

Article

Information-Dispersion-Entropy-Based Blind Recognition of Binary BCH Codes in Soft Decision Situations

Jing Zhou *, Zhiping Huang, Chunwu Liu, Shaojing Su and Yimeng Zhang

School of Mechatronics Engineering and Automation, National University of Defense Technology, Deya Road, Changsha 410073, Hunan Province, China; E-Mails: h.zhiping@hotmail.com (Z.H.); liuchunwu@nudt.edu.cn (C.L.); su_shaojing@163.com(S.S.); zhang_yimeng@nudt.edu.cn(Y.Z.)

* Author to whom correspondence should be addressed; E-Mail: zhou.jing@nudt.edu.cn; Tel.: +86-0731-84576387; Fax: +86-0731-84576387.

Received: 18 March 2013; in revised form: 24 April 2013 / Accepted: 1 May 2013 / Published: 8 May 2013

Abstract: A method of blind recognition of the coding parameters for binary Bose-Chaudhuri-Hocquenghem (BCH) codes is proposed in this paper. We consider an intelligent communication receiver which can blindly recognize the coding parameters of the received data stream. The only knowledge is that the stream is encoded using binary BCH codes, while the coding parameters are unknown. The problem can be addressed on the context of the non-cooperative communications or adaptive coding and modulations (ACM) for cognitive radio networks. The recognition processing includes two major procedures: code length estimation and generator polynomial reconstruction. A hard decision method has been proposed in a previous literature. In this paper we propose the recognition approach in soft decision situations with Binary-Phase-Shift-Key modulations and Additive-White-Gaussian-Noise (AWGN) channels. The code length is estimated by maximizing the root information dispersion entropy function. And then we search for the code roots to reconstruct the primitive and generator polynomials. By utilizing the soft output of the channel, the recognition performance is improved and the simulations show the efficiency of the proposed algorithm.

Keywords: information dispersion entropy; blind recognition; channel coding; BCH codes; adaptive coding and modulation (ACM)

PACS Codes: 89.70.+c

1. Introduction

Channel coding, applied to reduce the errors in transmissions, is an important part in digital communications [1]. The binary Bose-Chaudhuri-Hocquenghem (BCH) codes are widely used for their powerful error-correcting capability [2], convenient design of encoders [3], and efficiency of decoding algorithms [4,5]. In traditional communication systems, the coding parameters are known by both the transmitters and receivers, but with the development of cognitive radios and intelligent techniques, blind recognition of the channel coding parameters is becoming available and realizable. Cognitive radio (CR) was introduced in [6] as a smart spectrum sharing technology, and it is becoming a hot research topic [7–10]. The adaptive coding and modulations (ACM) technique [11–14] is an important section of the CR to adapt the channel. In an ACM system, the transmitter chooses optimized coding and modulation parameters according to the channel quality. Thus at the reception, the receiver need to recognize those parameters before demodulation and decoding. Another application field of the blind recognition technique is non-cooperative communications [15,16]. In this case, a non-cooperative receiver does not known the modulation and coding parameters before recognizing them. In the future communications, the terminals are required to be as intelligent as possible to adapt themselves to a specific context and to blindly estimate the transmitter parameters for self-reconfiguration purpose, only with knowledge of the received data stream [17].

To the best of our knowledge, most of the blind recognition algorithms proposed in the literature are focused on convolutional codes. In [18], a Euclidean-algorithm-based method was proposed to identify a 1/2 rate convolutional encoder in noiseless cases. However, it is not suitable for noisy channels. In [19], another approach was presented to identify a 1/n rate convolutional encoder in noisy cases based on the Expectation Maximization algorithm. The authors of [20,21] developed methods for blind recovery of convolutional encoders in turbo code configuration. In [13] and [17], dual code methods for blind identification of k/n rate convolutional codes ware proposed for cognitive radio receivers.

In this paper we consider the problem of blind recognition of the coding parameters for a cognitive receiver. The main focus is on the widely used BCH codes. Some previous literature reports [22–24] proposed and developed recognition algorithms for BCH codes in hard decision situations. In [22] and [23], the authors proposed a blind recognition algorithm for BCH codes based on Root Information Dispersion Entropy and Root Statistic (RIDERS). This algorithm can achieve correct recognition with a bit error rate (BER) of 10^{-2} , but it is computationally intensive, especially when the code length is large. The authors of [24] improved that algorithm proposed in [22,23] to reduce the computational complexity, which made the recognition procedure faster.

However, the previous works on blind recognition of BCH codes are all in hard decision situations, and are based on utilizing the algebraic properties of the codes in Galois Fields (GF). To achieve available recognition results, large amounts of training data are always required. With the development of sampling and signal processing techniques, the soft output of the channel has become available. Many soft-decision-based decoding algorithms have been applied to the error-correcting codes [25–30] and yield better decoding performances than hard decision algorithms. Some blind frame synchronization techniques for error correcting codes also utilize the soft output of the channel to improve the synchronization performances [31–34]. Inspired by the soft decision decoding algorithms, we develop the

RIDERS algorithm introduced by [22–24] in the soft decision situations in this paper. As an example we mainly discuss the problem of the soft decisions of BPSK modulation on AWGN channels.

The remaining of this paper is organized as follows: Section 2 presents the code length and coding starting positions estimation approach. Section 3 gives the code roots recognition method. Section 4 discusses the recognition of the primitive polynomial. In Section 5, we compare the computational complexity between hard decision and soft decision situations. Finally, the simulation results and conclusions are given in Section 6 and Section 7.

2. Code Length Estimation and Blind Synchronization

2.1. Introduction of the Recognition Algorithm in Hard Decision Situations

In cyclic coding theories, the algebraic model of the encoding operation can be expressed as follows:

$$c(x) = m(x) \times g(x) \tag{1}$$

or in systematic form [35]:

$$c(x) = m(x) \times x^{n-k} + ((m(x) \times x^{n-k}) \mod g(x))$$
(2)

Here g(x) is the generator polynomial, m(x) is the input information polynomial and c(x) is the codeword polynomial, which are all defined over an extension field GF (2^m) ($m \ge 1$). In Equation (2), *n* and *k* are the length of the codeword and input information, respectively, and k/n is the code rate. Obviously in Equations (1) and (2), the roots of g(x) are also the roots of c(x). We define the code roots to be the roots of the generator polynomial g(x) in this paper. The number of the elements in an extension field GF (2^m) is 2^m -1. We define the set of these elements as G_m . For a binary BCH code defined in GF (2^m) , the possible code roots, which form a root set A, are limited and included in G_m. A is a subset of G_m . We consider a sequence of M valid codewords C_1, C_2, \ldots, C_M and let $c_i(x)$ $(1 \le j \le M)$ be the codeword polynomial of C_i . Initialize an integer vector $[N_1, N_2, ..., N_M]$ to zeros and get all the roots of each codeword polynomial $c_i(x)$ when j increases from 1 to M. If the element α^i ($1 \le i \le 2^m - 1$) in GF(2^m) is a root of $c_i(x)$, we let $N_i = N_i + 1$, where α is a primitive element of GF (2^m) . Finally, the value of N_i $(1 \le i \le 2^m - 1)$ reflects the probability of the element α^{i} ($1 \le i \le 2^{m} - 1$) being a code root. Note that not all the roots of a valid codeword polynomial c(x) are the code roots, but all the code roots must be the roots of c(x). The elements, which are the roots of c(x) but not the code roots, appear randomly in different codewords, because the information polynomials are not the same in different encoding operations. According to this fact, the probability of being a code root for an element in A should be larger than that in \overline{A} , which is a complement of A in \mathbf{G}_m , so the values of $\{N_i \mid \alpha^i \in \mathbf{A}\}$ corresponding to the elements in \mathbf{A} are generally larger than those corresponding to the elements in \overline{A} , but this property is only true when the considered data blocks are valid codewords, *i.e.*, the code length is correct and the codewords synchronization is achieved. When the code length or the coding starting positions are not correctly estimated, the property described above does not exist and the bits in the codewords can be considered to appear randomly. In this case, the authors of [22,23] proposed the following hypothesis:

1708

Hypothesis 1: When the coding parameters are not correct, the probabilities of being codeword roots of the elements in \mathbf{G}_m can be assumed to be uniform, *i.e.*, the values of N_i is uniform no matter in \mathbf{A} or $\overline{\mathbf{A}}$.

Therefore, in the correct parameters cases, the entropy of the distribution of the roots should be lower than that in the incorrect parameters cases. Based on this fact, the authors of [22,23] introduced a root information dispersion entropy function (RIDEF) to describe the imbalance degree of the N_i on different α^i and proposed the correct coding parameters that maximize the RIDEF. The RIDEF is defined to be the difference between the entropy of the uniform distribution and the distribution of the code roots on all the elements in \mathbf{G}_m as follows:

$$\Delta H = -\sum_{i=1}^{n} \frac{1}{n} \log \frac{1}{n} - \left(-\sum_{i=1}^{n} p_i \log p_i \right) = \sum_{i=1}^{n} p_i \log p_i + \log n$$
(3)

where $n = 2^m - 1$ is the number of the elements in G_m . p_i is the probability of α^i being a root of the code, it is calculated as follows:

$$p_{i} = \frac{N_{i}}{\sum_{i=1}^{2^{m}-1} N_{i}}, 1 \le i \le 2^{m} - 1$$
(4)

In the recognition procedure, we traverse all the possible values of code length and coding starting positions to find the ones that make ΔH be the largest as the estimation of code length and synchronization positions.

The RIDERS algorithm has good performance [22–24] in the parameter recognition of BCH codes. However, there are still some problems in the algorithm. Firstly, it is restricted to hard decision situations, which limits the recognition performance. Secondly, Hypothesis 1, as a basis of the algorithm, is not always true. In the following paragraphs we propose the recognition methods inspired by the RIDERS algorithm in soft decision situations. In the Appendix, we give the proof of the faultiness of Hypothesis 1.

2.2. Principles of the Proposed Recognition Algorithm in Soft Decision Situations

2.2.1. Calculation of p_i in Soft Decision Situations

To develop the application of this algorithm to be suitable for soft decision cases, we should modify Equation (4) to utilize the soft output of the channel. To calculate p_i in Equation (4) in soft decision situations, we define the minimal parity check matrix (MPCM) $H_{\min}(\alpha^i)$ corresponding to the element α^i in GF(2^m) as follows:

$$H_{\min}(\alpha^{i}) = \left((\alpha^{i})^{l-1}, (\alpha^{i})^{l-2}, \cdots, (\alpha^{i})^{1}, (\alpha^{i})^{0} \right)$$
(5)

where *l* is the code length. According to the coding theories, if α^i is a root of a codeword $C_i (1 \le j \le M)$, we have:

$$H_{\min}(\alpha^{i}) \times C_{i} = 0 \tag{6}$$

1709

Soft outputs of the channel can provide more information about the reliability of each decision symbol. Instead of verifying whether α^i is a root of a codeword polynomial $c_j(x)$, we calculate $p_{j,i}(1 \le i \le 2^m - 1, 1 \le j \le M)$, which is the probability of α^i being a root of $c_j(x)$, and calculate p_i in Equation (4) as follows:

$$p_{i} = \frac{\sum_{j=1}^{M} p_{j,i}}{\sum_{i=1}^{2^{m}-1} \sum_{j=1}^{M} p_{j,i}}$$
(7)

To calculate $p_{j,i}$, we transform the MPCM defined in Equation (5) to its binary form by replacing the symbol elements in $H_{\min}(\alpha^i)$ with their binary column vector patterns. For example, $H_{\min}(\alpha^3)$, a BCH code corresponding to the element α^3 in GF(2³) with code length l = 7, is as follows:

$$H_{\min}(\alpha^3) = \begin{pmatrix} \alpha^{18} & \alpha^{15} & \alpha^{12} & \alpha^9 & \alpha^6 & \alpha^3 & 1 \end{pmatrix}$$
(8)

Based on the primitive polynomial $p(x) = x^3 + x + 1$, because the symbol α is a root of p(x), we have $\alpha^3 + \alpha + 1 = 0$, and $\alpha^3 = \alpha + 1$. Other symbols are processed similarly. Finally, we can calculate all the symbols in $H_{\min}(\alpha^3)$ and get their binary vector patterns listed in Table 1. By replacing the symbols in $H_{\min}(\alpha^3)$ with the binary patterns, the MPCM can be written in its binary form as follows:

$$Hb_{\min}(\alpha^{3}) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$
(9)

Table 1. Symbols in <i>l</i>	$H_{\min}(\alpha^3)$ and t	their binary vector	patterns.
------------------------------	----------------------------	---------------------	-----------

Symbols	Polynomial expressions	Vector form
1	1	001
α	α	010
α^{3}	$\alpha + 1$	011
α^{6}	α^2 +1	101
α^{9}	α^3	100
$lpha^{12}$	$\alpha^2 + \alpha + 1$	111
$lpha^{15}$	α	010
$lpha^{18}$	$\alpha^2 + \alpha$	110

Note that the number of rows in $H_{\min}(\alpha^i)$ equals to *m*, which is the degree of $GF(2^m)$. And then the syndromes corresponding to the MPCM $H_{\min}(\alpha^i)$ and codeword $C_j (1 \le j \le M)$ are calculated in binary forms as follows:

$$\mathbf{S}_{j,i} = [S_{j,1}, S_{j,2}, \cdots, S_{j,m}]^T = Hb_{\min}(\alpha^i) \times C_j$$
(10)

Let $c_j(x)$ be the codeword polynomial of C_j , if the element α^i corresponding to $Hb_{\min}(\alpha^i)$ in Equation (10) is a root of $c_j(x)$, then we have $\mathbf{S}_{j,i} = 0$. So the probability of $\mathbf{S}_{j,i} = 0$ can be considered as $p_{j,i}$ in Equation (7) and calculated by the mean probabilities of $S_{j,k} = 0$ ($1 \le k \le m$):

$$p_{j,i} = P_r(\mathbf{S}_{j,i} = 0) = \frac{1}{m} \sum_{k=1}^{m} P_r(S_{j,k} = 0)$$
(11)

In Equation (11) and the remainder of the paper we use $P_r(\Psi)$ to depict the probability of the event Ψ . We assume that the transmitter is sending a binary sequence of codewords and using a BPSK modulation, *i.e.*, let +1 and -1 be the modulated symbols of 0 and 1. The modulation operation from coded bit c_u to modulated symbol s_u can be written as:

$$s_u = 1 - 2c_u, \qquad u = 1, 2, 3, \dots$$
 (12)

We assume that the propagation channel is a Binary Symmetry Channel (BSC) which is corrupted by an AWGN with the variance $\sigma_n^2 = N_0/2$. For each configuration, the information symbols in the codes are randomly chosen. A soft decision symbol r_u at the reception can be expressed as:

$$r_u = s_u + w_u, \qquad u = 1, 2, 3, \dots$$
 (13)

where $(w_1, w_2, w_3,...)$ is an AWGN sequence. According to [30], it is easy to prove that the probability $P_r(S_{i,k} = 0)$ ($1 \le k \le m$) can be calculated as follows:

$$P_r(S_{j,k} = 0) = \frac{1}{2} + \frac{1}{2} \prod_{u=1}^{n_e} \tanh(r_u / \sigma^2), \quad 1 \le k \le m$$
(14)

where n_e is the number of ones in the e^{th} row of the binary MPCM $Hb_{min}(\alpha^i)$ $(1 \le i \le 2^m - 1)$.

2.2.2. Adaptive Processing of MPCM

Note that the matrix $Hb_{\min}(\alpha^i)$ is not sparse, so a fault decision symbol has negative influences on many syndromes. Considering that the unreliable decision bits have higher probabilities of being error decisions than the reliable decision bits, the authors of [28] proposed an adaptive processing algorithm for the binary MPCM $Hb_{\min}(\alpha^i)$ to reduce the influences of the unreliability decision bits when decoding the RS codes by utilizing the belief propagation (BP) algorithm. In this paper, we adopt that idea on the calculation of $P_r(\mathbf{S}_{j,i} = 0)$. The detailed adaptive processing steps for a given binary MPCM $Hb_{\min}(\alpha^i)$ and a codeword C_i with the code length *l* are listed below:

Step 1: Combine $Hb_{\min}(\alpha^{i})$ and C_{j}^{T} to form a new matrix $H^{*}(\alpha^{i})$ as follows:

$$H^{*}(\alpha^{i}) = \begin{bmatrix} \frac{r_{1}}{h_{11}} & \frac{r_{2}}{h_{12}} & \cdots & \frac{r_{l}}{h_{ll}} \\ h_{21} & h_{22} & \cdots & h_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ h_{m1} & h_{m2} & \cdots & h_{ml} \end{bmatrix}$$
(15)

1711

where r_1 , r_2 , ..., r_l are the soft decision bits of the codeword C_j , $\{h_{ij} | 1 \le i \le m, 1 \le j \le l\}$ are the elements of $Hb_{min}(\alpha^i)$ in GF(2).

Step 2: Replace each r_u $(1 \le u \le l)$ in $H^*(\alpha^i)$ with their absolute values to form a matrix $H_r^*(\alpha^i)$, adjust the positions of the columns in $H_r^*(\alpha^i)$ to make the first row in $H_r^*(\alpha^i)$ is ranked from the lowest to the highest and record the indexes. The absolute values of $\{r_u | 1 \le u \le l\}$ denote the reliability of the received soft decision bits. As shown in Equation (16), $|r_{i_1}| \le |r_{i_2}| \le \cdots \le |r_{i_l}|$ and i_1, i_2, \cdots, i_l are the column indexes of $r_{i_l}, r_{i_l}, \cdots, r_{i_l}$ in $H^*(\alpha^i)$.

$$H_{r}^{*}(\alpha^{i}) = \begin{vmatrix} |r_{i_{1}}| & |r_{i_{2}}| & \cdots & |r_{i_{l}}| \\ \hline h_{1i_{1}} & h_{1i_{2}} & \cdots & h_{1i_{l}} \\ h_{2i_{1}} & h_{2i_{2}} & \cdots & h_{2i_{l}} \\ \vdots & \vdots & \ddots & \vdots \\ h_{mi_{1}} & h_{mi_{2}} & \cdots & h_{mi_{l}} \end{vmatrix}$$
(16)

Step 3: Transform $H_r^*(\alpha^i)$ by elementary operations to make the last *m* elements of the first column in $H_r^*(\alpha^i)$ has only one "1" at the top, as shown in Equation (17). The first row does not join the elementary transformation.

$$H_{r}^{*}(\alpha^{i}) = \begin{bmatrix} |r_{i_{1}}| & |r_{i_{2}}| & \cdots & |r_{i_{l}}| \\ \hline 1 & x & \cdots & x \\ 0 & x & \cdots & x \\ \vdots & \vdots & \ddots & \vdots \\ 0 & x & \cdots & x \end{bmatrix}$$
(17)

This transformation limits the influences of the most unreliability decision bit to only one syndrome element, which is $S_{j,1}$ in Equation (10). Furthermore, we continue the elementary transformation on $H_r^*(\alpha^i)$ to limit the numbers of "1" in the following some columns to one (except the first row), as shown in Equation (18):

$$H_{r}^{*}(\alpha^{i}) = \begin{bmatrix} |r_{i_{1}}| & |r_{i_{2}}| & |r_{i_{3}}| & \cdots & |r_{i_{4}}| & |r_{i_{1}}| \\ \hline 1 & 0 & 0 & \cdots & x & x \\ 0 & 1 & 0 & \cdots & x & x \\ 0 & 0 & 1 & \cdots & x & x \\ \vdots & \vdots & \vdots & \ddots & x & x \\ 0 & 0 & 0 & \cdots & x & x \\ 0 & 0 & 0 & \cdots & x & x \end{bmatrix}$$
(18)

When the elementary transformation is unavailable, stop the operation. The number available operation times equal to the rank of $Hb_{\min}(\alpha^i)$. Then the last *m* rows in $H_r^*(\alpha^i)$ form a new matrix. We recovery its original column orders and call it $Hb_{\min_a}(\alpha^i)$. Because the transformation is elementary, the relationship $Hb_{\min_a} \times C_j = 0$, in the hard decision situations still exists if C_j is a valid codeword, so

we can calculate the probability $P_r(S_{j,k} = 0)$ defined in Equation (14) according to $Hb_{\min_a}(\alpha^i)$. This replacement reduces the influences of the unreliable decision bits.

If $rank(Hb_{\min_a}(\alpha^i)) = m$, the left $m \times m$ area of Hb_{\min_a} forms an identity matrix. But if $rank(Hb_{\min_a}(\alpha^i)) < m$, Hb_{\min_a} becomes the following style after the elementary transformations:

$$Hb_{\min_{a}a}(\alpha^{i}) = \begin{bmatrix} 1 & 0 & 0 & \cdots & x & x \\ 0 & 1 & 0 & \cdots & x & x \\ 0 & 0 & 1 & \cdots & x & x \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$
(19)

In this case, only the first *p* rows are non-zeros, where $p = rank(Hb_{\min_a}(\alpha^i))$. The last m - p rows have no contribution for the calculation of $P_r(\mathbf{S}_{j,i} = 0)$ in Equation (11). So we modify Equation (11) and Equation (20):

$$p_{j,i} = P_r(\mathbf{S}_{j,i} = 0) = \frac{1}{rank(Hb_{\min_a}(\alpha^i))} \sum_{k=1}^{rank(Hb_{\min_a}(\alpha^i))} P_r(S_{j,k} = 0)$$
(20)

Now based on Equation (20), Equation (14) and Equation (7), we can calculate the RIDEF defined in Equation (3). However, there is still a problem in the algorithm. Note that the element $\alpha^{2^m-1} = 1$ is a root of a codeword polynomial only if the code weight is even. This fact severely affects the assumption of uniformity of p_i . As shown in Figure 1, in the case of recognition of BCH (63, 51), when we assume the code length is 31, the value of p_{31} is obviously larger than other p values, although the real code length is not 31 and $\alpha^{31} = 1$ is not a root of the code.

Figure 1. The problem of p_{31} .



To avoid that, we drop the calculation of the probability $P_r(\mathbf{S}_{j,i} = 0)$ on the element $\alpha^{2^m-1} = 1$ when recognizing the code length. So we modify Equation (3) and Equation (7) to Equation (21) and Equation (22) respectively as follows:

$$\Delta H = \sum_{i=1}^{2^{m}-2} p_i \log p_i + \log(2^m - 2)$$
(21)

$$p_{i} = \frac{\sum_{j=1}^{M} p_{j,i}}{\sum_{i=1}^{2^{m}-2} \sum_{j=1}^{M} p_{j,i}}$$
(22)

Considering that not all the rows in the parity check matrix $Hb_{\min_a}(\alpha^i)$ have the same value of n_e [in Equation (14)], and this affects the comparability of $P_r(S_{j,k} = 0)$ among different rows. For the uniformization reasons, so we modify Equation (14) to:

$$P_{r}[S_{j,k} = 0] = \frac{1}{2} + \frac{1}{2} \operatorname{sign}\left[\prod_{u=1}^{n_{e}} \tanh(r_{u} / \sigma^{2})\right] \left|\prod_{u=1}^{n_{e}} \tanh(r_{u} / \sigma^{2})\right|^{\frac{1}{n_{e}}}$$

$$= \frac{1}{2} + \frac{1}{2} \prod_{u=1}^{n_{e}} \operatorname{sign}(r_{u}) \times \left|\prod_{u=1}^{n_{e}} \tanh(r_{u} / \sigma^{2})\right|^{\frac{1}{n_{e}}}$$
(23)

2.2.3. Summary of the Recognition Steps

According to Equation (20), Equation (21), Equation (22) and Equation (23), we propose the code length estimation and blind synchronization in the following steps for the convenience of computer program automatic processing:

Step 1: According to some prior information, set the searching range of the degree *m*, *i.e.*, set the minimal and maximal degree m_{min} and m_{max} .

Step 2: Design a window W which has a length L at least $5 \times (2^{m_{max}} - 1)$, *i.e.*, $M \ge 5$ in Equation (22).

Step 3: Full fill the window *W* with the received soft decision bits.

Step 4: Set the initial degree $m = m_{min}$.

Step 5: Set the code length $l = 2^{m-1}$.

Step 6: Set the initial synchronization position t at 0, which is the starting position of W.

Step 7: Assume the code length is *l* and the synchronization position is *t* and calculate ΔH . Note that the window *W* has more than one assumed codewords, we calculate the ΔH on all the codewords and compute the mean of them as $\Delta H(l,t)$.

Step 8: If t < l, then let t = t + 1 and go back to **Step 7**; if t = l, then jump to **Step 9**. **Step 9**: If $l < 2^m - 1$, then let l = l + 1 and go back to **Step 6**; if $l = 2^m - 1$, then jump to **Step 10**. **Step 10**: If $m < m_{max}$, then let m = m + 1 and go back to **Step 5**; if $m = m_{max}$, then jump to **Step 11**. Step 11: Compare all the calculated $\Delta H(l,t)$, select the maximum one and get the corresponding values of *l*, *t* and *m* as the estimated code length, synchronization position and the degree of the GF of the being recognized codes, respectively.

By traversing all the possible l and t, finally we can search out the parameters pair (\hat{l}, \hat{t}) that maximizes the RIDEF ΔH . \hat{l} and $\hat{t} + k\hat{l}(k \in \mathbb{Z})$ are the estimated code length and synchronization positions.

3. Code Roots Recognition and Generator Polynomial Reconstruction

3.1. Principles of the Code Roots Recognition

As mentioned in Section 2, for a given valid codeword C_j , the elements in **A** have higher probabilities of being the code roots of C_j than the elements in $\overline{\mathbf{A}}$, where **A** is the set of the roots of the generator polynomial. So after the code length and synchronization position estimation, we can compare the Log-Likelihood Ratios (LLR) of $P_r(\mathbf{S}_i = 0)$ on different elements in GF (2^m) from α^1 to α^{2^m-2} and choose the elements which make the LLR of $P_r(\mathbf{S}_i = 0)$ be obviously higher as the estimated code roots. Finally, we propose a method to judge whether the element $\alpha^{2^m-1} = 1$, be a root of the code.

In Equation (24), we define the LLR of $P_r(\mathbf{S}_i = 0)$ as follows, which is also written as $L(\alpha^i)$:

$$L(\alpha^{i}) = L[P_{r}(\mathbf{S}_{i}=0)] = \sum_{j=1}^{M} L[P_{r}(\mathbf{S}_{j,i}=0)] = \sum_{j=1}^{M} \left[\frac{1}{rank(Hb_{\min}(\alpha^{i}))} \sum_{k=1}^{rank(Hb_{\min}(\alpha^{i}))} L(S_{j,k}=0)\right]$$
(24)

where:

$$L(S_{j,k} = 0) = \log \frac{P_r(S_{j,k} = 0)}{P_r(S_{j,k} \neq 0)} = 2\operatorname{artanh}\left[\prod_{u=1}^{n_e} \operatorname{sign}(r_u) \times \left|\prod_{u=1}^{n_e} \operatorname{tanh}(r_u / \sigma^2)\right|^{\frac{1}{n_e}}\right]$$
(25)

But for a computer, the "previous higher" is difficult to follow. For the realization of the computer automatic recognition of the code roots, we propose the procedure includes the following steps:

Step1: Let *l* be the estimated code length, calculate the LLRs to form a vector: $L_{l} = \left[L(\alpha), L(\alpha^{2}), \dots, L(\alpha^{2^{m}-2}) \right].$

Step2: Rank the vector L_l from the lowest to the highest, in order to form a new vector L_{lR} , and record the indexes.

Step3: Calculate *dL*, which is the difference of L_{lR} : $dL(i) = L_{lR}(i+1) - L_{lR}(i)$ $(1 \le i \le l-1)$.

Step4: Find the maximum of dL, record the corresponding value of *i*.

Step5: Select the $(i + 1)^{th}$ to the l^{th} elements in L_{lR} , get their positions in the vector L_l and find corresponding GF elements $\{\alpha^{j_1}, \alpha^{j_2}, \cdots\}$ as the estimated roots.

As an example, we consider a BCH (63, 51) code which is corrupted by an AWGN with SNR $E_s/N_0 = 5$ dB, and the corresponding hard decision BER is $10^{-2.19}$. The recognizing procedure is shown in Figure 2.

Figure 2. Code roots recognition of BCH (63, 51). (a) Original LLRs. (b) Rank the vector L_l to form L_{lR} (c) dL: the difference of L_{lR} (d) Insert $L(\alpha^{2^m-1})$ into L_l .



Figure 2a is the original LLRs calculated in **Step 1**. By ranking the original LLRs from the lowest to the highest according to **Step 2**, we can get L_{iR} as shown in Figure 2b. We can see that the change between $L_{iR}(39)$ and $L_{iR}(40)$ is the largest, *i.e.*, the difference dL(50) on i = 50 in Figure 2c is the largest. Thus, we propose $L_{iR}(i)$ (i > 50) as the LLRs of the generator polynomial roots, which are α^1 , α^3 and their conjugate roots.

3.2. Discussion of the Element $\alpha^{2^{m-1}}$

Up to now, we have estimated the code roots from the set $\{\alpha^1, \alpha^2, \dots, \alpha^{2^m-2}\}$ but the element α^{2^m-1} is ignored. To verify whether the element $\alpha^{2^m-1} = 1$ is a root of the code, a method is calculating the LLR $L(\alpha^{2^m-1})$ according to its MPCM:

$$H_{\min}(\alpha^{2^{m}-1}) = \left(\left(\alpha^{2^{m}-1} \right)^{n-1-l_{s}} \left(\alpha^{2^{m}-1} \right)^{n-2-l_{s}} \cdots \left(\alpha^{2^{m}-1} \right)^{l} 1 \right)$$
(26)

But obviously, we have $H_{\min}(\alpha^{2^{m-1}}) = Hb_{\min}(\alpha^{2^{m-1}}) = Hb_{\min_{\alpha}}(\alpha^{2^{m-1}}) = (1 \ 1 \ \cdots \ 1)$, because $\alpha^{2^{m-1}} \equiv 1$. The MPCM is an overall-ones vector which provides very little information for the calculation of the LLR. We propose to create a new MPCM for the element $\alpha^{2^{m-1}}$ simply by the NOT of the minimal check matrix of one of the estimated roots. To get high reliability, we choose the root that has the highest LLR.

The parity check matrix of a code, which has k + 1 roots include $\alpha^{2^{m-1}}$, has a form as shown in Equation (27):

$$H = \begin{pmatrix} \alpha_1^{n-1} & \alpha_1^{n-2} & \cdots & \alpha_1 & 1 \\ \alpha_2^{n-1} & \alpha_2^{n-2} & \cdots & \alpha_2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha_k^{n-1} & \alpha_k^{n-2} & \cdots & \alpha_k & 1 \\ 1 & 1 & \cdots & 1 & 1 \end{pmatrix}$$
(27)

We choose one of its first k rows and its last row to form a new matrix H' as follows:

$$H' = \begin{pmatrix} \alpha_i^{n-1} & \alpha_i^{n-2} & \cdots & \alpha_i^{1} & 1\\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$
(28)

Then we transform H' to its binary pattern as follows:

$$H'_{bin} = \begin{pmatrix} x & x & \cdots & x & 1 \\ x & x & \cdots & x & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ x & x & \cdots & x & 1 \\ 1 & 1 & \cdots & 1 & 1 \end{pmatrix}$$
(29)

We define H'_{bin_f} to be any single row of H'_{bin} except the last row, and H'_{bin_l} to be the last row. For each valid codeword *C*, we have $H'_{bin} \times C = 0$. Thus, we have $(H'_{bin_f} \oplus H'_{bin_l}) \times C = 0$. Because all the elements in the last row of H'_{bin} are 1, the XOR and NOT operation is equivalent. Therefore, NOT $(H'_{bin_f}) \times C = 0$. As a result, the NOT of any row in the MPCM of one of the estimated roots is still a valid MPCM when the code has a root α^{2^m-1} .

Based on the new MPCM, we calculate the LLR on $\alpha^{2^{m-1}}$, and insert it into the vector L_l referred in **Step1**. Then we re-rank the LLRs and estimate the code roots according to the previous steps. Finally, we can write the generator polynomial based on all of the estimated code roots as:

$$g(x) = (x - \alpha_1)(x - \alpha_2) \cdots (x - \alpha_p)$$
(30)

where $\alpha_1, \alpha_2, ..., \alpha_p$ are the estimated code roots. In the example of the recognition of BCH (63, 51) code referred previous as shown in Figure 2, we insert $L(\alpha^{2^{m-1}})$ into L_l and draw the stems of the LLRs in Figure 2(d). Re-execute the code roots recognition steps, it is easy to verify that the element $\alpha^{2^{m-1}}$ is not a root of the code and the case of Figure 1 does not appear in this algorithm. Thus the generator polynomial is as follows:

$$g(x) = (x - \alpha)(x - \alpha^{2})(x - \alpha^{4})(x - \alpha^{8})(x - \alpha^{16})(x - \alpha^{32})$$

$$(x - \alpha^{3})(x - \alpha^{6})(x - \alpha^{12})(x - \alpha^{24})(x - \alpha^{48})(x - \alpha^{33})$$

$$x^{12} + x^{10} + x^{8} + x^{5} + x^{4} + x^{3}$$
(31)

4. Primitive Polynomial Recognition

In Section 2 and Section 3, the primitive polynomial of the being recognized code is not considered. But in fact, the corresponding primitive polynomial should be given when discussing an extension field $GF(2^m)$, because there are more than one primitive polynomials in $GF(2^m)$ and the calculation rules based on different primitive polynomials are not the same. However, we propose a theorem that a binary cyclic codeword based on a primitive polynomial is also a valid binary cyclic codeword based on another primitive polynomial. According to this, we can choose any one primitive polynomial p(x) randomly and estimate the code length and code roots based on p(x). Then, we can recognize the actual primitive polynomial according to the root properties of the BCH codes.

Theorem 1: A binary cyclic codeword C_r , which is encoded based on a primitive polynomial $p_1(x)$ over $GF(2^m)$, is also a valid binary cyclic codeword based on another primitive polynomial $p_2(x)$ over $GF(2^m)$ with the same number of code roots.

Proof. The coefficients of a generator polynomial g(x) of a binary cyclic code is in GF(2) and can be factored into the product of some minimal polynomials over GF(2):

$$g(x) = m_1(x)m_2(x)\cdots m_p(x)$$
 (32)

Let $GF_1(2^m)$ and $GF_2(2^m)$ be extension fields based on two different primitive polynomials $p_1(x)$ and $p_2(x)$, respectively, then $GF_1(2^m)$ and $GF_2(2^m)$ have the same structure. Each minimal polynomial $m_i(x)$ $(1 \le i \le p)$ can be factored in the extension fields $GF_1(2^m)$ and $GF_2(2^m)$ both. We define α and β to be the roots of $p_1(x)$ and $p_2(x)$, respectively. According to Theorem 2.18 in [36], let e1 and e2 be the smallest integers such that $\alpha^{2^{e1}} = 1$ and $\beta^{2^{e^2}} = 1$, respectively, then we have:

$$m_{i}(x) = \prod_{j=0}^{e^{1-1}} \left(x + \alpha^{2^{j}} \right) = \prod_{j=0}^{e^{2-1}} \left(x + \beta^{2^{j}} \right)$$
(33)

Since a minimal polynomial has only one distinct root, then according to Equation (33), we have e1 = e2, *i.e.*, each minimal polynomial has the same number of conjugate roots, even if it is based on different primitive polynomials. Consequently, a codeword, which is based on a primitive polynomial $p_1(x)$, is also a valid codeword based on any other primitive polynomial $p_2(x)$, and the number of code roots and error-correcting capabilities are the same. Therefore, when recognizing the parameters of a binary cyclic code, we can just randomly choose a primitive polynomial provisionally. In order to reduce the computational complexity, we recommend choosing the primitive polynomial with the smallest number of terms.

According to the basic character of the BCH codes, a generator polynomial has 2t roots with consecutive degrees (the 2t roots do not form all the code roots, but include all the distinct roots), where *t* is the correction capability of the codes. In other words, if α is a primitive element in GF(2^m), the generator polynomial g(x) of a BCH code for correcting *t* errors has α , α^2 , α^3 , ..., α^{2t} as its roots [3]:

$$g(\alpha^{i}) = 0, \quad i = 1, 2, ..., 2t$$
 (34)

After the estimation of code roots based on a randomly chosen primitive polynomial p(x) and the estimated degree *m*, we can calculate the number of roots and the correction capability *t*. Then, we can traverse all the primitive polynomials over $GF(2^m)$ and get the one that makes the code roots be in accordant with the character of BCH codes as the primitive polynomial of the being recognized code.

As an example, we still consider the BCH(63, 51) code referred in Section 3. The codewords are encoded based on the primitive polynomial $p_1(x) = x^6 + x + 1$ and the code roots are α , α^2 , α^4 , α^8 , α^{16} , α^{32} , α^3 , α^6 , α^{12} , α^{24} , α^{48} , and α^{33} . The first six ones and the last six ones are two groups of

conjugate roots. Here α is a primitive root in GF(2⁶) based on the primitive polynomial $p_1(x)$, *i.e.*, $p_1(\alpha) = 0$. But in the recognition procedure, we do the calculations over GF(2⁶) with another primitive polynomial $p_2(x) = x^6 + x^5 + 1$ and let the symbol β be a root of $p_2(x)$. As a result, we can get the code roots β^{15} , β^{30} , β^{60} , β^{57} , β^{51} , β^{39} , β^{31} , β^{62} , β^{61} , β^{59} , β^{55} , and β^{47} after the recognition as shown in Figure 3. It is easy to verify that the estimated code roots also form two groups of conjugate roots, thus the error-correction capability *t* equals two. Now we traverse all the other primitive polynomials and find the one under which the code roots are in accordant with the characters of BCH codes.

Figure 3. Recognizing the BCH(63, 51) codes using a primitive polynomial different from that of the encoder.



5. Computational Complexity

In the proposed soft decision recognition algorithm, the most complex computation is the calculation of p_i . The major computational consumption appears in Equation (23), which includes the n_e^{th} root calculations, productions and *tanh* function in the real number field. While, the hard decision algorithm only has production and addition calculations over GF(2). However, our proposed algorithm utilizes the soft decision outputs of the channel, which can provide more information about the reliability decision bits, so require very lower calculation times of $p_{j,i}$ than the hard decision one, which can also reduce the total computational complexity in the recognition procedure.

6. Simulations

The simulation results of the proposed blind recognition algorithm are shown in this section. In the simulations, we assume that the propagation channel is a Binary Symmetry Channel (BSC) which is corrupted by an AWGN with the variance $\sigma_n^2 = N_0/2$. For each configuration, the information symbols in the codes are randomly chosen and the modulation mode is BPSK. All the simulations have the same settings of the observation window with length L = 3,000 bits and the searching range of the code length is 7–127.

When applying the algorithm to BCH(63,45) codes, the simulation results of the code length estimation are shown in Figures 4–6. The signal is corrupted by an AWGN with the SNR $E_s/N_0 = 5$ dB.

In Figure 4, we draw the stems of p_i on different elements α^i taken from $GF(2^m)$ when the code length and synchronization positions are correct. It is shown in Figure 4 that the values of p_i on the code roots are obviously higher than those on the other elements. And we also draw the stems on a fault code length and synchronization position in Figure 5.



Figure 4. Values of p_i on correct code length and synchronization positions.

Comparing with Figure 4, the values of p_i in Figure 5 is uniform, so the information entropy of p_i on correct coding parameters should be lower and the corresponding dispersion entropy function is higher. As shown in Figure 6, we draw the stems of RIDEF on different code length l and coding starting positions t when the start position t = 0 of the observation window is at the fortieth bit of a codeword. In the figure, the value of p_i in the case of l = 63 and t = 23 is the highest, thus we consider the parameters l = 63 and $t = 23k(k \in \mathbb{Z})$ as the estimated code length and synchronization positions. The result is in accordance with the simulation settings.

Figure 5. Values of p_i on incorrect code length and synchronization positions.





Figure 6. Code length and synchronization positions estimation of BCH(63, 45) codes.

The performance of the algorithm is affected by the channel quality. In Figure 7, we draw the performance of the proposed algorithm when applied to code length and synchronization positions recognitions for several different binary BCH codes, including the shortened codes. The curves depict the false recognition probabilities (FRP) of the code length and coding starting positions estimations on different SNRs. We also compare the performance of our proposed recognition algorithm with the hard-decision-based RIDERS algorithm proposed in [22–24]. The PFR of our proposed algorithm fall rapidly when SNR increases, and it is much lower than that of the previous hard decision algorithms on each single SNR value.

After the code length and synchronization position estimation, the generator polynomial can be recognized by searching for the code roots according to the steps proposed in Section 3. Figure 8 shows the performance of the proposed generator polynomial recognition algorithm when applied to several different binary BCH codes, which are BCH(63, 51), BCH(63, 39) and BCH(30, 20). The curves show the false recognition probabilities on different noise levels. As E_s/N_0 rises, the curves fall rapidly. When E_s/N_0 is above 5 dB, no false recognition occurred during our 200,000 instances of simulation. We also compare our proposed algorithm with the previous hard-decision-based recognition algorithms proposed by [22–24] in the figure. It shows that the recognition performance is improved obviously in soft decision situations. A gap of 1–2 dB exists between the two groups of curves.



Figure 7. Performances of code length estimations for some binary BCH codes.

Figure 8. Performances of code roots estimations for some binary BCH codes.



7. Conclusions

A soft-decision-based blind recognition method for binary BCH codes with BPSK modulations on AWGN channels is proposed aiming at the non-cooperative communications and ACM techniques. The code length estimation and block synchronization are achieved by checking the minimal parity check matrix. After that, the code rate and generator polynomials are reconstructed by searching for the code roots. To the best of our knowledge, this paper is the first publication in literature which introduces an approach for complete-blind recognition of binary BCH codes in soft decision situations. Simulations show that our proposed blind recognition algorithm yields better performance than that of the previous hard-decision-based ones.

Appendix: Proof of the faultiness of the Hypothesis 1

In this Appendix, we present that the Hypothesis 1 proposed in [22–24] is not always correct. The proof is shown below:

Proof. We can consider a codeword *C* with length *n* and the binary pattern of one of the MPCM $Hb_{\min}(\alpha^i)$, where $\alpha^i (0 \le i \le 2^m - 1)$ is an element in \mathbf{G}_m . We let c(x) be the codeword polynomial of *C*. If α^i is a root of c(x), then we have $c(\alpha^i) = 0$ and $Hb_{\min}(\alpha^i) \times C = 0$. There are *m* rows in $Hb_{\min}(\alpha^i)$ and we define $\mathbf{h}_j (1 \le j \le m)$ to be the j^{th} row of $Hb_{\min}(\alpha^i)$. Then the equation $Hb_{\min}(\alpha^i) \times C = 0$ means the productions of all the rows with the codeword *C* equal to zeros, as shown in Equation (35):

$$Hb_{\min}(\alpha^{i}) \times C = 0 \Leftrightarrow \begin{cases} \boldsymbol{h}_{1} \times C = 0 \\ \boldsymbol{h}_{2} \times C = 0 \\ \vdots \\ \boldsymbol{h}_{m} \times C = 0 \end{cases}$$
(35)

So we can calculate the probability of α^i being the root of c(x), *i.e.*, the probability of $Hb_{\min}(\alpha^i) \times C = 0$ as follows:

$$P_r \left[Hb_{\min}(\alpha^i) \times C = 0 \right] = P_r \left(\boldsymbol{h}_1 \times C = 0, \boldsymbol{h}_2 \times C = 0, \dots, \boldsymbol{h}_m \times C = 0 \right)$$
(36)

Let $h_{jl}(1 \le l \le n)$ and C_l be the l^{th} element in the vector h_j and C and we define the *checking indexing* set **S**_i for h_i and C_l as follows:

$$\mathbf{S}_{i} = \{C_{i} \mid h_{il} = 1\}$$
(37)

Obviously, when the number of nonzero elements in S_i is even, we have:

$$\boldsymbol{h}_i \times \boldsymbol{C} = \boldsymbol{0} \tag{38}$$

And when the number of nonzero elements in S_i is odd, we have:

$$\boldsymbol{h}_{i} \times \boldsymbol{C} = 1 \tag{39}$$

When the code length and synchronization positions are not estimated correctly, the restriction among the elements in *C* does exist. So the elements in the codewords can be considered to appear randomly. In this case, the probabilities of the number of nonzero elements in S_j being odd and even are all about 0.5. When $Hb_{min}(\alpha^i)$ is full rank, the rows of $Hb_{min}(\alpha^i)$ is linearly independent, so we can calculate Equation (36) as follows:

$$P_r \Big[Hb_{\min}(\alpha^i) \times C = 0 \Big] = \prod_{j=1}^m P_r \left(\boldsymbol{h}_j \times C = 0 \right) = (0.5)^m$$
(40)

But if $Hb_{\min}(\alpha^i)$ is not full rank, the calculation of Equation (36) is not correct. We define the *maximum linearly independent vector group* **MI** of the row vectors set $\mathbf{H} = \{\mathbf{h}_i | 1 \le j \le m\}$ as follows:

MI is a subset of **H** and meets the following conditions:

- (1) The vectors in **MI** is linearly independent;
- (2) Any vector in **H** can be obtained by linear combinations of the vectors in **MI**.

 $\langle \mathbf{a} \mathbf{a} \rangle$

And it is easy to prove that the number of vectors in **MI** is the rank of $Hb_{\min}(\alpha^i)$.

According to the condition 2 of the definition of **MI**, if all the vectors in $\{\mathbf{h}_j | \mathbf{h}_j \in \mathbf{MI}\}$ make $\mathbf{h}_j \times C = 0$, then also for all the vectors in $\{\mathbf{h}_j | \mathbf{h}_j \in \mathbf{H}\}$, we have $\mathbf{h}_j \times C = 0$, so the calculation of Equation (36) should be:

$$P_r \Big[Hb_{\min}(\alpha^i) \times C = 0 \Big] = \prod_{t=1}^{rank \left(Hb_{\min}(\alpha^i) \right)} P_r \left(\boldsymbol{h}_{jt} \times C = 0 \right) = (0.5)^{rank \left(Hb_{\min}(\alpha^i) \right)}$$
(41)

where $\{ \mathbf{h}_{jt} | 1 \le t \le rank(Hb_{min}(\alpha^{i})) \}$ are the vectors in **MI**, *i.e.*, a maximum linearly independent vector group of the rows of $Hb_{min}(\alpha^{i})$.

According to Equation (41), **Hypothesis 1** is true only if all the $Hb_{min}(\alpha^i)$, where $1 \le i \le 2^m - 1$, have the same rank. But unfortunately, this condition cannot always be met. For example, when considering the BCH(63, 51) codes, we have the following results:

$$\begin{cases} rank (Hb_{\min}(\alpha^{1})) = 6 \\ rank (Hb_{\min}(\alpha^{9})) = 3 \\ rank (Hb_{\min}(\alpha^{63})) = 1 \\ \dots \end{cases}$$
(42)

Therefore, we have:

$$\begin{cases}
P_r \Big[Hb_{\min}(\alpha^1) \times C = 0 \Big] = \left(\frac{1}{2}\right)^6 \\
P_r \Big[Hb_{\min}(\alpha^1) \times C = 0 \Big] = \left(\frac{1}{2}\right)^3 \\
P_r \Big[Hb_{\min}(\alpha^1) \times C = 0 \Big] = \left(\frac{1}{2}\right)^1 \\
\dots
\end{cases}$$
(43)

so we find that the **Hypothesis 1** is not correct.

References

- Lin, S.; Costello, D.J. Coding for reliable digital transmission and storage. In *Error Control Coding: Fundamentals and Applications*, 2nd ed.; Horton, J.M., Riccardi, D.W., Eds.; Pearson Prentice Hall: Upper Saddle River, NJ, USA, 2004; pp. 1–23.
- Wang, X.; Xiao, G.Z. BCH codes and Goppa codes. In *Error correcting coding: principles and methods*; Li, J., Ed.; Xidian University Press: Xian, China, 2001; pp. 242–317.
- Moreira, J.C.; Farrell, P.G. BCH Codes. In *Essentials of Error-Control Coding*; John Wiley & Sons Ltd.: Chichester, UK, 2006; pp. 97–112.
- 4. Peterson, W.W. Encoding and error-correction procedures for Bose-Chaudhuri codes. *IRE Trans. Inf. Theory* **1960**, *9*, 459–470.

- 5. Chien, R.T. Cyclic Decoding procedure for the Bose-Chaudhuri-Hocquenghem Codes. *IEEE Trans. Inf. Theory* **1964**, *10*, 357–363.
- 6. Mitola, J.; Maguire, G.Q. Cognitive radio: making software radios more personal. *IEEE Personal Commun.* **1999**, *6*, 13–18.
- 7. Oreay, O.; Ustundag, B. A pattern construction scheme for neural network-based cognitive communication. *Entropy* **2011**, *13*, 64–81.
- 8. Li, M.; Batalama, N.S.; Pados, A.; Melodia, T.; Medley, M.J.; Matyjas, J.D. Cognitive code-division links with blind primary-system identification. *IEEE Trans. Wireless Commun.* **2011**, *11*, 3743–3753.
- 9. Liu, Y.; Tan, X.; Anghuwo, A.A. Joint power and spectrum allocation algorithm in cognitive radio networks. *J. Syst. Eng. Electron.* **2011**, *4*, 691–701.
- 10. Jiang, T.; Grace, D.; Mitchell, P.D. Efficient exploration in reinforcement learning-based cognitive radio spectrum sharing. *IET Commun.* **2011**, *10*, 1309–1317.
- 11. Marazin, M.; Gautier, R.; Burel, G. Algebraic method for blind recovery of punctured convolutional encoders from an errorneous bitstream. *IET Signal Process.* **2012**, *2*, 122–131.
- Moosavi, R.; Larsson, E.G. A fast scheme for blind identification of channel codes. In Proceedings of 54th GLOBECOM, Houston, TX, USA, 5–9 December 2011; pp. 1–5.
- Marazin, M.; Gautier, R.; Burel, G. Dual code method for blind identification of convolutional encoder for cognitive radio receiver design. In Proceedings of IEEE Globecom Workshops, Honolulu, HI, USA, 30 November–4 December 2009; pp. 1–6.
- 14. Goldsmith, A.J.; Chua, S.G. Adaptive coded modulation for fading channels. *IEEE Trans. Commun.* **1998**, *5*, 595–602.
- Burel, G.; Gautier, R. Blind estimation of encoder and interleaver characteristics in a non cooperative context. Presented at International Conference on Communications, Internet and Information Technology, Scottsdale, AZ, USA, 17–19 November 2003.
- Choqueuse, V.; Marazin, M.; Collin, L.; Yao, K.C.; Burel, G. Blind reconstruction of linear spacetime block codes: a likelihood-based approach. *IEEE Trans. Signal Process.* 2010, *3*, 1290–1299.
- 17. Marazin, M.; Gautier, R.; Burel, G. Blind recovery of k/n rate convolutional encoders in a noisy environment. *EURASIP J. Wireless Commun. Netw.* **2011**, *168*, 1–9.
- Wang, F.; Huang, Z.; Zhou, Y. A method for blind recognition of convolution code based on Euclidean algorithm. In Proceedings of IEEE WiCom, Shanghai, China, 21–25 September, 2007; pp. 1414–1417.
- 19. Dignel, J.; Hagenauer, J. Parameter estimation of a convolutional encoder from noisy observations. In Proceedings of IEEE ISIT, Nice, France, 24–29 June 2007; pp. 1776–1780.
- 20. Marazin, M.; Gautier, R.; Burel, G. Blind recovery of the second convolutional encoder of a turbocode when its systematic outputs are punctured. *Military Tech. Acad. Rev.* **2009**, *2*, 213–232.
- 21. Yongguang, Z. Blind recognition method for the Turbo coding parameters. *J. Xidian Univ.* **2011**, *2*, 167–172.
- 22. Wen, N.; Yang, X. Recognition methods of BCH codes. *Electron. Warf.* 2010, *6*, 30–34.
- 23. Yang, X.; Wen, N. Recognition method of BCH codes on roots information dispersion entropy and roots statistic. *J. Detect. Control.* **2010**, *3*, 69–73.
- 24. Lv, X.; Huang, Z.; Su, S. Fast recognition method of generator polynomial of BCH codes. *J. Xidian Univ.* **2011**, *6*, 187–191.

- 25. Fossorier, M.; Lin, S. Bit error probability for maximum likelihood decoding of linear block codes and related soft decision decoding methods. *IEEE Trans. Inf. Theory.* **1998**, *11*, 3083–3090.
- 26. Kaneko, T.; Nishijima, T.; Inazumi, H.; Hirasawa, S. An effifient maximum likelihood decoding of linear block codes with algebraic decoder. *IEEE Trans. Inf. Theory.* **1994**, *3*, 320–327.
- 27. Sankaranarayanan, S.; Vasic, B. Iterative decoding of linear block codes: A parity-check orthogonalization approach. *IEEE Trans. Inf. Theory* **2005**, *51*, 3347–3353.
- 28. Jiang, J.; Narayanan, K.R. Iterative soft-input-soft-output decoding of Reed-Solomon codes by adapting the parity check matrix. *IEEE Trans. Inf. Theory.* **2006**, *8*, 3746–3756.
- 29. Ni, L.; Yao, F.; Zhang, L. A rotated quasi-othogonal space-time block code for asynchronous cooperative discovery. *Entropy* **2012**, *14*, 654–664.
- 30. Hagenauer, J.; Offer, E.; Papke, L.; Iterative decoding of binary block and convolutional codes. *IEEE Trans. Inf. Theory* **1996**, *2*, 429–445.
- Imad, R.; Sicot, G.; Houcke, S. Blind frame synchronization for error correcting codes having a sparse parity check matrix. *IEEE Trans. Commun.* 2009, *6*, 1574–1577.
- 32. Imad, R.; Houcke, S. Theoretical analysis of a MAP based blind frame synchronizer. *IEEE Trans. Wireless Commun.* **2009**, *11*, 5472–5476.
- Imad, R.; Houcke, S.; Jego, C. Blind frame synchronization of product codes based on the adaptation of the parity check matrix. In Proceedings of IEEE ICC2009, Dresden, Germany, 14– 18 June, 2009; pp. 1574–1577.
- Imad, R.; Poulliat, C.; Houcke, S.; Gadat, G. Blind frame synchronization of Reed-Solomon codes: Non-binary vs. binary approach. In Proceedings of 2010 IEEE Eleventh International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Marrakech, Morocco, 20–23 June 2010; pp. 1–5.
- Moreira, J.C.; Farrell, P.G. Cyclic codes. In *Essentials of Error-Control Coding*; John Wiley & Sons Ltd.: Chichester, UK, 2006; pp. 81–94.
- Lin, S.; Costello, D.J. Introduction to algebra. In *Error Control Coding: Fundamentals and Applications*, 2nd ed.; Horton, J.M., Riccardi, D.W., Eds.; Pearson Prentice Hall: Upper Saddle River, NJ, USA, 2004; pp. 25–65.

 \bigcirc 2013 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/3.0/).