Using algebraic structures to improve LDPC code reconstruction over a noisy channel

Pierre Loidreau

To cite this version:

Pierre Loidreau. Using algebraic structures to improve LDPC code reconstruction over a noisy channel. International Symposium on Information Theory ISIT 2019, Jul 2019, Paris, France. pp.2439-2443, 10.1109/ISIT.2019.8849756 . hal-02387819

HAL Id: hal-02387819
https://hal.science/hal-02387819

Submitted on 30 Nov 2019

Using algebraic structures to improve LDPC code reconstruction over a noisy channel

Pierre Loidreau
DGA MI and IRMAR
Email: Pierre.Loidreau@univ-rennes1.fr

***Abstract*—In this paper we show that algebraic structure of codes can be used to improve dramatically the efficiency of code reconstructions techniques especially in the case of quasi-cyclic LDPC codes of large block sizes which are widely used in standards. We focus on the case where the receiver gets noisy blocks, but the principle could be used in the case of non noisy reception. We also show that the smoother the length of the quasi-cycle is, the better the trade-off can be tuned.**

I. Introduction

We consider a framework where the communication channel between the transmitter and the receiver is partially cooperative.. For instance the context could be that of a receiver who for some or other reason has access to a sequence of noisy codewords for some partially unknown code. The sequence has been transmitted through a binary symmetric channel and the user has some partial information about the code. He knows for example the type of the family, the blocklength or the memory of the encoder and maybe the architecture upon which the code was built as could be the case in some standards. From this partial knowledge, our receiver wants to reconstruct the code and the encoder that was used for the encoding of the message so as to

1) Remove the noise from the received sequence ;
2) Recover the information vector, if for instance the encoding is systematic.

Reconstruction problems were already addressed for various classes of codes and in the general context of knowing only general information on the code and not necessarily access to information on the algebraic structure. [1]–[6].

However, when implemented in systems with limited resources, codes generally come with a strong algebraic structure. The reason is that an algebraic structure can be extremely useful to save memory and to potentially parallelize the encoding. This is especially the case for LDPC codes that one finds in standards. These codes usually come with quasi-cyclic or convolutional structures. They traditionally have large lengths of several thousands of bits. The quasi-cyclic or convolutional structure enables to perform the encoding by storing only a few rows of the parity-check matrix and using an LFSR based architecture. For a good survey of binary LDPC codes in standards, the reader can refer to [7], chapter 4 (in French). This is the reason why we focus on the so-called reconstruction of LDPC codes in a noisy environment. These codes are widely implemented in standards, versatile and have usually a strong algebraic structure. For instance the DVB-S2 standard authorizes LDPC codes with blocklengths $16\ 200$ and $64\ 800$ pseudo-randomly designed [8], with quasi-cycles of length 360.

In our setting reconstructing means finding low-weight parity-check equations or equivalently low weight codewords from the dual code. Without using the extra structure of the codes, finding low-weight parity-check equations of the code can be extremely painful. In the noisy case the most efficient techniques presented in [5], [6] have a huge complexity in memory and time even if one targets only very small weight parity-check equations. In the case of cyclic codes some attempts were made to use the structure to improve the efficiency of reconstructing the generator polynomial of the code. One guesses a candidate polynomial, computes the weight distribution of the syndromes and compares this distribution to the theoretical distribution. If there is a distortion, then the candidate polynomial is a factor of the generator polynomial, [9]–[11].

In this paper we consider a completely different approach. The idea comes from some cryptographic techniques which analyzed the security of McEliece-like cryptosystems using quasi-cyclic structures, [12], [13]. We suppose that the code is quasi-cyclic and that the receiver knows the position of the orbits under the action of the permutation group of the code. Some codewords are received after transmission through a binary symmetric channel *BSC*. From the knowledge of the orbits we can make a projection of folding in the sense of [18] which transfers the problem of reconstructing the parent LDPC code to the problem of reconstructing a code with smaller parameters but from the knowledge of noisier codewords since the folding operation also operates on the channel.

Provided the parameters are well chosen and there are enough received codewords, there is a strong relation between parity-check equations of the projected code and the original code. This idea can be iterated if the index of the quasi-cyclic code is smooth (has many factors). And in this case offers a potential trade-off between computational and data complexity and amount of data available at the receiver's end.

Since reconstructing LDPC codes reduces to finding low weight codewords in the dual code, we briefly recall the complexity of one of the most efficient technique in a noisy environment. This is also one of the simplest to analyze. In a second part, we recall the definition of quasi-cyclic codes. In a third part, we describe the so-called folding operation. We show that in the case where the code is quasi-cyclic, the dual of the folded code is also the folded of the dual code. In a fourth part, we finally explain how to exploit this property to improve the complexity of the recovery of low-weight parity-check equations.

II. Finding small weight codewords in noisy environment

At the receiver's end one has M noisy codewords $\mathbf{b}_1, \ldots, \mathbf{b}_M$ coming from the same targeted code $\mathcal{C}$ of length n and dimension k. Reconstructing $\mathcal{C}$ in this context implies finding parity-check equations. Ideally sufficiently enough to recover the whole code, but practically recovering only a few equations could provide some useful information. In [5] the author study two techniques to find parity-check equations of low Hamming weight. In the context of a noisy environment, the most efficient is the Chose-Joux-Mitton [14] (CJM) algorithm which is based on the search for collisions. The other one is based on Gaussian elimination [3], which is rather efficient whenever there are no noisy codewords, but which is not noise-proof. This approach was further improved in [6] combining the best of the two techniques by a sort of trade-off between Gaussian elimination and collision techniques. Since our goal in the paper is to show the advantage of using the algebraic structure of the targeted code in terms of data and computational complexity, to simplify our analysis we only consider the CJM algorithm.

We consider the following parameters:

1) The number of noisy available codewords $\mathbf{b}_1, \ldots, \mathbf{b}_M$ of length n is equal to M. This corresponds to the amount of data available at the receiver's end ;
2) The cross-over probability of the BSC channel is equal to $1/2 - \epsilon$;
3) The weight of the targeted parity-check equations is w.

For simplicity we assume that n and w are even numbers. Then results from [5], [7] imply that for sufficiently low false alarm one needs at least

- $M > \left(4/(2\epsilon)^w\right)^2$ codewords, that is one needs at least Mn bits ;
- If M is sufficiently large, the average computational cost in binary operations to find one parity-check equation is given by

$$C_{CF} = 2\frac{2^\ell + \binom{n/2}{w/2}}{(n-k)\binom{n/2}{w/2}(1+(2\epsilon)^w)^\ell}\binom{n}{w} \tag{1}$$

  where ℓ is a constant giving the size of the window on which collision are considered. In applications this quantity can be optimized depending on the weight of the targeted equations and the length of the code. The impact in term of list size is:

$$\mathcal{L}_{CJM} = c \cdot \binom{n}{w/2} \tag{2}$$

  where c is a small constant.

Equation (1) shows that for constant ℓ small noise (ϵ small), provided $w << n$, the complexity is proportional to $n^w/w!$, thus exponential in the size of the code.

Example 1: Here are some parameters if we suppose regular LDPC codes with rate $1/2$, parity-check equations of weight 8 over a BSC with cross-over probability $2 \cdot 10^{-3}$.

n	M_{min}	Bin. Ops.	List Size
64 800	31	2^{73}	2^{41}
16 200	218	2^{60}	2^{35}

If $\epsilon = 0$, then there exist more efficient algorithms to recover small weight codewords. However when the noise increases (ϵ decreases), then the most efficient algorithms use these kind of techniques. Thus, considering the complexities, it can be of interest to find a trade-off between the noise and the code length to improve the efficiency.

III. Quasi-cyclic codes

We consider only binary codes, although this framework can be adapted to the non-binary case. Quasi-cyclic codes can be considered as module codes over polynomial rings, [15], [16] but for our purpose it is more meaningful to describe them as being invariant codes under the action of a cyclic permutation with constant size orbits. In the following we will consider any vector $\mathbf{c} \in GF(2)^n$ where $n = Np$ under the form

$$\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_N),$$

where $i = 1, \ldots, L$ $\mathbf{c}_i = (c_{1+(i-1)p}, \ldots, c_{p+(i-1)p})$.

Definition 1 (Quasi-cyclic codes): Let $\mathcal{C} \subset GF(2)^n$, where $n = Np$. Then $\mathcal{C}$ is quasi-cyclic of index p if up to permutation of the components of the code

$$\forall(\mathbf{c}_1, \ldots, \mathbf{c}_N) \in \mathcal{C}, \text{ then } (\mathbf{c}_1 >>> 1, \ldots, \mathbf{c}_N >>> 1) \in \mathcal{C}$$

where for $i = 1, \ldots, N$, we have $\mathbf{c}_i = (c_{1+(i-1)p}, \ldots, c_{p+(i-1)p})$ and $>>> 1$ denotes the right circular permutation of the components of the vectors $\mathbf{c}_i$ of size p.

This implies in particular that the permutation group $Perm(\mathcal{C})$ is non trivial and contains a cyclic group generated by a permutation τ which is a product of cycles of size p. The group action of τ is described in figure 1.

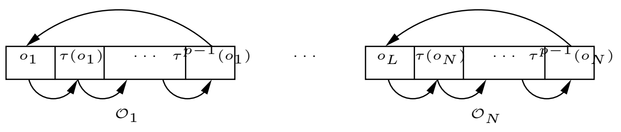

Fig. 1. Action of the generator τ of the cyclic group

It is also usual to find more algebraic description of quasi-cyclic codes as module codes over a quotient polynomial ring [15], [16]. However in our case we will use only the permutation group structure. So we will focus only on the binary representation of code vectors.

Another well-known property is that if $\mathcal{C}$ is quasi-cyclic of index p, then $\mathcal{C}^\perp$ is also quasi-cyclic of index p. Hence a quasi-cyclic code has a parity-check/generator matrix of the form

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1N} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{J1} & \mathbf{A}_{J2} & \cdots & \mathbf{A}_{JN} \end{pmatrix}, \qquad (3)$$

where $\mathbf{A}_{ij}$ are $p \times p$ binary circulant matrices. Such an algebraic is of interest in applications where limited resources is available since the encoding of the information can be implemented by clocking LFSR's of size p. This enables simple parallelization and saves memory, see [17] for instance.

IV. Projection or folding operation

The technique of projecting of folding code by considering a subcode of the invariant code and keeping only column per orbit is not new, [18]. This technique has already been recently applied to cryptanalysis of public-key encryption schemes based on MDPC/LDPC codes or LRPC codes using doubly-circulant structure, [12], [13]. However since the term of folding is more employed in cryptanalysis papers, we will use this term rather than projecting.

Let $\tau \in \mathcal{S}_n$ be a product of N cycles of size p. For $i = 1, \ldots, N$ let us denote by

$$\mathcal{O}_i = \{o_i, \tau(o_i), \ldots, \tau^{p-1}(o_i)\}.$$

Thus $\mathcal{O}_i$ corresponds to the ith cycle of size p. Since the orbits form a partition of the set $\{1, \ldots, n\}$, a vector of length n will be labeled accordingly under the form

$$\mathbf{c} = (\mathbf{c}_{\mathcal{O}_1}, \ldots, \mathbf{c}_{\mathcal{O}_N}).$$

With this notation we define the following folding operation.

Definition 2 (Folding operation): Let $n = pN$, $\tau \in \mathcal{S}_n$, and let $\mathcal{O}_i$ be the corresponding orbits of size p. Let $\mathbf{b} = (\mathbf{b}_{\mathcal{O}_1}, \ldots, \mathbf{b}_{\mathcal{O}_N}) \in GF(2)^n$. Then the folding operation relatively to τ is defined by

$$\widetilde{b} = \left(\sum_{u \in \mathcal{O}_1} b_u, \ldots, \sum_{u \in \mathcal{O}_N} b_u \right).$$

Vector $\widetilde{\mathbf{b}}$ is a binary vector of length N, and the vector $\mathbf{b}$ is its preimage. An obvious property of this operation is that it necessary decreases the Hamming weight

Proposition 1: If wt is the Hamming weight function, then

$$wt(\widetilde{\mathbf{b}}) \le wt(\mathbf{b}).$$

Proof 1: Since we have $wt(\mathbf{b}) = \sum_{i=1}^{N} wt(\mathbf{b}_{\mathcal{O}_i})$, if the ith component $\widetilde{b}_i$ of $\widetilde{\mathbf{b}}$ is non-zero, this implies necessarily that $wt(\mathbf{b}_{\mathcal{O}_i}) \ge 1$. ∎

The folding operation is naturally additive, and thus can be extended to vector spaces and codes.

Definition 3 (Folded code): Let $\tau \in \mathcal{S}_n$ and let $\mathcal{C} \subset GF(2)^n$ be a $[n, k, d]$ linear code. Then the folded code is defined by

$$\widetilde{\mathcal{C}} \overset{def}{=} \{\widetilde{\mathbf{c}} \mid \mathbf{c} \in \mathcal{C}\}$$

Note that proposition 1 does not necessarily imply that t he minimum distance of the folded code is less than that of the parent code. Namely in theory, many codewords of weight could be folded onto the all zero vector. A description of the operation can be found at figure 2.

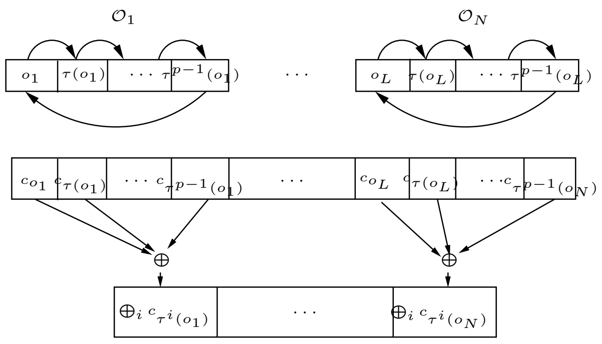
$\mathcal{O}_1$
$\mathcal{O}_N$
o_1
$\tau(o_1)$
$\cdots$ $\tau^{p-1}(o_1)$
o_L
$\tau(o_L)$
$\cdots$ $\tau^{p-1}(o_L)$
c_{o_1} $c_{\tau(o_1)}$ $\cdots$ $c_{\tau^{p-1}(o_1)}$
c_{o_L} $c_{\tau(o_L)}$ $\cdots$ $c_{\tau^{p-1}(o_N)}$
$\oplus_i c_{\tau^i(o_1)}$
$\oplus_i c_{\tau^i(o_N)}$

Fig. 2. Folding the code

In general case there is no obvious relation between a $[n, k]$-code $\mathcal{C}$ and its corresponding $[N, \widetilde{k}]$-code $\widetilde{\mathcal{C}}$ except the obvious relation relating the dimensions, $\widetilde{k} \le pk$. However whenever $\mathcal{C}$ is a quasi-cyclic code of index p this is another story and some useful properties can be derived.

Proposition 2: Let $\mathcal{C} \subset GF(2^n)$ be quasi-cyclic of index p, given by a generator matrix under the form (3). Then the dimension $\widetilde{k}$ of the projected code $\widetilde{\mathcal{C}}$ satisfies $\widetilde{k} \le J$.

Proof 2: The idea is that the $p \times n$ each submatrix $(\mathbf{A}_{i1}, \ldots, \mathbf{A}_{iN})$ of the generator matrix of the code $\mathcal{C}$ projects onto a $p \times N$ matrices of rank 1. See [13], [18] for more details.

The core result on which is based all the subsequent analysis and simulations is the following theorem ∎

Theorem 1: Let $\mathcal{C} \subset GF(2^n)$ be a quasi-cyclic code of index p. Then we have

$$\widetilde{C}^{\perp} = \widetilde{C^{\perp}}$$

Proof 3: Let $\widetilde{\mathbf{h}} \in \widetilde{\mathcal{C}^{\perp}}$, and let $\mathbf{h} \in \mathcal{C}^{\perp}$ be a preimage of $\widetilde{\mathbf{h}}$. Now let $\widetilde{\mathbf{c}} \in \widetilde{\mathcal{C}}$ and let $\mathbf{c} \in \mathcal{C}$ be a preimage. We have

$$\mathbf{h} \cdot \mathbf{c} = \sum_{i=0}^{N} \sum_{u \in \mathcal{O}_i} h_u c_u = 0$$

Let τ be a generator of the quasi-cycle as described in the previous section. Let us denote by $\mathbf{h}^{\tau}$ be the vector $\mathbf{h}$ on which the permutation τ acts, *i.e.*

$$\mathbf{h}^{\tau} \overset{def}{=} (h_{\tau(1)}, \ldots, h_{\tau(n)})$$

Now since $\tau \in Perm(\mathcal{C})$, it is also in the permutation group of $\mathcal{C}^{\perp}$, therefore

$$\forall j = 0, \ldots, p-1, \ \mathbf{h}^{\tau^j} \cdot \mathbf{c} = \sum_{i=1}^{N} \sum_{u \in \mathcal{O}_i} h_{\tau^j(u)} c_u = 0$$

by summing on j one obtains

$$\sum_{j=0}^{p-1} \sum_{i=1}^{N} \sum_{u \in \mathcal{O}_i} h_{\tau^j(u)} c_u = 0 = \sum_{i=1}^{N} \sum_{u \in \mathcal{O}_i} \sum_{j=0}^{p-1} h_{\tau^j(u)} c_u$$

Since the orbits have same size p, for $j = 0, \ldots, p-1$ and for any $u \in \mathcal{O}_i$, $\tau(u)$ describes the orbit. This implies that $\sum_{j=0}^{p-1} h_{\tau^j(u)}$ does not depend on the chosen element u in the orbit $\mathcal{O}_i$. Hence

$$\sum_{i=1}^{N} \sum_{u\in\mathcal{O}_i} \sum_{j=0}^{p-1} h_{\tau^j(u)} c_u = \underbrace{\sum_{i=1}^{N} \sum_{u\in\mathcal{O}_i} h_u \cdot \sum_{u\in\mathcal{O}_i} c_u}_{\widetilde{\mathbf{h}}\cdot\widetilde{\mathbf{c}}} = 0$$

■

V. Folding the channel

Suppose that at the receiver's end one gets $\lfloor \in GF(2^n)$ such that

$$\mathbf{b} = (b_1, \ldots, b_n) = \underbrace{(c_1, \ldots, c_n)}_{\in\mathcal{C}} + \mathbf{e},$$

where $\mathcal{C}$ is a quasi-cyclic code of index p, and $\mathbf{e} = (e_1, \ldots, e_n) \in GF(2)^n$ is an error vector modelizing the effect of a BSC with cross-over probability $1/2 - \epsilon$. This implies that the e_i's are realizations of identically distributed independent Bernoulli trials such that

$$\forall\ i = 1, \ldots, n \ \begin{cases} \Pr(e_i = 0) = 1/2 + \epsilon \\ \Pr(e_i = 1) = 1/2 - \epsilon \end{cases}$$

By applying the folding operation of definition 2 to $\mathbf{b}$ we obtain

$$\widetilde{\mathbf{b}} = \underbrace{\widetilde{\mathbf{c}}}_{\widetilde{C}} + \widetilde{\mathbf{e}},$$

where $\widetilde{\mathbf{e}} = (\widetilde{e}_1, \ldots, \widetilde{e}_N)$ is an error-vector modelizing a BSC channel with cross-over probability $1/2(1-(2\epsilon^p))$, that is

$$\forall\ i = 1, \ldots, n \ \begin{cases} \Pr(\widetilde{e}_i = 0) = 1/2(1 + (2\epsilon)^p), \\ \Pr(\widetilde{e}_i = 1) = 1/2(1 - (2\epsilon)^p) \end{cases}$$

The independence of the random variables $\widetilde{e}_i = \sum_{u\in\mathcal{O}_i} e_u$ is directly induced from the fact that the orbits $\mathcal{O}_i$ consist of distinct elements.

Example 2: Suppose that the channel has cross-over probability 10^{-3} then

- if $p = 2$, the projected channel has cross-over probability 2×10^{-3} ;
- if $p = 45$, the projected channel has cross-over probability 4.3×10^{-3} ;
- if $p = 360$, the projected channel has cross-over probability 0.25.

VI. Reconstruction methodology

The receiver has access to M binary vectors of length n, say $\mathbf{b}_1, \ldots, \mathbf{b}_M$. The reconstruction hypotheses are the following:

- for all $j = 1, \ldots, M$

  $$\mathbf{b}_j = \mathbf{c}_j + \mathbf{e}_j$$

  where $\mathbf{c}_j \in \mathcal{C}$ and $\mathbf{e}_j$ modelizes a BSC channel with cross-over probability $1/2 - \epsilon$;
- The code $\mathcal{C}$ is a quasi-cyclic code of index p, and the positions of the orbits are known to the receiver;
- We use as a black-box the CJM algorithm described at section II;
- The receiver aims at finding parity-check equations of $\mathcal{C}$ of weight w.

The reconstruction principle is:

1) Select a folding index p' dividing p such that p' is the largest integer satisfying

$$M > \left(\frac{4}{(2\epsilon)^{p'w}}\right)^2 ; \tag{4}$$

2) Apply the folding operation on

$$\mathbf{b}_1, \ldots, \mathbf{b}_M \in (GF(2)^n)^M \overset{p'}{\to} \widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_M \in (GF(2)^{n/p'})^M;$$

3) Iterate CJM algorithm on the folded received vectors and obtain T parity-check equations of weight w for $\widetilde{C}$ say $\widetilde{\mathbf{h}}_1, \ldots, \widetilde{\mathbf{h}}_T$;
4) Find preimages $\mathbf{h}_j$ of weight w of the equations $\widetilde{\mathbf{h}}_j$.

In the second item, the folding operation is applied to received vectors, which are from previous results, noisy codewords $\widetilde{C}$ modified by the action of a BSC with cross-over probability $1/2(1-(2\epsilon^{p'}))$ The inequality condition on the amount of received data ensures that there are enough codewords so that the statistical test of CJM algorithm is meaningful.

Concerning the third item, we know from proposition 1 that a parity-check equation $\mathbf{h}$ of weight w of the code $\mathcal{C}$ is folded onto a parity-check equation $\widetilde{\mathbf{h}}$ of weight at most w of $\widetilde{C}$. The Hamming weights are equal if and only if there is at most one non-zero position per orbit. In our reconstruction hypotheses we suppose that for most of the parity-check equations of weight w in $\widetilde{\mathcal{C}}$, there exists a preimage of weight w.

Two results back up this claim:

- In standards, parity-check equations of LDPC codes have usually at most one non-zero bit per orbit. Therefore any parity-check equation $\mathbf{h}$ of weight w can be folded onto a parity-check equation of weight exactly w;
- The Hamming weight w of the targeted parity-check equations is well below Varshamov-Gilbert bound of $\widetilde{C}^{\perp}$. We use this condition to guarantee that a randomly chosen parity-check equation for $\mathcal{C}$ cannot be folded onto a parity-check equation of weight w for $\widetilde{C}$. This condition is clearly necessary, but is not sufficient. Namely, if w is larger than this bound it is not possible to distinguish the obtained parity-check equation from a randomly chosen parity-check equation.

Different techniques to achieve step 4 can be imagined. But this is well beyond the scope of the article. The efficiency of the technique that we describe on an examples supposes that the index of the quasi-cyclic code is smooth (can be decomposed in small factors). Roughly speaking, a quasi-cyclic code of index p is also a quasi-cyclic code of index p' where p' is any divisor of p. This idea is thus to find a suitable p' and a path of divisors from 1 to p':

$$1 \to p_1 \to \cdots \to p_t \to p'$$

This path of divisors gives also a path of folded codes

$$\mathcal{C} \to \mathcal{C}_1 \to \cdots \to \mathcal{C}_t \to \widetilde{C}$$

where p_{i+1}/p_i is small typically less than 5. We find weight w parity-check equations for $\mathcal{C}_i$ form weight w parity-check equations form $\mathcal{C}_{i-1}$ by testing the $(p_i/p_{i-1})^w$ possibilities. Therefore, if the maximum of the p_i/p_{i-1}, the complexity of step 4 is negligible.

This approach is particularly well suited to the family of LDPC codes specified in the DVB-S2 standard, [8]. These codes have length $16\ 200$ or $64\ 800$. These are not quasi-cyclic, but can be made quasi-cyclic of index $p = 360$ by a public permutation and puncturing some positions. The index $360 = 2^3 3^2 5$ is very smooth. The trellis of divisors is presented in figure 3

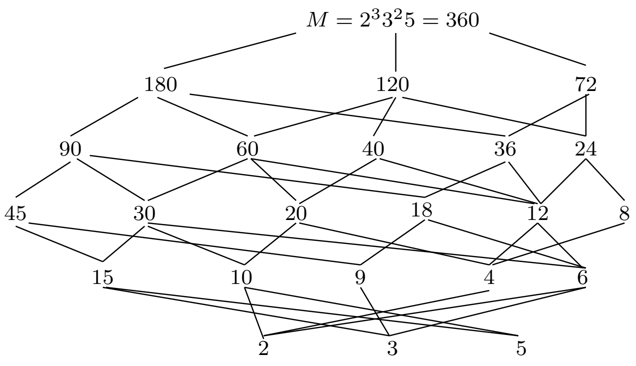

Fig. 3. Trellis of 360

With this technique, from the results of section II, we obtain the following tables, where $Min\ M$ is the minimum number of noisy codewords necessary for the reconstruction, and $Nb.\ Eq./Iter$ is the average number of parity-check equations of weight 8 found by one iteration of the CJM algorithm where the window ℓ is optimized accordingly.

N	p'	$Min\ M$	$Nb.\ Eq./Iter$	$Ops\ Bins$	$List\ Size$
64 800	1	31	2 370	2^{66}	2^{41}
32 400	2	60	1 179	2^{62}	2^{38}
16 200	4	218	518	2^{58}	2^{36}
8 100	8	2074	274	2^{54}	2^{33}
5 400	12	40 554		2^{51}	2^{31}

TABLE I
CROSS-OVER PROBABILITY: 2%

N	p'	$Min\ M$	$Nb.\ Eq./Iter$	$Ops\ Bins$	$List\ Size$
1 800	36	180	112	2^{45}	2^{27}
1 440	45	287	90	2^{44}	2^{26}
720	90	5 136	45	2^{40}	2^{23}

TABLE II
CROSS-OVER PROBABILITY: 2.10^{-3}

We ran an example with an implementation in C of the algorithm presented in section II. The code we considered was a rate $3/5$ code of length $n = 16\ 200$ generated via the standards of DVB-S2, and noisy codewords were received over a channel with noise $1 \cdot 10^{-6}$. We ran 1 iteration of the algorithm for different indexes. The amount of data on which we operated is $M = 200$ noisy codewords. The results are presented in the following table:

w	p'	Number of Eqs.	Ov. time(s)	Time per Eq.(s)	List Size
6	1	33	1 200	37	2^{17}
8	1	7 676	$9 \cdot 10^6$	1 100	2^{17}
8	72	28	1 174	23	2^9

TABLE III
SIMULATION RESULTS FOR DVB-S2 LDPC CODE WITH RATE $3/5$ AND LENGTH $16\ 200$

When searching parity-check equations of weight 8, our method speeds up the computational complexity by a factor of 50, and the list size factor gain is 300.

REFERENCES

[1] G. Planquette, “Identification de trains binaires codés,” Ph.D. dissertation, Univerité de Rennes 1, 1996.

[2] A. Valembois, “Detection and recognition of a binary linear code,” *Discrete applied Math.*, vol. 111, no. 1-2, pp. 199–218, 2014.

[3] J. Barbier, G. Sicot, and S. Houcke, “Algebraic approach of the reconstruction of linear and convolutional error-correcting codes,” in *CCIS 2006*, 2006.

[4] C. Chabot, “Recognition of a code in a noisy environment,” in *Proc. IEEE Int. Symposium Inf. Theory - ISIT2007*, Nice, France, Jun. 2007, pp. 2210–2215.

[5] M. Cluzeau and M. Finiasz, “Recovering a code’s length and synchronization from a noisy intercepted bitstream,” in *Proc. IEEE Int. Symposium Inf. Theory - ISIT2009*, Seoul, Korea, Jun. 2009, pp. 2737–2741.

[6] K. Carrier and J.-P. Tillich, “Identifying unknown code by partial Gaussian elimination,” in *Des. Codes Cryptogr.*, 2018, to appear.

[7] A. Tixier, “Reconnaissance de codes correcteurs,” Ph.D. dissertation, Universié Pierre et Marie Curie, Paris 6, 2015, available at: https://hal.archives-ouvertes.fr/tel-01238629v2.

[8] ETSI EN 302 307 V1.2.1., “Digital video broadcasting (dvb) ; second generation fra- ming structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications (dvb-s2).” Tech. Rep., 2009.

[9] J. Wang, Y. Yue, and J. Yao, “A method of blind recognition of cyclic code generator polynomial,” in *6th IEEE International Conference on Wireless Communications Networking and Mobile Computing, WiCOM*, 2010.

[10] A. D. Yardi, S. Vijayakumaran, and A. Kumar, “Blind reconstruction of binary cyclic codes,” in *Proceedings of the European Wireless*. VDE VERLAG GMBH, Berlin, Offenbach, Germany, 2014, pp. 849–854.

[11] ——, “Blind reconstruction of binary cyclic codes from unsynchronized bitstream,” *IEEE Trans. Comm.*, vol. 64, no. 7, pp. 2693–2706, 2016.

[12] C. Löndahl and T. Johansson, “Improved algorithms for finding low-weight polynomial multiples in $\mathbf{F}_2[x]$ and some cryptographic applications,” *Des. Codes Cryptogr.*, vol. 73, no. 2, pp. 625–640, 2014.

[13] P. Loidreau, “On cellular codes and their cryptogaphic applications,” in *ACCT 2014, Algebraic and Combinatorial Coding Theory*, 2014.

[14] P. Chose, A. Joux, and M. Mitton, “Fast correlation attacks: an algorithmic point of view,” in *Advances in Cryptology - EUROCRYPT 2002*, ser. LNCS, L. R. Knuden, Ed., vol. 2332, 2002, pp. 209–221.

[15] K. Lally and P. Fitzpatrick, “Discr. appl. math.” *Discrete applied Math.*, vol. 111, pp. 157–175, 2001.

[16] S. Ling and P. Solé, “On the algebraic structure of quasi-cyclic codes i: Finite fields,” *IEEE Trans. Inf. Theo.*, vol. 47, no. 7, pp. 2751–2759, 2001.

[17] E. R. Berlekamp, *Algebraic Coding Theory*, 1968.

[18] P. Loidreau, “Codes derived from binary Goppa codes,” *Probl. Inf. Transm.*, vol. 37, no. 2, pp. 91–99, 2001.