Block code reconstruction using iterative decoding techniques

Mathieu Cluzeau
INRIA projet CODES
B.P. 105
78153 Le Chesnay Cedex - France
Email: Mathieu.Cluzeau@inria.fr

***Abstract*— In this paper, we present a new algorithm for recovering a linear block code from noisy codewords. To achieve this, we introduce a version of Gallager algorithm with weighted parity-check equations. This new algorithm efficiently recovers LDPC codes and can also be used for other block codes.**

I. Introduction

The problem we address here is to be understood in the more general context of reverse-engineering a communication system. The general problem is, for an observer, to recover the transmitted information from the knowledge of the observed stream. But he does not know anything about the characteristics of the different elements except the noisy channel, and so his first goal is to determine which elements have been used in the communication system.

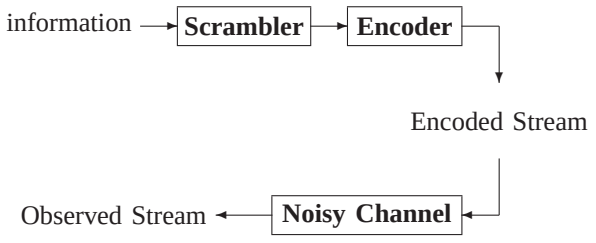

Fig. 1. Communication system.

Here, we are interested recovering the error-correcting code which has been used for the communication when this code is a linear block code over $\mathbb{F}_2$. Note that finding the error-correcting code does not mean finding the encoder. It only means that we want to find a basis of the linear space defined by the code or equivalently a generator matrix for this code. In [Val01], Valembois shows that the associated decision problem lies in the class of NP-complete problems. Even if this problem seems to be intractable, practical instances could be easy. Thus, we try to find some techniques to reconstruct an error-correcting code, especially if the code length is not too large and/or if the error rate is low.

In the following, we will denote by $\mathcal{C}$ the error-correcting code which has been used. Let n, k and d_{min} respectively denote its length, its dimension and its minimum distance.

First, we describe the main reconstruction technique as presented in [Val00]. Then we present a new algorithm for reconstructing an error-correcting code based on the classical tools of iterative decoding [Gal63]. The last section will be devoted to the simulation results based on this new algorithm.

II. Reconstruction technique

A. Basic principle

We assume that the synchronisation and the block length n are known. Then, we are able to split the observed binary stream into M noisy codewords of length n. In the following, we consider the $M \times n$ binary matrix X whose rows consist of all these n-bit noisy codewords. The technique we are presenting here consists in constructing a basis of the dual $\mathcal{C}^\perp$ of the targeted code.

$$\mathcal{C}^\perp = \{h | \forall c \in \mathcal{C},\ hc^T = 0\}$$

where hc^T denotes the scalar product between h and c over $\mathbb{F}_2^n$. If there was no error during the transmission, we would have:

$$\forall h \in \mathcal{C}^\perp, hX^T = 0,$$

where X^T denote the transposed matrix of X. Here, the binary sequence has been sent through a noisy channel and so, instead of having $hX^T = 0$, hX^T has a low Hamming weight. We will use the following result.

Proposition 1: If m is the word received after transmission through a binary symmetric channel with cross-over probability τ, then the probability that the number of error positions is even is :

$$\frac{1 + (1-2\tau)^n}{2}.$$

Corollary 1: Let $h \in \mathcal{C}^\perp$ and m be the result of the transmission of a codeword in $\mathcal{C}$ through a binary symmetric channel with cross over probability τ. Then:

$$Pr[hm^T = 0] = \frac{1 + (1-2\tau)^{wt(h)}}{2},$$

$$Pr[hm^T = 1] = \frac{1 - (1-2\tau)^{wt(h)}}{2}$$

where $wt(h)$ denote the Hamming weight of h, i.e. the number of its non-zero coordinates.

Let $h \notin \mathcal{C}^\perp$, then the Hamming weight of hX^T will be in average $\frac{M}{2}$ and if $h \in \mathcal{C}^\perp$, then the weight of hX^T will be lower: $\frac{M}{2}(1 - (1-2\tau)^{wt(h)})$.

This gives us a method for distinguishing the words of the dual code from the others (especially when the word h has a low Hamming weight) and so a starting point for reconstructing a parity-check matrix for the code $\mathcal{C}$. Furthermore, it gives us a method for finding words in the dual code: we have to search for linear combinations of the rows of X^T which have a low weight. In fact, when we find a word of low weight in the code spanned by X^T, it comes, with a high probability, from a word of $\mathcal{C}^\perp$.

B. Finding low-weight codewords

This problem has been studied in another context, for finding the minimum distance of a linear code. So there exist numerous papers on this topic [Leo88], [LB88], [Ste89]. Here we choose to use the following algorithm inspired by [CC98] in all simulations. The problem is as follows:

Let G be the generator matrix of an $[n, k]$ code. We want to find words of Hamming weight less than a given threshold T in the code spanned by G. Our algorithm involves three parameters: s, p and T.

We start by selecting a random information set I for the $[n, k]$ code. Then, we permute the columns of G such that the first k columns correspond to I and we perform a Gaussian reduction on G:

$$P^{-1}G = G_I = (I_k|Z)$$

where P is a k by k invertible matrix. We split the information set into two parts I_1 and I_2 of same size and we randomly choose a subset J of s positions outside the information set. We are then searching for all the words m such that

$$wt_H(m_{|_{I_1}}) = wt_H(m_{|_{I_2}}) = p \text{ and } wt_H(m_{|_J}) = 0.$$

A word m is expected to have a low Hamming weight since it has weight $2p$ on $k + s$ positions and in practice $p = 1$ or 2. We compute its actual Hamming weight and if it is smaller than the threshold T we output its original coordinate in G. That is, for $h = m_k$ (the first k positions of m), we have $m = hG_I = hP^{-1}G$ and if $wt(hZ) + 2p \leq T$, we output hP^{-1}.

Once we have done that for a given information set, we restart with another one until we get enough small weight codewords. In order to speed up the process and to avoid expensive Gaussian reduction, we change only one position of I at each step like in [CC98].

The parameters s and p could be optimised as shown in [Cha92]. The value of T will change the probability that the output words belong to the dual code. We will see later the role played by this parameter when we compute the probability that a given word h belongs to the dual code.

C. Reconstruction algorithm

In the previous paragraph, we have seen that we are able to find words h such that hX^T has a low weight. Those h are candidates to be in the dual code of $\mathcal{C}$. A test to decide whether or not h belongs to $\mathcal{C}^\perp$ was proposed by Valembois [Val01]. Whenever the ratio

$$\frac{Pr[X|h \in \mathcal{C}^\perp]}{Pr[X]} = (1 + (1-2\tau)^{wt(h)})^M \times \left(\frac{1-(1-2\tau)^{wt(h)}}{1+(1-2\tau)^{wt(h)}}\right)^{wt(hX^T)}$$

is greater than a given threshold, we decide that h belongs to $\mathcal{C}^\perp$. The algorithm proposed by Valembois consists in finding $n-k$ linear independent words such that they form a basis of $\mathcal{C}^\perp$. This algorithm is what he calls the first-order algorithm.

He also presents higher-order algorithms, the main difference being that, for an r-order algorithm, instead of testing words one by one, it tests whether a r-dimensional subspace is included in $\mathcal{C}^\perp$ or not. Let H_r denote the event: the linear space spanned by $h_1, \ldots, h_r$ is included in $\mathcal{C}^\perp$. Let $H = (h_1, \ldots, h_r)$, then:

$$\frac{Pr[X|H_r]}{Pr[X]} = \prod_{i=1}^{N} \sum_{B \in \mathbb{F}_2^r} (-1)^{<B, x_i H>} (1-2\tau)^{wt(BH^T)}.$$

When we test if the subspace spanned by r words of low weight is included or not in $\mathcal{C}^\perp$, we just compute the previous ratio and compare it to a threshold.

In each case, first-order or higher-order algorithms, we could summarise the algorithms by both following steps:

- generate words of low weight.
- test if they are in $\mathcal{C}^\perp$ and if so, eliminate the dependent vectors in order to compute a basis of $\mathcal{C}^\perp$.

III. Improvements

A. Basic principle

We have seen that in the algorithms proposed by Valembois, we just perform some statistical tests to decide whether or not a word is in the dual code. Our idea is instead to try to decode using our likely parity check-relations and then get some feedback on these equations. For that, we will compute the probability that a word belongs to the dual code and give it a weight accordingly. Then, we will use an iterative algorithm with weighted parity-check equations to correct some errors.

B. How to compute $Pr[h \in \mathcal{C}^\perp | wt(hX^T) = w]$

Let (h, w) be an output of the algorithm which searches for low weight words. So, we have $wt(hX^T) = w$. We want to compute $Pr\left[h \in \mathcal{C}^\perp | wt(hX^T) = w\right]$.

But, we can only compute $Pr\left[wt(hX^T) = w | h \in \mathcal{C}^\perp\right]$ and $Pr\left[wt(hX^T) = w | h \notin \mathcal{C}^\perp\right]$. Let m_i denote one codeword and $\widetilde{m}_i$ the noisy codeword, we have

$$X^T = \begin{pmatrix} \widetilde{m}_1 & \widetilde{m}_2 & \cdots & \widetilde{m}_M \end{pmatrix}.$$

Let $p_h = \frac{1-(1-2\tau)^{wt(h)}}{2}$ and $q_h = \frac{1+(1-2\tau)^{wt(h)}}{2}$, then using corollary 1, we can compute:

$$Pr\left[wt(hX^T) = w | h \in \mathcal{C}^\perp\right] = \binom{M}{w} p_h^w q_h^{M-w}.$$

Now, we want to compute $Pr\left[wt(hX^T) = w | h \notin \mathcal{C}^\perp\right]$. For large M, we can assume that our m_i are independent and randomly and then we are able to derive

$$Pr\left[wt(hX^T) = w | h \notin \mathcal{C}^\perp\right] = \frac{\binom{M}{w}}{2^M}.$$

Let $P_0 = Pr[wt(hX^T) = w | h \in \mathcal{C}^\perp]$ and $P_1 = Pr[wt(hX^T) = w | h \notin \mathcal{C}^\perp]$. We can then compute the probability $Pr[h \in \mathcal{C}^\perp | wt(hX^T) = w]$ and it is equal to:

$$\frac{Pr[h \in \mathcal{C}^\perp]P_0}{Pr[h \notin \mathcal{C}^\perp]P_1 + Pr[h \in \mathcal{C}^\perp]P_0}. \quad (1)$$

Finally, we can write this probability as a function of $Pr[h \in \mathcal{C}^\perp]$. Unfortunately, we could not compute this probability: indeed $Pr[h \in \mathcal{C}^\perp]$ depends on the weight distribution of $\mathcal{C}^\perp$ and of course, we do not know this distribution since we are searching for the code $\mathcal{C}$. In practice, for such communication schemes, only some well-known codes are used and we then know their weight distributions, at least for low weights (and that is what we are looking for). So, we will have to assume that the code which has been used belongs to a given family and then we could be able to compute $Pr[h \in \mathcal{C}^\perp | wt(hX^T) = w]$.

C. An iterative decoding algorithm

In this section, we recall the algorithm of Gallager introduced in [Gal62]. At each iteration we update the probability that a received bit is correct using parity-check equations. Using an equation of weight d:

$$EQ: \quad a_{t_0} + \sum_{j=1}^{d-1} a_{t_j} = 0,$$

we can compute a new probability for the bit a_{t_0} using other bits a_{t_j}. Assume we have a probability $Pr[a_{t_j} = 1]$ for each t_j, we want to compute $Pr[a_{t_0} = 1 | EQ]$. So,

$$Pr[a_{t_0} = 1 | EQ] = Pr[\sum_{j=1}^{d-1} a_{t_j} = 1].$$

Similarly to Proposition 1, we can prove that:

$$Pr[a_t = 1 | EQ] = \frac{1 - \prod_{i=1}^{d-1}(1 - 2Pr[a_{t_j} = 1])}{2}.$$

Now, let us consider a bit a_t, such that we have several informations coming from all equations in which a_t appears and from the observation.
In the following, we will denote by x the event $x = 1$ and by $\overline{x}$ the event $x = 0$. For two binary random variables y_1, y_2, if we assume that the channel is memoryless and that we have $Pr[x] = Pr[\overline{x}]$, then

$$Pr[x | y_1, y_2] = \frac{Pr[x|y_1]Pr[x|y_2]}{Pr[x|y_1]Pr[x|y_2] + Pr[\overline{x}|y_1]Pr[\overline{x}|y_2]}.$$

Let u and v be two reals in $[0, 1]$, we denote

$$u \otimes v = \frac{uv}{uv + (1-u)(1-v)}.$$

We could notice that we have:

$$Pr[x | y_1, y_2] = Pr[x | y_1] \otimes Pr[x | y_2].$$

Then the probability for a bit a_t is obtained by combining all the informations we have with the previous law $\otimes$.

We assume that the transmission has been done through a binary symmetric channel with cross-over probability τ. Let $Obs(a_t)$ denote the probability for a_t coming from the observation, $Extr_e(a_t)$ the probability for a_t coming from the e-th equation. And let $A_e(a_t)$ denote the partial APP (a posteriori probability) for a_t, i.e. this probability is computed using all informations on a_t except the one coming from the e-th equation. $APP(a_t)$ will denote the total probability. In the algorithm, there are three steps :

- Initialisation:
  - Compute $Obs(a_t) = 1 - \tau$ if the observation is 1. $Obs(a_t) = \tau$ if the observation is 0.
  - $\forall e$, $Extr_e(a_t) = \frac{1}{2}$.
- Iterate nb_{it}:
  For all equations e, update the probabilities $A_e(a_t)$ and then $Extr_e(a_t)$.
- End:
  For all t:
  - compute

$$APP(a_t) = Obs(a_t) \otimes \left[\bigotimes_i Extr_i(a_t)\right].$$

  - If $APP(a_t) > \frac{1}{2}$ then $a_t = 1$, otherwise $a_t = 0$.

D. Iterative decoding with probabilistic parity-checks

In our case, the main difference is that we have likely parity-check equations instead of exact ones. So we will take this into account when we update the probabilities. Consider we have an equation of weight d:

$$EQ: \quad a_{t_0} + \sum_{j=1}^{d-1} a_{t_j} = 0,$$

which holds with probability p. We want to compute $Pr[a_{t_0} = 1 | Pr[EQ] = p]$. This probability is equal to:

$$pPr[\sum_{j=1}^{d-1} a_{t_j} = 1] + (1-p)Pr[\sum_{j=1}^{d-1} a_{t_j} = 0].$$

We can compute it by

$$Pr[a_{t_0} = 1 | Pr[EQ] = p] = p\frac{1 - \prod_{i=1}^{d-1}(1 - 2Pr[a_{t_j} = 1])}{2} + (1-p)\frac{1 + \prod_{i=1}^{d-1}(1 - 2Pr[a_{t_j} = 1])}{2}.$$

and we can combine all the informations on a bit as before. We also want to update the probability of parity-check equations. When we decode a given noisy codeword $\widetilde{m}_i$ we can compute the probability $\widetilde{p}_{e,i}$ that the e-th equation is true according to the probabilities of the bits involved:

$$\widetilde{p}_{e,i} = \frac{1 + \prod_{t \in I_e} (1 - 2A_e(a_t))}{2}.$$

For a given equation, we have many informations: our first estimation (Est_e) and, for all i, $\widetilde{p}_{e,i}$. Then, we update the probability of the e-th equation in the decoding of the i-th word by:

$$p_{e,i} = Est_e \otimes \left[\bigotimes_{j \neq i} \widetilde{p}_{e,j} \right].$$

So, we can use the following algorithm which corresponds to Gallager algorithm in which the extrinsic probabilities are updated using the fact that the e-th parity-check equation holds with probabilities $p_{e,i}$ when we are decoding the i-th word.

- Initialisation:
  For all equations, compute Est_e
  For all words $\widetilde{m}_i$
  - Compute $Obs(a_t) = 1 - \tau$ if the observation is 1. $Obs(a_t) = \tau$ if the observation is 0.
  - $\forall e$, $Extr_e(a_t) = \frac{1}{2}$.
  - $\forall e$, $\widetilde{p}_{e,i} = \frac{1}{2}$.
- Iterate nb_{it}:
  For all words $\widetilde{m}_i$ and for all e:
  - Compute
    $$A_e(a_t) = Obs(a_t) \otimes \left[\bigotimes_{j \neq e} Extr_j(a_t) \right].$$
  - Compute
    $$\widetilde{p}_{e,i} = \frac{1 + \prod_{t \in I_e} (1 - 2A_e(a_t))}{2}$$
    where $I_e = \{i | a_i \text{ belongs to Equation } e\}$.
  - Compute
    $$p_{e,i} = Est_e \otimes \left[\bigotimes_{j \neq i} \widetilde{p}_{e,j} \right].$$
  - Compute
    $$Extr_e(a_t) = (\frac{1}{2} - p_{e,i}) \prod_{j \in I_e \setminus \{t\}} (1 - 2A_e(a_j)) + \frac{1}{2}.$$
- End:
  For all words $\widetilde{m}_i$, for all t:
  - compute
    $$APP(a_t) = Obs(a_t) \otimes \left[\bigotimes_{j} Extr_j(a_t) \right].$$
  - If $APP(a_t) > \frac{1}{2}$ then $a_t = 1$, otherwise $a_t = 0$.

E. New algorithm

Now, we have all the building blocks for reconstructing a linear code. Our algorithm will be as follows:
Iterate:

- Find some words of low weight in the code spanned by the matrix X^T, return h where h likely belongs to the dual code of $\mathcal{C}$. We will then associate to h a parity-check equation. Let denote H the set of candidate vectors h.
- Eliminate parity-check equations in H which involve cycles of short size (in our tests, cycles of length less than 4 are eliminated). For remaining words $h \in H$, compute $Pr[h \in \mathcal{C}^\perp]$ an approximation of the probability that h belongs to $\mathcal{C}^\perp$ as explained in section III-B.
- Use parity-check relations in H and their probabilities in order to correct some errors using the algorithm described in section III-D.

From Corollary 1, we can see that it will be easier to find a word of low weight in the code generated by X^T if there are words of low weight in $\mathcal{C}^\perp$. Furthermore, our decoding algorithm inspired by Gallager algorithm will be much more efficient when the weight of the parity-check equations are low. For these reasons, it is expected that the complexity of this algorithm decreases with the minimum distance of $\mathcal{C}^\perp$.

IV. Results

A. Reconstruction of a LDPC (3,6)

Since our decoding algorithm is based on the existence of low-weight words in the dual code, a first very natural example is the reconstruction of a LDPC. We here consider a LDPC (3,6) code, i.e. each column of the parity-check matrix has exactly 3 ones and each row has 6 ones (in this case, we have $\frac{k}{n} = \frac{1}{2}$ and so the dual code is also a [n,k] code). We have to compute $Pr[h \in \mathcal{C}^\perp]$. This requires an approximation of the weight distribution of the dual code at least for the low weights. We make some rough approximation on the number of words of weight less than 24 in the dual code of a LDPC (3,6) just using the fact that there are k words of weight 6. For example, most of words of weight 10 are obtained with two words of weight 6 which have only one position in common, this leads us to say that there are approximately $2k$ words of weight 10 because every positions participates in exactly 3 equations. For the others weights, we use the same kind of reasoning.

Then we are able to approximate the probability

$$Pr[h \in \mathcal{C}^\perp] = \frac{\#\{a \in \mathcal{C}^\perp | wt(a) = wt(h)\}}{\binom{n}{wt(h)}}.$$

Thus, we can also approximate $Pr[h \in \mathcal{C}^\perp | wt(hX^T) = w]$ for all h of weight less than 24. In practice, we seldom find words of weight larger than 24 using our algorithm and so we choose not to take them into account. For simulations, we separate the algorithm for finding words of low weight (FWLW) from the decoding algorithm in which we choose to take 30 iterations. We iterate the algorithm for finding low-weight words a bounded time. We recall that τ denote

the cross-over probability of our channel and we denote p_{err} the error rate/bit probability after decoding with our decoding algorithm. For a LDPC (3,6) of length 100, simulations on a Pentium 4 at 3.2GHz provide the following results:

τ	FWLW time	p_{err}	decoding time
0.01	1s	0	1s
0.02	1s	0.003	0.7s
0.02	5s	0.00017	0.8s
0.02	10s	0	0.8s
0.03	10s	0.027	0.5s
0.03	30s	0.022	0.74s
0.03	1min	0.011	1.12s
0.03	5min	0.002	1.43s
0.03	10min	0.0018	1.39s
0.03	1h	0.0012	1.37s

We can remark that, if we have to recover our code when the cross-over probability of our channel is 0.03, we can do better than using our algorithm for finding words of low weights during 20min and then decode using our decoding algorithm. For that, we use our FWLW algorithm during 1 minute, then we decode using our decoding algorithm. We have reduced our cross-over probability to 0.01 and we could now use our FWLW algorithm on the output stream of the decoding algorithm. With such a cross-over probability, an only second is enough to recover the dual code. In practice, we need a few more time since most of the new words of low weight would be linearly dependent from the ones used for the first decoding step.
Let us see what happens for the more realistic case of a LDPC code of length 1000.

τ	FWLW time	p_{err}	decoding time
0.001	1min	0.0001	53s
0.001	2min	0.000006	56s
0.001	3min	0.000008	55s
0.001	4min	0	57s
0.001	5min	0	57s
0.002	1min	0.0019	12s
0.002	5min	0.0017	16s
0.002	10min	0.0014	25s
0.002	20min	0.0011	30s
0.002	30min	0.0007	36s
0.002	1h	0.0005	43s

We can see that for a code of length 1000, we manage to correct main errors only if the cross-over probability of our channel is very low so we will be able to reconstruct it only in this case. However, in practice, demodulation provides soft information. If the amount of data is large enough, we can choose most probable noisy codewords. Then cross-over probabilities like 0.001 can be reached.

B. Reconstruction of a random code of length 100 and dimension 50

We have seen that our algorithm is efficient in the case of a LDPC (3,6) code and now, we will see what happens in the general case of a random code. As we have taken for $\mathcal{C}$ a random code, $\mathcal{C}^{\perp}$ is also a random code and then, we can use the weight distribution of a random code which is a binomial distribution. We then have:

$$Pr[h \in \mathcal{C}^{\perp}] = \frac{1}{2^k}.$$

Thus, we are able to approximate $Pr[h \in \mathcal{C}^{\perp} | wt(hX^T) = w]$ for all h. Our algorithms for finding words of low weight and using them to decode provide the following results for a random code of length 100 and dimension 50.

τ	FWLW time	p_{err}	decoding time
0.005	10s	0.003	1.22s
0.005	30s	0.003	1.18s
0.005	1min	0.003	1.20s
0.01	10s	0.006	1.18s
0.01	30s	0.006	1.19s
0.01	1min	0.006	1.18s
0.01	10min	0.006	1.19s
0.02	30s	0.02	0s
0.02	5min	0.02	0s
0.02	1h	0.02	0s

For random codes, our algorithm is less efficient than for LDPC codes since it takes more time to find words of low weight and to decode, so it will be harder to recover our code. However, we know that random codes are very difficult to decode. Moreover, we manage to correct some errors (and so to recover easily our code) in a few minutes when the cross-over probability is lower than 1 percent.

ACKNOWLEDGMENT

The author would like to thank J-P. Tillich for his helpful advice. He also wants to aknowledge F.Didier and Y.Laigle-Chapuy for their help in the algorithms implementations.

REFERENCES

[CC98] A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: application to primitive narrow-sense BCH codes of length 511. *IEEE Trans. Info. Theory*, 44(1):367–378, january 1998.

[Cha92] F. Chabaud. Asymptotic analysis of probabilistic algorithms for finding short codewords. In P. Camion, P. Charpin, and S. Harari, editors, *EUROCODE 92*, number 339 in CISM Courses and Lectures, pages 175–183. Springer-Verlag, 1992.

[Gal62] R. G. Gallager. Low density parity check codes. *IRE Trans. Info. Theory*, IT-8:21–28, 1962.

[Gal63] R. G. Gallager. *Low Density Parity Check Codes*. MIT Press, 1963.

[LB88] P.J. Lee and E.F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In C.G. Günter, editor, *Advances in Cryptology - EUROCRYPT'88*, number 330 in Lecture Notes in Computer Science, pages 275–280. Springer-Verlag, 1988.

[Leo88] J.S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Trans. Info. Theory*, 34(5):1354–1359, 1988.

[Ste89] J. Stern. A method for finding codewords of small weight. In G. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, number 388 in Lecture Notes in Computer Science, pages 106–113. Springer-Verlag, 1989.

[Val00] A. Valembois. *Décodage, Détection et Reconnaissance des Codes Linéaires Binaires*. PhD thesis, Université de Limoges, 2000. in French.

[Val01] A. Valembois. Detection and recognition of a binary linear code. *Discrete Applied Mathematics*, 111:199–218, 2001.