Recovering Synchronization with Iterative Decoders: LDPC Codes

Raúl Martínez-Noriega
Grad. School of Electro-Comms.
The University of Electro-Comms.,
Tokyo, Japan 182-8585
Email: raul@ice.uec.ac.jp

Brian Kurkoski and Kazuhiko Yamaguchi
Grad. School of Informatics and Eng.
The University of Electro-Comms.,
Tokyo, Japan 182-8585
Email: {kurkoski,yama}@ice.uec.ac.jp

Kingo Kobayashi
Inform. Security Research Center
National Inst. of Inf. and Comms. Tech.,
Tokyo, Japan 184-8795
Email: kingo@ice.uec.ac.jp

***Abstract*—We study a new synchronization algorithm based on low-density parity check codes. The algorithm was developed for scenarios with redundant information in 2010, [1]. We describe a revised version of the algorithm and, for the first time, we discuss the fundamentals about the coding and decoding theory. The results of this analysis define the scope and restrictions of the algorithm. We show that the algorithm is capable of recovering synchronization even in scenarios without redundant information. The algorithm has the characteristic of using not only cyclically permutable codes like the related proposals. Nevertheless special attention must be paid to short codes. Finally, an accurate approximation of the bound is introduced by using maximum likelihood decoding.**

I. Introduction

Reliable communications involve techniques to prevent errors generated in the transmissions. Block error-correcting codes are algorithms that attempt to correct those errors by dividing the information into blocks and adding redundancy to each block. However the encoder and the decoder must be in perfect synchrony, that is they must know exactly where a certain block begins and ends. Otherwise, even with the lack of noise, the decoder will emit unintelligible results. Hence, synchronization is important when the information is transmitted as blocks called codewords.

Synchronization can be lost due to many reasons. For example, clipped codewords produced by deletion of symbols in the channel, interruption of the communication between transmitter-receiver, carrier phase offset, etc.

One of the first attempts to keep the synchronization was the use of a special symbol which delimited the end and the beginning of different codewords, with the constraint that the special symbol was not a member of the information alphabet. Morse code applies this technique, and the letter space is the delimiter among codewords.

Codeword separation is implicit on comma-free codes [2] and [3]. These codes have the property that no valid codeword is a prefix of any other valid codeword. Therefore, if the received stream is read from left to right the codeword is immediately recognized because it is the only block that looks like valid codeword. However comma-free codes do not have such good error correction capability as other good codes. In addition, if the code length is large, the decoding produces delays.

In 1965, Levenshtein proposed binary codes which aided maintaining the synchronization [4]. His aim was not codeword separation, he defined a code that is capable of recovering a codeword with s or fewer deletions or insertions instead. However, depending on the code, some restrictions must be applied. For example, the code can correct only one deletion for every three codewords.

Common codes used in schemes with loss of synchronization are cyclically permutable (CP) codes proposed by Gilbert [5]. CP codes do not have implicit synchronization capability by themselves, but the synchronization property appears if there are redundant codewords at the decoder. An example of this principle was applied by Kuribayashi [6] in digital watermarking to desynchronization attacks. Generally, CP codes are constructed by discarding codewords from a cyclic code [5] and [7]. Therefore, CP codes have low coding rates which is its major disadvantage.

Most of the algorithms with self-synchronous capability are tied to some specific code, e.g. CP-codes, comma-free, Levenshtein's codes. However powerful error correcting codes, e.g. low-density parity-check (LDPC) codes or turbo codes, are desirable in noisy channels.

LDPC codes, [8] and [9], have better BER performance than turbo codes for long codes. They work under the general principle that the longer the code length, the closer they are to the theoretical channel capacity. Unlike BCH codes, decoding complexity of LDPC codes increases linearly in its codeword length. Besides LDPC codes have been shown to approach the capacity of various channels.

In [10] and [11], LDPC codes have been already used in schemes that keep the transmitter and the receiver in sync. However, the desynchronization problem is from a different nature. They both assume desynchronization because carrier phase offset which differs from the aim of this paper. They do not consider loss of synchronization because of clipped codewords at decoder.

We consider scenarios where the desynchronization is due to deletions in the channel that produce clipped codewords at the receiver. This problem is common in digital watermarking where attacks like clipping causes desynchronization [12]. The problem is described and formulated in Sec. II.

In Sec. III, we describe a revised and extended version of the self-synchronous algorithm introduced in [1] with addition of the fundamentals behind the synchronization capability. The algorithm assumes a received bit stream with redundant codewords where there are clipped codewords. Thus, the decoder achieves synchronization with the nearest unclipped codeword.

The algorithm is analyzed, the scope and restrictions are defined and the performance is shown through simulations in Sec. IV. We show that the difference between synchronized transmission and our algorithm is reduced when the code length is increased. The algorithm is also capable to recover synchronization in scenarios without redundant codewords.

We introduce an idea about the performance bound of the proposed algorithm, in Sec. V, with the aid of maximum likelihood decoding. An interesting point about the bounding analysis is the generation of a code which is non-linear but with cyclic-like characteristics.

Finally, our proposal is not restricted to a specific code, e.g. CP-codes. Indeed, a wide variety of codes can be used including non CP-codes. Sec. VI concludes the paper.

II. Problem description and statements

Consider transmissions where codewords C^i from a codebook $\mathscr{C}$ are sequentially and iteratively transmitted $p > 1$ times each of them, i.e. the same codeword C^i is continuously transmitted p times before a different codeword C^j could be transmitted:

$$\dots, C_1^i, C_2^i, \dots, C_p^i C_1^j, C_2^j, \dots, C_p^j, \dots,$$

and every codeword $C = (c_1, c_2, \dots, c_n)$ contains n bits.

Assume loss of synchronization due to any consecutive deletion of α bits. Therefore, the decoder obtains a stream that contains a clipped codeword followed of an unclipped codeword,

$$\dots, c_{\alpha+1}^i, c_{\alpha+2}^i, \dots, c_n^i, c_1^i, c_2^i, \dots, c_n^i, \dots,$$

thus the decoder does not know where the unclipped codeword begins. The problem is to recover the synchronization with the nearest, unclipped codeword.

Deletions could occur among same codewords, $C_1^i, C_2^i, \dots, C_p^i$, or different ones $C^i, C^j, \dots$. We will focus on the former case however the algorithm is not limited to deletions between same codewords. Any other deletion pattern, e.g. a bit deletion per each codeword, is out of the scope of this paper.

III. Synchronization scheme

In this Section the proposed algorithm will be described along with some necessary definitions.

A. Definitions

Let us define $C = (c_1, c_2, \dots, c_n)$ as a codeword of length n. Thus, one-bit cyclic shift named "cyclic equivalent permutation" RC of the codeword C is:

$$RC = (c_2, c_3, \dots, c_n, c_1),$$

in the same way we will find that C contains n cyclic equivalent permutations, $RC, R^2C, \dots, R^nC$, where $C = R^nC$.

In 1963, Gilbert [5] defined an error correcting code called cyclically permutable (CP) code. The main characteristic of this code is that any valid codeword cannot be obtained by cyclically permuting another valid codeword.

Definition 1 (Cyclically permutable (CP) code): CP code is a set of codewords of length n such that for any two valid codewords $C^i = c_1^i, c_2^i, \dots, c_n^i$, $C^j = c_1^j, c_2^j, \dots, c_n^j$ and considering $t \in [1, n-1]$:

$$C^i \neq R^t C^i,$$

and for any j, where $i \neq j$,

$$R^t C^i \neq C^j.$$

Gilbert [5] and Kuribayashi [7], individually, proposed constructions of CP codes based on cyclic codes. However these codes have a disadvantage from the point of view of coding rate because CP codes are constructed by discarding cyclic equivalent codewords from the cyclic codebook.

Consider two codewords, C^i and C^j of length n them both. Hamming distance $d_H(C^i, C^j)$ is defined as the number of bits in which they differ. If the cyclic permutations of the codewords are considered, define *cyclic Hamming distance* as:

$$D_H(C^i, C^j) = \min_x d_H(C^i, R^x C^j),$$

where $R^x C^j$ means that the codeword C^j has been cyclic permuted x bits

Definition 2 (CP-distance): In a code (n, k) with 2^k codewords and code length n the *CP-distance* S is defined:

$$S = \min_{i \neq j} D_H(C^i, C^j). \tag{1}$$

B. Self-synchronizable decoding

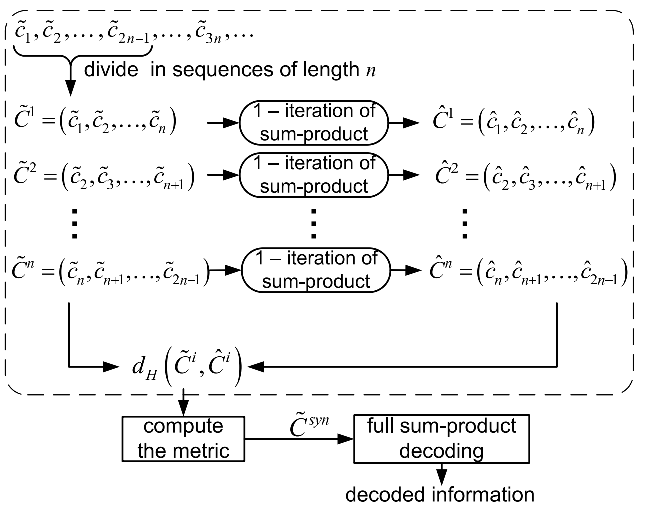
The synchronization capability of the proposed algorithm is based mainly on the decoding part. The encoder is a traditional LDPC code which takes blocks of k information bits and produces codewords of n bits.

The decoder receives a desynchronized stream of noisy bits, $\dots, \tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_{2n-1}, \dots, \tilde{c}_{3n}, \dots$. The index of $\tilde{c}_i$ could or could not match with the index of a certain codeword, i.e. $\tilde{c}_1$ does not necessarily mean the beginning of a codeword. The decoding algorithm aims to synchronize with the nearest unclipped codeword. Fig. 1 depicts this idea.

The algorithm takes the first $2n-1$ symbols from the noisy received stream. Those symbols are divided into n sequences $\tilde{C}^i$ of length n, each sequence differs from the previous one in a shifted bit to the right.

Sum-product algorithm (SPA) with only 1 iteration is applied to each sequence $\tilde{C}^i$ producing updated sequences $\hat{C}^i$. Hamming distance $d_H(\tilde{C}^i, \hat{C}^i)$ between the updated sequences $\hat{C}^i$ and its corresponding $\tilde{C}^i$ is measured.

On the other hand, the number of invalid parity checks B^i from each sequence $\hat{C}^i$ are computed. With long codes, the Hamming distance, $d_H(\tilde{C}^i, \hat{C}^i)$, provides reliable information about which sequence is the synchronized codeword. However



$\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_{2n-1}, \ldots, \tilde{c}_{3n}, \ldots$
divide in sequences of length n
$\tilde{C}^1 = (\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_n)$
1 – iteration of sum-product
$\hat{C}^1 = (\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_n)$
$\tilde{C}^2 = (\tilde{c}_2, \tilde{c}_3, \ldots, \tilde{c}_{n+1})$
1 – iteration of sum-product
$\hat{C}^2 = (\hat{c}_2, \hat{c}_3, \ldots, \hat{c}_{n+1})$
$\tilde{C}^n = (\tilde{c}_n, \tilde{c}_{n+1}, \ldots, \tilde{c}_{2n-1})$
1 – iteration of sum-product
$\hat{C}^n = (\hat{c}_n, \hat{c}_{n+1}, \ldots, \hat{c}_{2n-1})$
$d_H(\tilde{C}^i, \hat{C}^i)$
compute the metric
$\tilde{C}^{syn}$
full sum-product decoding
decoded information

Fig. 1. Decoding with self synchronization capability.

for codes with poor CP-characteristics or short codes, the Hamming distance is not enough because sometimes the minimum Hamming distance belongs to an invalid codeword. Hence, how reliable is the Hamming distance $d_H(\tilde{C}^i, \hat{C}^i)$ can be associated with the number of parity checks violated by $\hat{C}^i$.

A simple way to combine the Hamming distance and the number of violated parity checks is to add them. Then the metric m_i, which decides whether a sequence $\tilde{C}^i$ is synchronized, is computed with:

$$m_i = d_H(\tilde{C}^i, \hat{C}^i) + B^i. \tag{2}$$

Finally, the synchronized codeword is assumed to be the sequence which minimizes, $\tilde{C}^{syn} = \min_i m_i$, the metric. Therefore, only $\tilde{C}^{syn}$ is decoded using full iterations with SPA decoding. If $|\min_i m_i| > 1$, where $|\cdot|$ is the cardinality, the first element of the vector m_i is taken as the synchronized codeword.

For example, consider the next CP-codebook $\mathscr{C}$:

$$\begin{array}{ccc} 0 & 0 & 0 \\ 1 & 1 & 0. \end{array}$$

Assume that the next stream was transmitted:

$$110 \quad 110 \quad 110,$$

then desynchronization occurs because deletion of the first two bits producing 0110110. According to the algorithm, the first $2n-1 = 2(3)-1 = 5$ bits are taken 01101 and divided into sequences $\tilde{C}^i$ of $n=3$ continuous bits:

$$\tilde{C}^1 = 011$$
$$\tilde{C}^2 = 110$$
$$\tilde{C}^3 = 101.$$

Each sequence $\tilde{C}^i$ is updated with SPA producing $\hat{C}^i$.

$$\begin{array}{lcl} \tilde{C}^1 = 011 \neq \hat{C}^1 & & d_H(\tilde{C}^1, \hat{C}^1) > 0,\ B^1 \geq 0 \\ \tilde{C}^2 = 110 = \hat{C}^2 = 110 & \therefore & d_H(\tilde{C}^2, \hat{C}^2) = 0,\ B^2 = 0 \\ \tilde{C}^3 = 101 \neq \hat{C}^3 & & d_H(\tilde{C}^3, \hat{C}^3) > 0,\ B^3 \geq 0 \end{array}$$

Since $\{\tilde{C}^1, \tilde{C}^3\} \notin \mathscr{C}$ hence $\tilde{C}^1 \neq \hat{C}^1$ and $\tilde{C}^3 \neq \hat{C}^3$ because SPA attempts to correct the errors. Thus, $\tilde{C}^2$ is the only sequence with metric $m_2 = d_H(\tilde{C}^2, \hat{C}^2) + B^2 = 0$ equal to zero and therefore $\tilde{C}^{syn} = \min_i m_i = \tilde{C}^2$, $i \in [1, 3]$, is the correct and synchronized codeword.

Let us think about the input segment of $2n-1$ bits which is processed by the decoder. Assuming that desynchronization occurred between the same codewords, then the segment of $2n-1$ contains the codeword $C^i = (c_1^i, c_2^i, \ldots, c_n^i)$ concatenated with a prefix $C_{pre}^i = (c_1^i, c_2^i, \ldots, c_l^i)$ or suffix $C_{suf}^i = (c_{t+1}^i, c_{t+2}^i, \ldots, c_n^i)$ of the same codeword C^i, for $\{l, t\} \in [1, n)$. That is, the decoder could obtain one of the next concatenations: $[C_{suf}^i, C^i]$, $[C^i, C_{pre}^i]$ or $[C_{suf}^i, C^i, C_{pre}^i]$. Without distinction about which concatenation the decoder could obtain, all the n-tuples inside that segment are cyclic equivalent codewords of C^i. This claim can be proved with Theorem 1.

Theorem 1 (Cyclically equivalent codewords): Any sequence of n consecutive bits taken from the concatenation of $[C_{suf}, C]$, $[C, C_{pre}]$ or $[C_{suf}, C, C_{pre}]$ is a cyclic equivalent codeword of $C = (c_1, c_2, \ldots, c_n)$, where C has length n and $C_{suf} = (c_{t+1}, c_{t+2}, \ldots, c_n)$, $C_{pre} = (c_1, c_2, \ldots, c_l)$ are its suffix and prefix respectively with $\{t, l\} \in [1, n)$.

Proof: A cyclic permutation R^tC of a codeword $C = (c_1, c_2, \ldots, c_n)$, where $1 \leq t < n$, is defined as:

$$R^tC = (c_{t+1}, c_{t+2}, \ldots, c_n, c_1, \ldots, c_t). \tag{3}$$

Let us analyze the first case when $[C_{suf}, C]$. If C has length n, thus its suffix is:

$$C_{suf} = (c_{t+1}, c_{t+2}, \ldots, c_n).$$

Then, the concatenation of $[C_{suf}, C]$ produces:

$$[C_{suf}, C] = (c_{t+1}, c_{t+2}, \ldots, c_n, c_1, c_2, \ldots, c_n).$$

Taking any sequence of n consecutive elements from $[C_{suf}, C]$, we obtain:

$$c_{t+1}, c_{t+2}, \ldots, c_n, c_1, \ldots, c_t,$$

which is exactly the definition of cyclic permutation (3) of the codeword C.

Second case, $[C, C_{pre}]$. Since C has length n, the first n consecutive bits are the codeword C itself. From the second n consecutive bits the problem turns to be:

$$[C_{suf}, C_{pre}] = (c_{t+1}, c_{t+2}, \ldots, c_n, c_1, c_2, \ldots, c_l),$$

and

$$C_{pre} = c_1, c_2, \ldots, c_l,$$

with $1 \leq l < n$. Considering that C_{suf} has $n - t$ elements, any consecutive string of n inside $[C_{suf}, C_{pre}]$ is defined as:

$$c_{t+1}, c_{t+2}, \ldots, c_n, c_1, \ldots, c_t,$$

which is again the definition of cyclic permutation (3).

In the last case when $[C_{suf}, C, C_{pre}]$, there cannot be a sequence of n consecutive bits which contains elements of both C_{suf} and C_{pre} at the same time, because there is a codeword C of length n between them. Therefore, this problem can be broken in two sub-problems: $[C_{suf}, C]$ and $[C, C_{pre}]$ which were already proved. $\square$

If the encoder uses CP codes then no cyclic equivalent codeword of C^i exists in the codebook. Therefore is expected that the desynchronized sequences obtain more errors, because those sequences are invalid and also due to the channel noise. In consequence, SPA's convergence will need greater number of iterations and its number of invalid parity checks after the first iteration must be considerable. Then, computing $d_H(\tilde{C}^i, \hat{C}^i)$ and its reliability with B^i we can obtain a reliable guess about which $\tilde{C}^i$ is the synchronized codeword.

When the code is not precisely a CP code, it must have good CP-characteristics or a long code length. Good CP-characteristics mean that the code must have the less possible number of equivalent codewords in its codebook. For short codes, this can be achieved with equivalent codes generated by elementary row operation in its generator matrix.

With long codes is difficult to say that a code is strictly CP code because the number of codewords is huge. However this characteristic itself is a great advantage in our algorithm because the probability that any desynchronized sequence belongs to the codebook is reduced when the code length is increased. For long codes is expected that our method performs very close to perfect synchronized transmission.

It has been discussed the specific case when desynchronization happens between the same codewords. However desynchronization may also occur among different codewords, producing different concatenations, $[C^i_{suf}, C^j]$, $[C^i, C^j_{pre}]$ or $[C^h_{suf}, C^i, C^j_{pre}]$ where Theorem 1 does not hold. However if the code is large, the probability of errors due to an invalid sequence $R^t C^j$ turns into a codeword C^i, including $j = i$, is low. We refer to this error as *mis-decoding*.

IV. Performance of Self-synchronizable decoding scheme

In this Section comparison between desynchronized transmissions using the proposed algorithm and perfect synchronized transmissions are shown. The channel is a binary phase-shift keying with additive white Gaussian channel (AWGN). Finally, the scopes of the algorithm are discussed.

One of the advantages of this proposal is that a wide variety of codes, e.g. short or long, can be used. Fig. 2 shows the comparison between the proposed algorithm and synchronized transmission for a tiny LDPC code with rate .25, code length 12 and CP-distance $S = 2$. The simulation was conveyed by applying desynchronization to every transmitted codeword but, in this case, the desynchronization happens only among

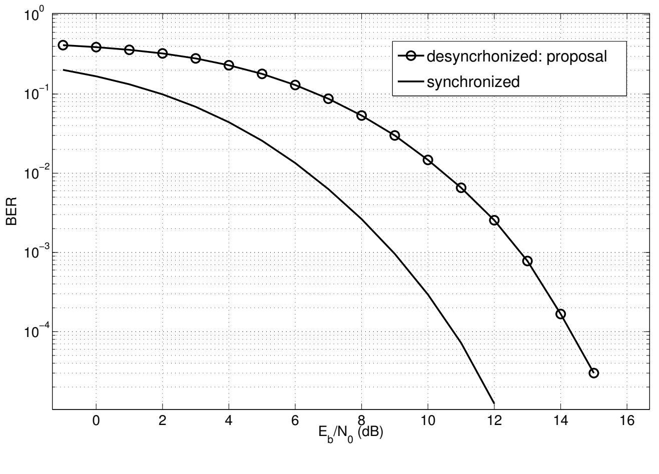


Fig. 2. Performance comparison between a synchronized transmission and our self-synchronizable decoding proposal over AWGN channel using a tiny LDPC code.

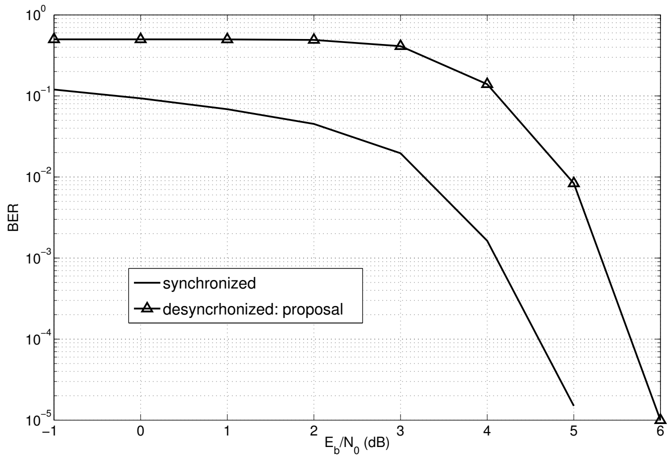


Fig. 3. Performance comparison between a synchronized transmission and our self-synchronizable decoding proposal over AWGN channel using a non CP-code: high rate LDPC code.

redundant codewords. The difference of performance is about 3 dB between both methods. This big gap is due to the code is very short, but is expected that the performance of both methods tend to be the same with large codes.

The proposed algorithm is not restricted to CP-codes, however attention must be paid to the CP-characteristics of the code. That is, the code could have CP-distance $S = 0$ but the code should contain a few number of equivalent cyclic codewords. Fig. 3 shows the previous idea. In this simulation the code is a high rate LDPC code with length 495 and rate .87, whether this code is precisely a CP-code or not is difficult to compute. Nevertheless, since the code is high-rate then is highly probable that the CP-distance is zero. In this case, the code length is bigger than Fig. 2 and therefore the difference between both methods has been reduced to only 1 dB.

In general, the proposed method works with codes of different lengths. The longer the code the better performance and less restrictions.

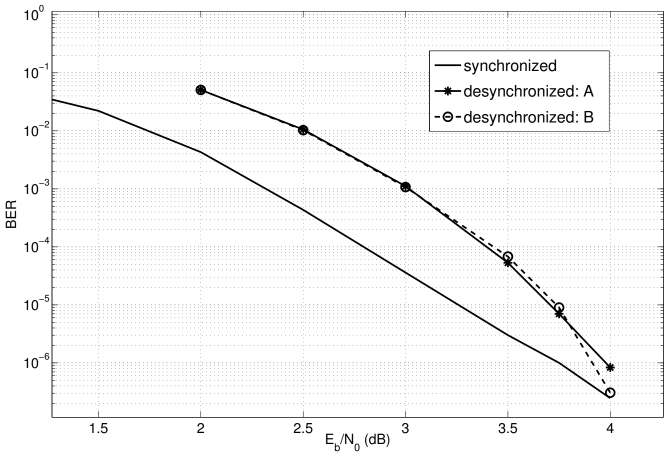


Fig. 4. Recovering synchronization due to, A: deletions among same codewords and B: deletions among different codewords.

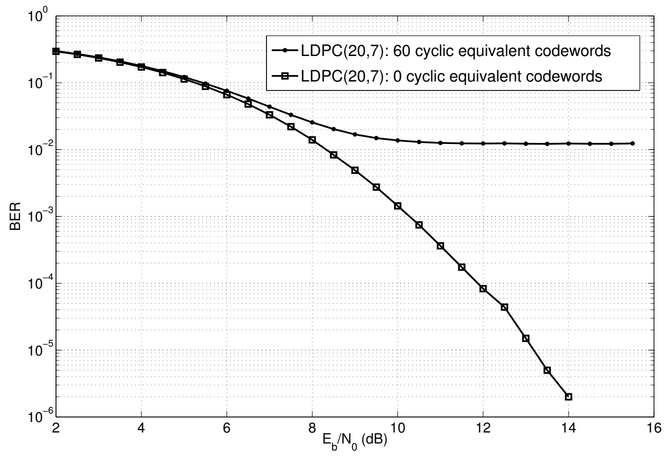


Fig. 5. Short codes: Performance of equivalent codes for desynchronized transmissions.

When the fundamentals behind this proposal were explained in Sec. III-B, we assumed that the desynchronization occurred in places with redundant codewords. We also explained that Theorem 1 does not hold when desynchronization happens among different codewords. However, if the code has long code length or good CP-characteristics the probability of misdecoding due to desynchronization among different codewords is low and the proposed algorithm performs as well as before.

Fig. 4 pictures the above situation. We used a half rate LDPC code with length 504. "synchronized" scheme is a normal transmission which was perfectly synchronized and only contaminated with AWGN. "desynchronized: A" is our proposal when the desynchronization happens only among redundant codewords and "desynchronized: B" is when desynchronization occurs among different codewords. Without distinction of the desynchronization place, the algorithm performs around .75 dB less than perfect synchronized transmission for low SNR and almost the same for high SNR. Comparing the gaps between synchronized transmission and our method in Figs. 2, 3 and 4, we are aware that the difference is reduced when the code length is increased.

Certain short codes are desirable in specific situations due to its good error correction capability or because of constraint length in the system. Those codes must have good CP-characteristics in order to work properly with our algorithm. That is, the less number of cyclic equivalent codewords in the codebook, the better.

Equivalent codes can be generated by applying elementary row operation in the generator matrix G. Those equivalent codes have a different number of cyclic equivalent codewords. Therefore, is preferable to choose a code with the minimum number of equivalent codewords.

In Fig. 5, a simulation using two equivalent tiny codes with the proposed algorithm is shown. First, a random LDPC code of length 20 is generated. However its performance is bad because it contains 60 cyclic equivalent codewords. Using its generator matrix, G, an equivalent code is found by elementary row operations. The equivalent code contains 0 cyclic equivalent codewords and it performs well with our method.

The main restrictions of this algorithm are two. The first one is related with the kind of code. By definition 3, a cyclic code contains the cyclic permutation of each valid codeword which is exactly what our algorithm is trying to avoid. Therefore cyclic codes are not suitable for this proposal.

Definition 3 (Cyclic Code): The linear code $\mathscr{C}^c$ of length n is a cyclic code if it is invariant under a cyclic permutation:

$$C^i \in \mathscr{C}^c \Leftrightarrow R^t C^i = C^j \in \mathscr{C}^c, \text{ for } t \in [1, n).$$

Then, a cyclic code contains all n cyclic permutation of any codeword.

The second restriction comes from the desynchronization nature. This proposal assumes that after the continuous deletion of α bits, there is a segment of $2n-1$ bits which contains the unclipped codeword C. Any other deletion pattern, e.g. one deleted bit every codeword or similar, is out of the scope of this proposal.

If the synchronized codeword were exhaustible searched in a segment of $2n-1$ bits, the decoder will need $n \times \lambda$ iterations to find the correct codeword, where n is the code length and λ is the number of iteration allowed by SPA. However this complexity is highly reduced in our algorithm because it only needs at most $n + \lambda$ iterations.

V. Toward the bound: Maximum likelihood decoding

The proposal's behavior has been defined and also its scopes have been discussed. However a theoretical bound that ensures the reliability of the algorithm is needed. In this Section, we introduce a simple approach based on maximum likelihood (ML) decoding which tends to bound the errors. Let us define $\mathscr{C}$ as the code used in our proposal. For convenience, a tiny LDPC code (12,3) will be used as toy example.

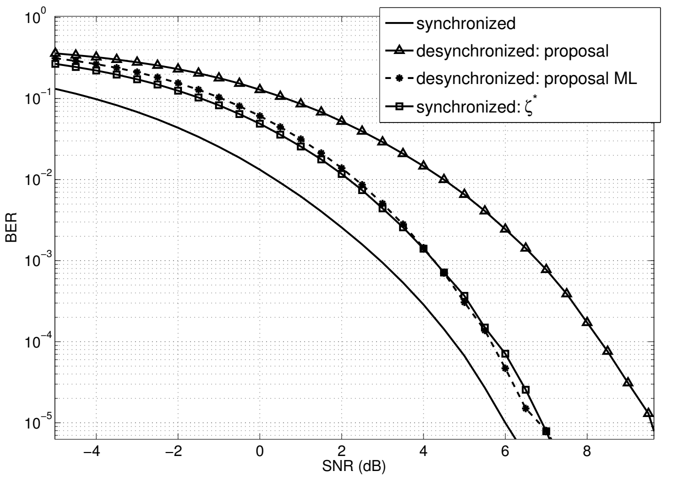


Fig. 6. Maximum likelihood decoding for our proposal and its potential bounded code.

Recapitulating, the proposed algorithm works under the premise that no cyclic equivalent codeword of a valid codeword C^i exists in the codebook or the number of cyclic equivalent codewords are very few. Consider codewords $\{C^i, C^j\} \in \mathscr{C}$ including $i = j$. Thus, the algorithm produces a decoding error when either: an invalid sequence R^tC^j becomes C^i or when the synchronized codeword C^i becomes a cyclic equivalent codeword R^tC^j of any valid codeword in the codebook.

Now, let us think about a cyclic code $\mathscr{C}^c$. For any codeword C^i in a cyclic codebook, its cyclic equivalent codewords are also valid codewords. Then, a decoding error occurs when either any cyclic equivalent codeword R^tC^j becomes C^i or vice versa, including $i = j$.

This behavior of cyclic codes under normal circumstances, i.e. synchronized transmissions, is very similar to our desynchronized model. Therefore, would be natural to think that the proposed algorithm could be bounded by a cyclic code $\mathscr{C}^c$ over a synchronized transmission, in which $\mathscr{C}$ is a subcode of $\mathscr{C}^c$.

However, it turns that the code $\mathscr{C}^c$ cannot be generated when $\mathscr{C}$ is extended, that is when all cyclic equivalent codewords of $\mathscr{C}$ are computed. A special code which is neither linear nor cyclic appears instead, we will define this code as $\mathscr{C}^*$. Therefore, well-known bounds for linear codes, e. g. *union bound*, cannot be applied.

Fig. 6 shows a simulation using the previous idea with a tiny LDPC code, rate .25 and code length 12. "synchronized" refers to synchronized transmission using $\mathscr{C}$, "desynchronized: proposal" is the proposed algorithm with desynchronization and the codebook $\mathscr{C}$, "desynchronized: proposal ML" means the proposal with ML decoding and $\mathscr{C}$, finally "synchronized: $\mathscr{C}^*$" stands for the potential bound using codebook $\mathscr{C}^*$, synchronized and with ML decoding. This simple idea shows a good approximation about the bound for the algorithm described in this paper.

VI. Conclusion

Synchronization is a very important issue for transmission where the information is sent in blocks. In this paper, we described a revised version of our proposal for transmission with loss of synchronization and we extended that study by defining its scope and restrictions.

The results showed that the algorithm is capable to use a wide range of codes, short and long ones. It is not restricted to CP-codes as previous proposals are. When short codes are needed, a solution was proposed by computing equivalent codes with better CP-characteristics.

Two major restrictions of this algorithm are the deletion pattern and the incompatibility with cyclic codes, that is the algorithm cannot work with cyclic codes. The algorithm is defined only for channels where continuous deletion occurs, e.g. watermarking under clipping attack.

Finally, an idea for bounding the number of errors in the algorithm was introduced. An important point is that the potential code which could bound the number of errors is neither linear nor cyclic. This idea was demonstrated with ML decoding and it produced almost the same curve as ML decoding. As future work, we are planning to polish the bound.

References

[1] R. Martinez-Noriega, I. Abe, and K. Yamaguchi, "Self-synchronizable decoding algorithms for transmission with redundant information at decoder," *IEICE Trans. on Fundamentals of Electronics Comms. and Computer Science*, 2010.

[2] S. W. Golomb *et al.*, "Comma-free codes," *Canadian Journal of Mathematics*, vol. 10, no. 2, pp. 202–209, 1958.

[3] H. Imai, "A construction method for path-invariant comma-free codes," *IEEE Trans. on Inf. Theory*, vol. 20, no. 4, pp. 555–559, 1974.

[4] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Cybernetics and Control Theory*, vol. 10, no. 8, pp. 707–710, 1966.

[5] E. N. Gilbert, "Cyclically permutable error-correcting codes*," *IEEE Trans. on Inf. Theory*, vol. 9, pp. 175–182, 1963.

[6] M. Kuribayashi and M. Morii, "Spread spectrum watermark with self-synchronization capability," in *Proc. of SCIS*, Sasebo, Japan, Jan. 2007.

[7] M. Kuribayashi *et al.*, "How to generate cyclically permutable codes from cyclic codes," *IEEE Trans. on Inf. Theory*, vol. 52, no. 10, pp. 4660–4663, 2006.

[8] R. Gallager, "Low-density parity-check codes," *IRE Trans. on Inf. Theory*, vol. 8, no. 10, pp. 21–28, 1962.

[9] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, 1999.

[10] W. Matsumoto and H. Imai, "Blind synchronization with enhanced sum-product algorithm for low-density parity-check codes," in *Proc. of The 5th Int. Symp. on Wireless Personal Mult. Comms.*, Oct. 2002, pp. 966–970.

[11] H. Steendam *et al.*, "Iterative carrier phase synchronization for low-density parity-check coded systems," in *Proc. of IEEE Int. Conf. on Comms.*, May 2003, pp. 290–294.

[12] R. Martínez-Noriega, M. Nakano, and K. Yamaguchi, "Self-synchronous time-domain audio watermarking based on coded-watermarks," in *Proc. of Int. Conf. on Int. Info. Hiding and Mult. Sig. Process.*, Oct. 2010.