute a threshold κ. Set $j \leftarrow 0$

```
while ( $j < \kappa$ )
• « Permutor » procedure selects a permutation scheme $\pi'$ ;
• Compute $\Delta f = f(\pi' C) - f(\pi C)$
if $(\Delta f < 0)$ then $\pi \leftarrow \pi'$
        if $(f(\pi C) < f_{\min})$ then
                $\pi_{best} \leftarrow \pi$   $f_{\min} \leftarrow f(\pi C)$   $j \leftarrow j + 1$
else
        $p \leftarrow$ random value in $[0,1]$
        if $p \le \exp(-\Delta f / \theta)$ then $\pi \leftarrow \pi'$
$\theta \leftarrow \varphi(\theta)$ where $\varphi$ is a decreasing function. Return in step 2.
Finally, return $\pi_{best}$
```

3.4 Results

Convergence speed of each of these heuristics can be evaluated by applying them on extended Hamming codes of order m, for which we know optimal trellis complexity measures [KAS-93]:

$$\begin{cases} |S| = \left(2^{2m+1} - 9 \times 2^{m-1} + 10\right)/3 \\ |B| = \left(2^{2m+2} - 9 \times 2^{m+1} + 20\right)/3 \end{cases}$$

Improved heuristics have time search far lesser than basic descent procedure. For codes of length 32, search usually takes few minutes. Even for codes of lenth 64 or higher, search never exceeds 12 minutes on station HP-UX-9.5.

In table 1, trellis complexity measures are tabulated for various codes of length up to 64. In column « best found», we indicate the best trellis found in litterature up to now, and precise under what complexity criterion. Trellises whose profiles match the DLP lower bound, are componentwise optimal and are marked with an asterisk.

Table.1: Trellis complexity measures for various codes

Code	$\lvert B\rvert_{wolf}$	$\lvert S\rvert$	$\lvert B\rvert_{best}$	$\lvert E\rvert$	best found	Ref
Ham(15,11,3,8)	284	110	188	16	|B|=196	[LIN-97]
Ham(15,11,3,8)$^{\perp}$	172	110	124	16		
Ham(16,11,4,8)	508	150	252	16	*	[KAS-93]
Ham(16,11,4,8)$^{\perp}$	316	150	172	16	*	[KAS-93]
BCH(15,7,5,4)	636	222	284	32	|B|=284	[LIN-97]
BCH(16,7,6,4)	764	326	420	64	|B|=420	[LIN-97]
BCH(15,5,7,4)	284	134	156	16	|B|=156	[LIN97]
BCH(16,5,8,4)	316	150	172	16	*	[LIN-97]
Ham(31,26,3,16)	1468	462	860	32	|S|=462	[KSC-94]
Ham(31,26,3,16)$^{\perp}$	796	462	492	32		
Ham(31,25,4,16)	2684	606	1116	32		
Ham(31,25,4,16)$^{\perp}$	1468	606	652	32		
Ham(32,26,4,16)	2812	638	1180	32	*	[KAS-93]
Ham(32,26,4,16)$^{\perp}$	1532	638	684	32	*	[KAS-93]
BCH(31,21,5,12)	26620	5038	8028	1024	|S|=5550	[KSC-94]
BCH(31,21,5,12)$^{\perp}$	15356	5038	6060	1024	|S|=5550	[KSC-94]
BCH(31,20,6,11)	45052	7486	11900	1024		
BCH(31,20,6,11)$^{\perp}$	26620	7486	9532	1024		
BCH(32,21,6,12)	49158	9534	14972	2048	|B|=14972	[WAN-96]
BCH(32,21,6,12)$^{\perp}$	28668	9534	11580	2048		
BCH(31,16,7,8)	196604	4286	5884	512	|B|=5884	[LIN-97]
BCH(31,16,7,8)$^{\perp}$	163836	4286	5628	512		
BCH(32,16,8,8)	262140	4798	6396	512	*	[KAS-93]
JMG(31,10,12,5)	15356	5278	6300	1024	|B|=7500	[LIN-97]
BCH(31,10,12,5)	15356	5726	6748	1024	|B|=7068	[LIN-97]
BCH(31,6,15,4)	1468	606	652	32	|S|=606	[KSC-94]
BCH(32,6,16,4)	1532	638	684	32	*	[KAS-93]
BCH(33,23,3,12)$^{\perp}$	17404	5158	6180	1024		
Golay(23,12,7,8)	12284	2174	3068	512	|B|=3068	[LIN-97]
Golay(24,12,8,8)	16830	2686	3580	512	*	[FOR-88]
Ham(64,57,4,32)$^{\perp}$	6908	2638	2732	64	*	[KAS-93]
BCH(64,51,6,24)$^{\perp}$	344060	136734	144924	8192		

4. Iterative decoding of parallel product codes (PPC)

4.1 Definition of PPC

Definition 4: Let C_1 and C_2 be two linear systematic block codes over F_q of parameters $\left(n_i, k_i, d_{\min_i}\right)$ $i \in [1,2]$. Let B_{data} a $k_1 \times k_2$ data matrix. The bidimensional parallel product code is the set of all codewords :

$$B_{data} \cup B^1_{parity} \cup B^2_{parity}$$

where $\cup$ is a concatenation operator, B^1_{parity} denotes a $(n_1 - k_1) \times k_2$ parity matrix produced by encoding the k_1 columns of B_{data} via C_1 and B^2_{parity} denotes a $k_1 \times (n_2 - k_2)$ parity matrix produced by encoding the k_2 rows of B_{data} via C_2. PPC can be seen as a special case of PCBC.

PPC are linear block codes over F_q of length $n = n_1 n_2 - (n_1 - k_1)(n_2 - k_2)$, of dimension $k = k_1 k_2$ and of minimum distance $d_{\min} = d_{\min_1} + d_{\min_2} - 1$. PPC code rate is $\rho = 1/\left(n_1/k_1 + n_2/k_2 - 1\right)$.

4.2 Iterative decoding of bidimensional PPC

Horizontal step: using C_2 SISO decoder, decode successively the k_2 received words of size n_2. For each word, C_2 SISO decoder is fed with the log-likelihood ratios on received bits, produced by the demodulator, and the log a priori ratios on data bits, coming from previous C_1 decoding. After C_2 decoding, the log extrinsic ratios on data bits are computed and passed as log a priori ratios on data bits for C_1 SISO decoder.

Vertical step: using C_1 SISO decoder, decode successively the k_1 received words of size n_1. For each word, C_1 SISO decoder is fed with the log-likelihood ratios on received bits, produced by the demodulator, and the log a priori ratios on data bits, coming from previous C_2 decoding. After C_1 decoding, the log extrinsic ratios on data bits are computed and passed as log a priori ratios on data bits for C_2 SISO decoder.

The two steps constitute one full iteration of the iterative process. A recapitulative functional diagram is shown in Fig.1.

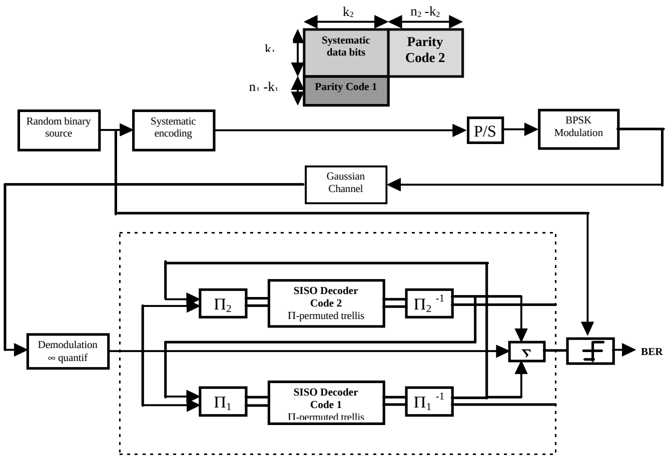

Fig.1: Iterative decoding algorithm

Only the FB and the log-FB algorithms, used as SISO decoders, deliver optimal reliabilities which really correspond to the observed bit error probabilities. The sub-optimal Min-Log-FB algorithm tends to deliver too optimistic soft values at low SNR, and therefore, the performance in interative decoding degrades (see Fig.2).

The SISO algorithms are applied onto optimized Kschichang-Sorokine trellises. When constituent codes have parameters satisfying inequality:

$$n - k < k$$

SISO decoders work onto trellises of dual codes, whose edge complexity is much smaller.

4.3 Performance and comparison with Turbo-codes

We have simulated three different PPC:

$(31{,}21)^2//\quad(24{,}12)^2//\quad(32{,}16)^2//$

A BSPK modulation is assumed. For each PPC, BER versus SNR performance on a Gaussian channel are shown after 4 iterations of the iterative process. On Fig.2 is pointed out the degradation, at low SNR, brought by the use of a sub-optimal Min-Log-FB SISO algorithm instead of a log-FB optimal SISO. This degradation reaches $0{,}5\,dB$ at $\frac{E_b}{N_0}=1{,}5\,dB$.

Taking [BAR-96] simulation results as a reference, it turns out that for equivalent rates and interleaver lengths, performance of PPC are slightly better than those of turbo-codes, which are very sensitive to interleaver gain. The well known error floor of parallel concatenation schemes cannot be avoided but it holds at lower BER for PPC. This is due to the fact that PPC have a better minimum distance that turbo-codes.

5. Conclusion

In this paper, we have described improved heuristics that have been used to find optimal coordinate permutations for various linear block codes, in sense of trellis edge complexity. This trellis complexity optimization is a crucial step as soon as iterative decoding of PPC based on FB decoders and involving more powerful components that simple Hamming codes is considered. Even for low code rates, simulation results show that PCBC can achieve similar and sometimes better performance than Turbo-Codes.

References

[BAH-74] L.R. Bahl, J. Cocke, F. Jelinek, J. Raviv, « Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate », *IEEE Trans. Inform. Theory*, Mar. 1974

[BAR-96] S. Barbulescu, « Iterative Decoding of Turbo-Codes and Other Concatenated Codes », *Doctoral Dissertation*, University of South Australia, Feb. 1996

[BAU-66] L.E. Baum, T. Petri, « Statistical Inference for Probabilistic Functions of Finite State Markov Chains », *Annals of Mathematical Statistics*, 1966

[BER-96] C. Berrou, A. Glavieux, « Near Optimal Error Correcting Coding and Decoding : Turbo-Codes », *IEEE Trans. Inform. Theory*, vol. 44, no. 10, Oct. 1996

[FOR-88] G.D. Forney, « Coset Codes - Part II : Binary Lattices and Related Codes. Annexes : Diagram Trellis », *IEEE Trans. Inform. Theory*, vol. 34, Sept. 1988

[FOR-94] G.D. Forney, « Dimension/Length Profiles and Trellis Complexity of Linear Block Codes », *IEEE Trans. Inform. Theory*, vol. 40, no. 6, Nov. 1994

[KAS-93] T. Kasami, T. Takata, T. Fujiwara, S. Lin, « On Complexity of Trellis Structure of Linear Block Codes », *IEEE Trans. Inform. Theory*, vol. 39, no. 3, May 1993

[KSC-94] F.R. Kschichang, F.R. Horn, « A Heuristic for Ordering a Linear Block Codes to Minimize Trellis State Complexity », *Proc. 32nd Annual Allerton, Conf. on Communications, Control and Computing*, Allerton Park, Illinois, Sept. 1994

[KSC-95] F.R. Kschichang, V. Sorokine, « On the Trellis Structure of Block Codes », *IEEE Trans. Inform. Theory*, vol. 41, Nov. 1995

[LIN-97] W. Lin « The Trellis Complexity of Block and Convolutional Codes », *Doctoral Dissertation*, California Institute of Technology, Pasadena, California, Feb. 1997

[McE-96] R.J. McEliece, « On the BCJR trellis for Linear Block Codes », *IEEE Trans. Inform. Theory*, vol. 42, no. 4, July 1996

[MUD-88] D.J. Muder, « Minimal Trellis for Block Codes, *IEEE Trans. Inform. Theory*, vol. 35, no. 5, Sept. 1988

[ROB-95] P. Robertson, E. Villebrun, P. Hoeher, « A Comparison of Optimal and Suboptimal MAP Decoding Algorithms operating in the Log-Domain », *Proc. Int. IEEE Conf. on Comm.* , 1995

[WAN-96] X. Wang, S.B. Wicker, « The Design and Implementation of Trellis Decoders for some BCH Codes », *submitted to IEEE Trans. Inform. Theory*, 1996

[WOL-78] J.K. Wolf, « Efficient Maximum Likelihood Decoding for Linear Block Codes, *IEEE Trans. Inform. Theory*, vol. 24, no. 1, Jan. 1978

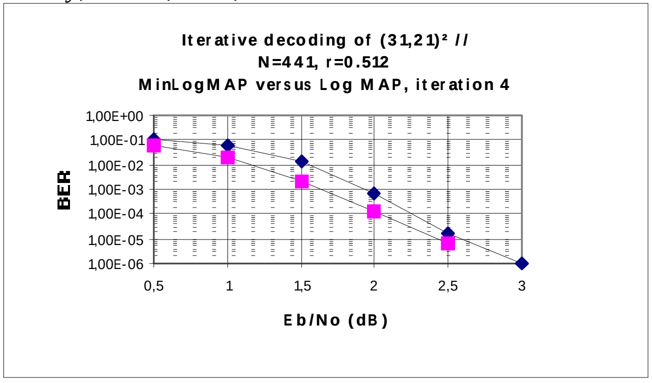

Fig.2: Comparison of SISO algorithms operating in the log-domain in case of PPC iterative decoding

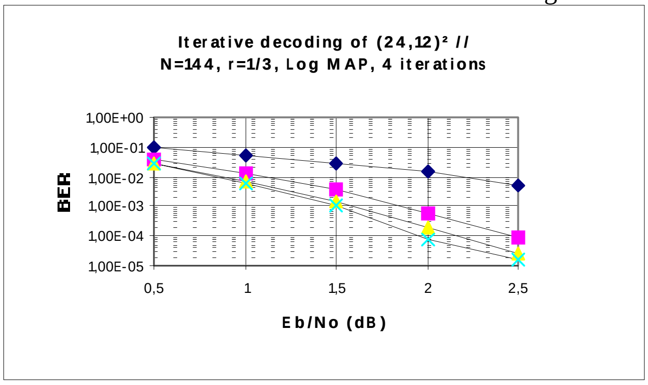

Fig.3: Performances of PPC $(24{,}12{,}8)^2$

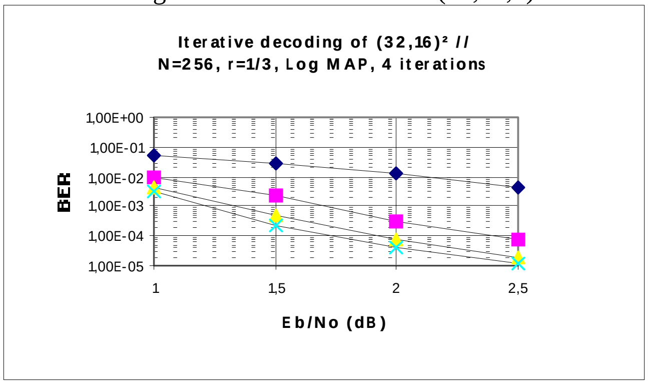

Fig.4: Performances of PPC $(32{,}16{,}8)^2$