Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 798612, 9 pages
http://dx.doi.org/10.1155/2014/798612

Research Article

Blind Identification of Convolutional Encoder Parameters

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

School of Mechatronics Engineering and Automation, National University of Defense Technology, Deya Road, Changsha, Hunan 410073, China

Correspondence should be addressed to Zhiping Huang; h.zhiping@hotmail.com

Received 13 March 2014; Accepted 5 May 2014; Published 21 May 2014

Academic Editor: Lei Cao

This paper gives a solution to the blind parameter identification of a convolutional encoder. The problem can be addressed in the context of the noncooperative communications or adaptive coding and modulations (ACM) for cognitive radio networks. 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 convolutional codes, while the coding parameters are unknown. Some previous literatures have significant contributions for the recognition of convolutional encoder parameters in hard-decision situations. However, soft-decision systems are applied more and more as the improvement of signal processing techniques. In this paper we propose a method to utilize the soft information to improve the recognition performances in soft-decision communication systems. Besides, we propose a new recognition method based on correlation attack to meet low signal-to-noise ratio situations. Finally we give the simulation results to show the efficiency of the proposed methods.

1. Introduction

In digital communication systems, error-correction codes are widely used. To meet high quality of services, new coding schemes are being developed ceaselessly. Therefore, for a communication receiver, it is very difficult to remain compatible with all standards used. But if it is an intelligent receiver, which is able to blindly recognize the coding parameters of a specific transmission context, it can adapt itself to the perpetual evolution of digital communications. Furthermore, the blind recognition techniques are also applied in noncooperative communications. In noncooperative communication contexts, a receiver does not know the coding parameters, so it must blindly recover the encoder before decoding. In this paper we focus on the blind recognition of coding parameters of an encoder which uses convolutional codes as error-correction coding and propose a method to take advantage of the soft information in soft-decision situations.

Some previous literatures discussed the problem of blind recognition of convolutional codes. The authors of [1–3] developed recognition methods in noiseless context, including the rate $1/n$ and k/n codes, where n denotes the codeword length and k the dimension. These methods are not suitable for noisy environment. For the case of noisy context, some algorithms were proposed in recent years [4–7]. The algorithm proposed in [7], which we call a Gauss-Jordan elimination through pivoting (GJETP) based algorithm in this paper, completely solved the blind parameter recognition of k/n convolutional codes with low complexity and excellent recognition performances.

However, the previous works are all discussed in hard-decision situations. In modern communication systems, more and more soft-decision based algorithms are applied to improve the signal processing performances. For example, the soft-decision based decoding methods always have better performances than hard-decision situations [8–12]. Similarly, some soft-decision based blind recognition of block code parameters also outperforms the hard-decision one [13, 14].

In Section 2 of this paper, based on [6, 7] we propose a method to utilize the soft information to improve the recognition performances in soft-decision systems. And we give another two improvements about the recognition algorithm in Sections 3 and 4: (1) because the authors of [6, 7] did not give a normative algorithm to automatically identify the parameter n for a computer program, we propose a formalized scheme to optimize the recognition of n; (2) the authors of [6, 7] did not consider the synchronization of codewords while the synchronization positions are usually

unknown in a noncooperative communication context; we propose to modify the algorithm to identify the codeword synchronization.

Besides, in Section 5 we propose a correlation attack method to meet very low signal-to-noise ratio (SNR) situations, which also includes both hard- and soft-decision algorithms. This method might require more computational time, but it can push the SNR limits of the GJETP-based algorithm proposed in [7]. And we propose some strategies to reduce the searching space according to the structural analysis of the dual codes.

Finally we show the efficiency of the proposed algorithm by computer simulations in Section 6 and conclude the paper in Section 7.

2. Utilizing the Soft Information

The details of the GJETP-based recognition of rate k/n convolutional encoder parameters are introduced in [7]. The algorithm includes three major sections:

(1) identification of code length n;

(2) identification of a dual code basis;

(3) identification of the generator matrix.

In the first procedure, the authors of [7] propose to recognize parameter n using the following steps.

Step 1. Set the minimum and maximum value of l, that is, $l_{\min}$ and $l_{\max}$. Initialize l to be $l_{\min}$.

Step 2. According to l and a sequence of received symbols, we generate an observed data matrix $\mathbf{R}_l$ as shown in Figure 1, the numbers in which denote the arriving order of the bits in the received data stream.

Step 3. Transform the matrix $\mathbf{R}_l$ to a lower triangular matrix $\mathbf{G}_l$ by GJETP (see [15] for details):

$$\mathbf{G}_l = \mathbf{A}_l \mathbf{R}_l \mathbf{B}_l. \tag{1}$$

Step 4. Obtain the set $\mathbf{Z}_l$ as follows:

$$\mathbf{Z}_l = \text{Card}\left\{ i \in \{1, \ldots, l\} \mid N_l(i) \le \frac{L-l}{2}\gamma_{\text{opt}} \right\}, \tag{2}$$

where $N_l(i)$ is the number of "1" in the lower part of the ith column in the matrix $\mathbf{G}_l$, γ_{opt} is an optimal threshold [6, 7], and Card$\{x\}$ denotes the cardinal of x. An $N_l(i)$ smaller than $(L-l)\gamma_{\text{opt}}/2$ indicates that the ith column of $\mathbf{R}_l$ has a high probability to be dependent on the other columns.

Step 5. If $l = l_{\max}$, execute Step 6. If $l < l_{\max}$, let $l = l + 1$ and go back to Step 2.

Step 6. Output the gap between two consecutive nonzero cardinals, $\mathbf{Z}_l$, as the estimated codeword size $\hat{n}$. The principle of this estimation is that when $l = mn (m \in \mathbf{Z}^+)$ and l is larger than a parameter n_a (see [7] for details), some columns of $\mathbf{C}_l$ can be determined by other columns, where $\mathbf{C}_l = \mathbf{R}_l + \mathbf{E}_l$ and $\mathbf{E}_l$ is the error pattern corresponding to $\mathbf{R}_l$.

According to the GJETP algorithm, the reliability of upper part of $\mathbf{R}_l$ has larger influence on the successful detection of dependent columns in $\mathbf{R}_l$. So if we can make the upper part of $\mathbf{R}_l$ have lower errors, the algorithm can be improved. In hard-decision situations, we cannot determine which rows in $\mathbf{R}_l$ have lower probabilities to have error decision bits. But in soft-decision situations, we can examine the reliabilities of the rows in $\mathbf{R}_l$ according to the soft-decisions bits. Therefore, here we propose the following processing procedure inserted into the steps (between Step 2 and Step 3) mentioned above.

(1) Fill the observed matrix $\mathbf{R}_l$ with the soft-decision bits. For each row in $\mathbf{R}_l$ we find the decision bit that has the lowest reliability and record its reliability value as the reliability of the row.

(2) Arrange the rows of $\mathbf{R}_l$ according to each row's lowest reliability value to make the first row of $\mathbf{R}_l$ have the highest reliability, the second row have the second highest reliability, and so on.

(3) Obtain the hard decisions of the bits in the rearranged $\mathbf{R}_l$ and continue Step 3 mentioned above.

After this processing, the upper rows of the rearranged $\mathbf{R}_l$ have higher reliabilities and therefore have lower probabilities to include error decision bits. So the probability of successful detection of dependent columns can be improved. And as the value of L rises, the improvement of the reliabilities of the upper part of $\mathbf{R}_l$ becomes more effective, so the recognition performance can be improved more obviously than the previous algorithm in [7]. After the estimation of n, we use Algorithms 1 and 2 in [7] to identify other coding parameters. Note that the parity check vector recognition procedure uses the GJETP algorithm too, so the proposed rearranged steps are also applied before each GJETP processing when recognizing the parity check vectors.

3. Formalizing the Estimation Algorithm of n

Note that, in Step 6 shown in the previous section, the authors of [7] only give the idea of the estimation of n. When the noise level is high, there exists a problem that the number of error decision bits is high and some cardinals $\mathbf{Z}_l$ corresponding to the values of l that equals mn are blank. So not all the gaps between two consecutive nonzero cardinals, $\mathbf{Z}_l$, equal n. In this case, we need a formalized algorithm to estimate the value of n more exactly. This can be done by simply searching all the gaps between two consecutive nonzero cardinals and find out which gap value appears mostly. The detailed algorithm steps are listed below.

(1) Let $\mathbf{V}_l = [l_1, l_2, \ldots, l_s]$ be the vector that consists of the values of l corresponding to the detected nonzero cardinals $\mathbf{Z}_l$ in Step 6.

(2) Calculate the vector $\Delta\mathbf{V}_l = [\Delta l_1, \Delta l_2, \ldots, \Delta l_{s-1}]$, the elements of which are calculated by

$$\Delta l_i \ (1 \le i \le s-1) = l_{i+1} - l_i. \tag{3}$$

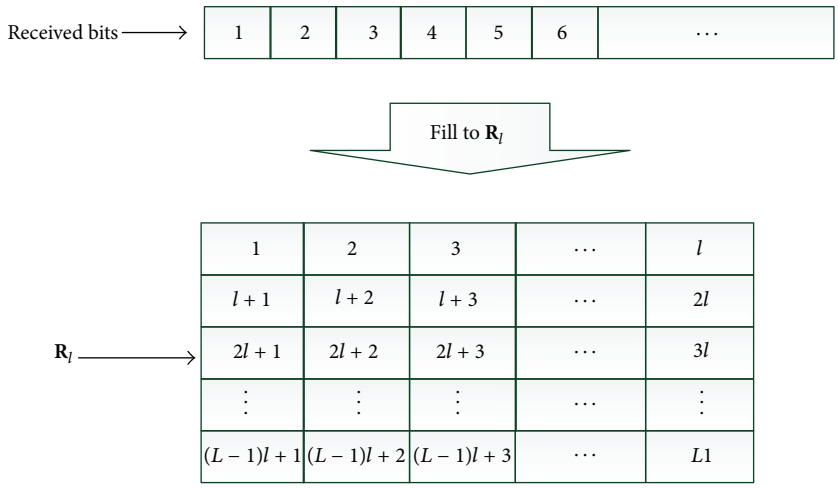

Figure 1: Generation of the observed matrix $\mathbf{R}_l$.

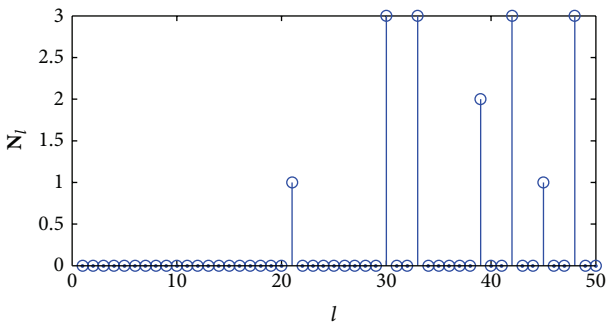

Figure 2: The recorded vector $\mathbf{N}_l$ based on low SNR.

(3) Initialize a vector $\mathbf{Q}$ with length $l_{\max}$ to be overall zeros and for each value of i from 1 to $s-1$, we let

$$\mathbf{Q}(\Delta l_i) = \mathbf{Q}(\Delta l_i) + 1. \tag{4}$$

(4) Finally we find the maximum element in $\mathbf{Q}$ and output the corresponding gap value Δl_i to be the estimated codeword size $\widehat{n}$.

After taking the previously mentioned searching method we can find the most probable gap between two consecutive nonzero cardinals, that is, the estimation of the code length. An example as follows further describes the searching procedure.

Figure 2 shows the recorded vector $\mathbf{N}_l$ when recognizing a $C(3,2,4)$ convolutional code based on correct synchronization positions with a low SNR ($E_s/N_0 = 4\,\text{dB}$). From the figure we can see that not all the gaps between two consecutive nonzero elements in $\mathbf{N}_l$ equal the code length, 3. If we estimate the code length by the gap between the first and the second nonzero elements in $\mathbf{N}_l$, the estimation is not correct. Implementing the searching steps mentioned above, we can firstly obtain the vector $\mathbf{V}_l$ as follows:

$$\mathbf{V}_l = [21,\ 30,\ 33,\ 39,\ 42,\ 45,\ 48]. \tag{5}$$

Then we can calculate the vector $\Delta\mathbf{V}_l$ according to $\mathbf{V}_l$ as follows:

$$\Delta\mathbf{V}_l = [9\ \ 3\ \ 6\ \ 3\ \ 3\ \ 3]. \tag{6}$$

Furthermore, we can calculate the vector $\mathbf{Q}$ by (4). In the vector $\mathbf{Q}$, we have $\mathbf{Q}(3) = 4$, $\mathbf{Q}(6) = \mathbf{Q}(9) = 1$, and other elements in $\mathbf{Q}$ equal to zero. $\mathbf{Q}(3)$ is the maximum element in the vector $\mathbf{Q}$, so, according to Step 4 mentioned above, we can estimate the code length: $\widehat{n} = 3$.

This example shows an implementation of the formalized algorithm proposed in this section to estimate the code length more exactly in a low SNR situation.

4. Recognition of Synchronization Positions

The GJETP-based dual code method proposed in [7] has good performances on convolutional encoder identifications. But unfortunately, the authors did not consider the codeword synchronization problem. In noncooperative context, the synchronization cannot usually be reached before the coding parameters are correctly estimated. The algorithm is discussed based on correct synchronization. In the case that the codeword synchronization position is unknown, we can but randomly choose a position in the received data stream to be the beginning of a codeword. The randomly chosen synchronization positions have low influence on the recognition of code length, but the recognition of key parameter n_a (the minimal length of the rows of matrix $\mathbf{R}_l$ so that $\mathbf{R}_l$ includes dependent columns; see [7] for details) and parity check vectors are not correct.

Figure 3 shows the recorded vector $\mathbf{N}_l$ when recognizing a $C(3,2,4)$ convolutional code based on correct synchronization positions. From the figure we can estimate the key parameter $n_a = 21$, which is accordant to the $C(3,2,4)$ convolutional code. But if we choose an incorrect synchronization position, as shown in Figure 4, the first nonzero element in vector $\mathbf{N}_l$ appears at the position $l = 24$, so the key parameter n_a is estimated to be 24, which is a fault estimation. In order

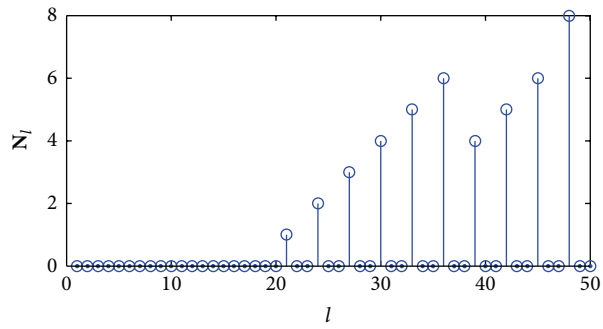

Figure 3: The recorded vector $\mathbf{N}_l$ based on correct synchronization.

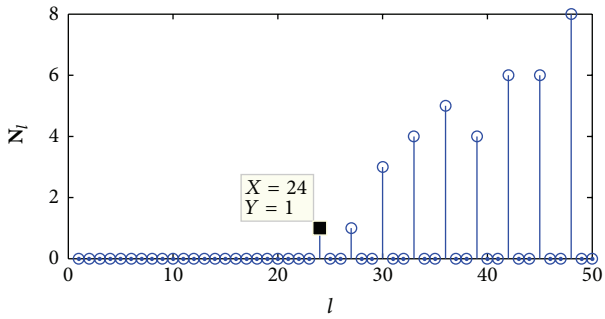

Figure 4: The recorded vector $\mathbf{N}_l$ based on incorrect synchronization.

to obtain correct parity check vectors, we must firstly locate the correct synchronization positions.

We assume that when recognizing the code length we fill the matrix $\mathbf{R}_l$ from the t_0th position from the received bit stream and the estimated code length and key parameter n_a are $\widehat{n}$ and $\widehat{n}_a$, respectively, and then we propose the following steps to find the correct codeword synchronization.

Step 1. Let $\tau = t_0 + 1$, $\overline{n}_a = n_a$, and $\overline{\tau} = t_0$.

Step 2. Let $l = \widehat{n}_a - n$.

Step 3. Fill the matrix $\mathbf{R}_l$ with row length l from the τth bit in the received data stream.

Step 4. Do the GJETP processing for $\mathbf{R}_l$ and calculate $\mathbf{N}_l$, the number of dependent columns.

Step 5. If $\mathbf{N}_l > 0$ and $l < \overline{n}_a$, let $\overline{n}_a = l$, $\overline{\tau} = \tau$, and $l = l - n$, and go back to Step 3; otherwise, execute Step 6.

Step 6. If $\tau < t_0 + n - 1$, let $\tau = \tau + 1$ and go back to Step 2.

Step 7. Let $\overline{n}_a$ and $\overline{\tau}$ be the newly estimations of n_a and synchronization position.

The previous steps corrected the codeword synchronization. For the following procedure of parity check matrix recognition, we use τ to be the synchronization position to fill the matrix $\mathbf{R}_l$ and replace n_a by $\overline{n}_a$ in the algorithms.

5. Correlation Attack Method

5.1. Hard-Decision Situations. If the polynomial-based generator matrices of a k/n convolutional code and the dual code are $\mathbf{G}(D)$ and $\mathbf{H}(D)$, respectively, we have [16]

$$\mathbf{G}(D) \times \mathbf{H}(D) = 0. \tag{7}$$

According to the analysis of [7], $\mathbf{H}(D)$ has the following style:

$$\mathbf{H}(D) = \begin{bmatrix} h_{1,1}(D) & \cdots & h_{1,k}(D) & h_0(D) & & \\ \vdots & \ddots & \vdots & & \ddots & \\ h_{n-k,1}(D) & \cdots & h_{n-k,k}(D) & & & h_0(D) \end{bmatrix}, \tag{8}$$

where $h_{j,s}(D)$ $(1 \le j \le n-k, 1 \le s \le k)$ is a polynomial, coefficients of which are all in GF(2):

$$h_{j,s}(D) = h_{j,s}(0) + h_{j,s}(1)\, D + \cdots + h_{j,s}\left(\mu^{\perp}\right) D^{\mu^{\perp}}. \tag{9}$$

Parameter $\mu^{\perp}$ in (7) is the memory of the dual code [6, 7].

$h_0(D)$ is also a polynomial on GF(2) as follows:

$$h_0(D) = h_0(0) + h_0(1)\, D + \cdots + h_0\left(\mu^{\perp}\right) D^{\mu^{\perp}}. \tag{10}$$

And according to [7], for a realizable convolutional encoder, we have

$$h_0(0) = 1. \tag{11}$$

Based on (8)–(10), we can define the binary form of $\mathbf{H}(D)$ as follows:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{\mu^{\perp}} & \mathbf{H}_{\mu^{\perp}-1} & \cdots & \mathbf{H}_0 & & \\ & \mathbf{H}_{\mu^{\perp}} & \mathbf{H}_{\mu^{\perp}-1} & \cdots & \mathbf{H}_0 & \\ & & \mathbf{H}_{\mu^{\perp}} & \mathbf{H}_{\mu^{\perp}-1} & \cdots & \mathbf{H}_0 \\ & & & \ddots & \ddots & \ddots \end{bmatrix}, \tag{12}$$

where

$$\mathbf{H}_i = \begin{bmatrix} h_{1,1}(i) & \cdots & h_{1,k}(i) & h_0(i) & & \\ \vdots & \ddots & \vdots & & \ddots & \\ h_{n-k,1}(i) & \cdots & h_{n-k,k}(i) & & & h_0(i) \end{bmatrix}. \tag{13}$$

It is shown in (12) that each row of the matrix $\mathbf{H}$ is a shift of the following $(n-k) \times n(\mu^{\perp}+1)$ matrix $\mathbf{H}_0$, which we call the basic check matrix of convolutional codes in this paper:

$$\mathbf{H}_0 = \left[\mathbf{H}_{\mu^{\perp}} \quad \mathbf{H}_{\mu^{\perp}-1} \quad \cdots \quad \mathbf{H}_0\right]. \tag{14}$$

So the recognition problem of a convolutional encoder is equivalent to the recognition of $\mathbf{H}_0$. We call the vectors in $\mathbf{H}_0$ the basic parity check vectors. According to the dual code principles, we can consider $\mathbf{H}_0$ to be a parity check matrix for $(\mu^{\perp}+1)$ consecutive codewords. That is to say, if a row vector $\mathbf{c}$ includes $(\mu^{\perp}+1)$ consecutive codewords, we have

$$\mathbf{c} \times \mathbf{H}_0^T = 0. \tag{15}$$

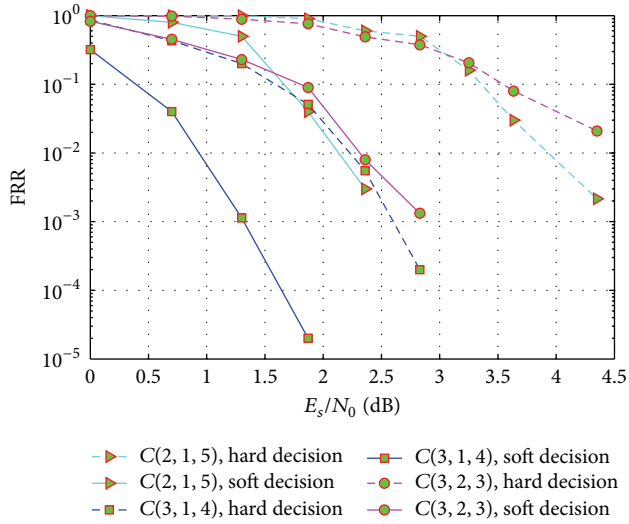

Figure 5: Recognition performances of GJETP method.

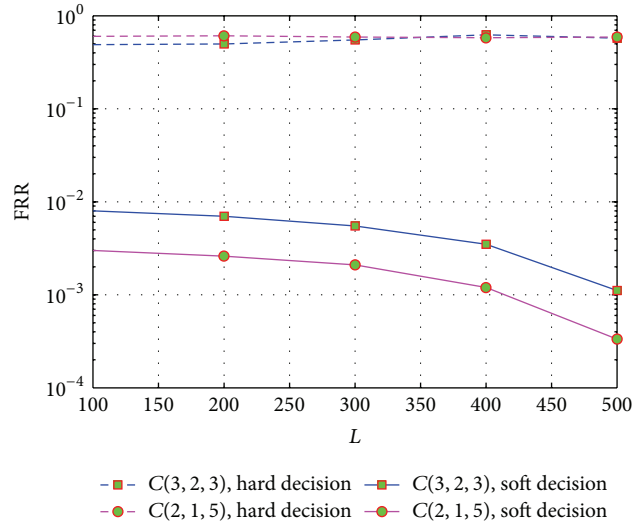

Figure 6: Recognition performances on different size of $\mathbf{R}_l$ for soft and hard decisions.

Furthermore, for a matrix $\mathbf{C}$ consisting of a number of such vectors, we have

$$\mathbf{C} \times \mathbf{H}_0^T = 0. \tag{16}$$

In this paper we say two vectors $\mathbf{c}$ and $\mathbf{h}$ are correlated if they have the same length l and

$$\sum_{i=1}^{l} c_i h_i = 0, \tag{17}$$

where the sum operator is defined in GF(2) and

$$\begin{aligned} \mathbf{c} &= [c_1 \;\; c_2 \;\; \cdots \;\; c_l], \\ \mathbf{h} &= [h_1 \;\; h_2 \;\; \cdots \;\; h_l]. \end{aligned} \tag{18}$$

One solution to the recognition of $\mathbf{H}_0$ is enumerating all possible length values l and listing all vectors with length l to see which vectors have likelihood to be the basic parity check vectors. This can be implemented by enumerating all possible length values l, creating the observed data matrix $\mathbf{R}_l$ with row length l (as shown in Figure 1), and trying to find the vectors which are correlated with most rows of $\mathbf{R}_l$. If we can find such vectors, the corresponding l can be considered to be a possible estimation of ω, which is defined to be the row length of the basic check matrix $\mathbf{H}_0$:

$$\omega = n\left(\mu^{\perp} + 1\right). \tag{19}$$

Furthermore, we can estimate the code length n and dual code memory length $\mu^{\perp}$ by investigating the factors of the estimated ω.

This scheme takes a very long elapsed time. Here we propose some principles to reduce the searching space.

(1) Reduce the candidates of l.

According to (19), the parameter ω is a product of two integers which are all larger than 1. Therefore, we can drop the prime values while searching. So we just need to enumerate the values of l from composite numbers.

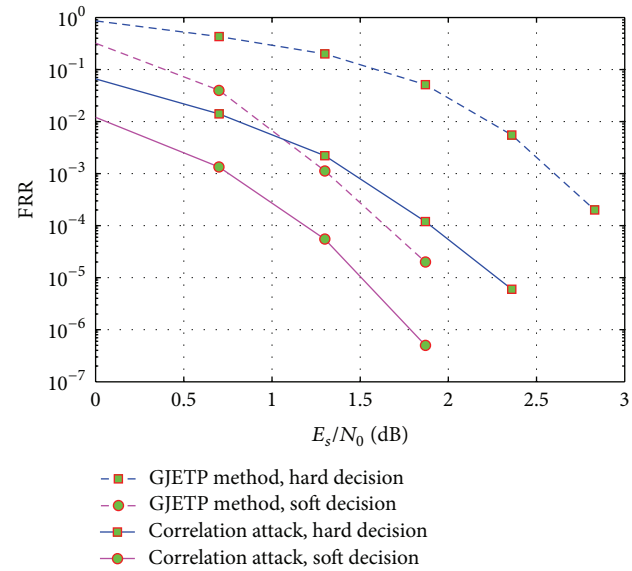

Figure 7: Comparison of GJETP method and correlation attack.

(1) Reduce the candidates of basic parity check vectors.

We can set the first and the last bit of each candidate vector to be 1. Hence, we just need to enumerate the combinations of the middle $(l-2)$ bits for each l-length vector.

In the noisy environment, not all the rows in $\mathbf{R}_l$ are correlated with the basic parity check vectors. $\mathbf{R}_l$ can be written as follows:

$$\mathbf{R}_l = \mathbf{C}_l + \mathbf{E}_l, \tag{20}$$

where $\mathbf{E}_l$ is the error pattern corresponding to the elements in $\mathbf{R}_l$ and the elements in $\mathbf{C}_l$ are the original encoded (noiseless) bits of the received bits in $\mathbf{R}_l$. If $l = n(\mu^{\perp} + 1)$ and no error

exists in the received stream, that is, $\mathbf{E}_l = 0$, according to (16), we have

$$\mathbf{R}_l \times \mathbf{H}_0^T = \mathbf{0}. \tag{21}$$

In most noisy environments, because of the existence of error bits, (21) is not always true even if l equals $n(\mu^{\perp} + 1)$. So we cannot determine whether a vector $\mathbf{h}$ is correlated with row vectors of $\mathbf{R}_l$ by checking the equation $\mathbf{R}_l \times \mathbf{h}^T = \mathbf{0}$. Instead, we need to compute the likelihood of a vector $\mathbf{h}$ to be correlated with $\mathbf{R}_l$ and an appropriate threshold to determine whether the vector $\mathbf{h}$ can be considered to be a basic check vector. We now deduce the threshold below.

For a given data matrix $\mathbf{R}_l$ and vector $\mathbf{h}$, we denote by $\mathbf{q}$ a vector calculated by $\mathbf{R}_l \times \mathbf{h}^T$ as follows:

$$\mathbf{q} = \mathbf{R}_l \times \mathbf{h}^T. \tag{22}$$

Equation (22) is defined in GF(2), so the elements in vector $\mathbf{q}$ are from the set $\{0, 1\}$. And we denote by $wt(\mathbf{q})$ the Hamming weight of the vector $\mathbf{q}$. In noisy environment, the expectation of $wt(\mathbf{q})$ can be calculated as follows:

$$E\left[wt\left(\mathbf{q}\right)\right] = L\left[1 - \Pr\left(\mathbf{c} \times \mathbf{h}^T = 0\right)\right], \tag{23}$$

where L is the number of rows of the observed data matrix $\mathbf{R}_l$ and $\mathbf{c}$ is any possible form of the rows of $\mathbf{R}_l$. $\Pr(\mathbf{c} \times \mathbf{h}^T = 0)$ is the probability of $\mathbf{c} \times \mathbf{h}^T = 0$. If the matrix $\mathbf{R}_l$ is filled with correct parameters, that is, $l = n(\mu^{\perp} + 1)$, the first bit filled into $\mathbf{R}_l$ is the beginning of a codeword, and the vector $\mathbf{h}$ is a valid check vector, $\Pr(\mathbf{c} \times \mathbf{h}^T = 0)$ equals the probability that the vector $\mathbf{c}$ has even error bits. So we can calculate the probability $\Pr(\mathbf{c} \times \mathbf{h}^T = 0)$ as follows:

$$\Pr\left(\mathbf{c} \times \mathbf{h}^T = 0\right) = \sum_{i=0}^{\lfloor w/2 \rfloor} \binom{w}{2i} \tau^{2i} (1-\tau)^{w-2i} = \frac{1 + (1-2\tau)^w}{2}, \tag{24}$$

where w is the Hamming weight of $\mathbf{h}$ and τ is the channel transition probability. According to (23) and (24), we have

$$E_1 = E\left(wt\left(\mathbf{q}\right)\right) = \frac{1 - (1-2\tau)^w}{2} L. \tag{25}$$

According to binomial-distribution theories, we can calculate the variance of $wt(\mathbf{q})$ as follows:

$$D_1 = D\left(wt\left(\mathbf{q}\right)\right) = \frac{1 - (1-2\tau)^{2w}}{4} L. \tag{26}$$

If the data matrix $\mathbf{R}_l$ is based on incorrect parameters or $\mathbf{h}$ is not a valid check vector, the probability $\Pr(\mathbf{c} \times \mathbf{h}^T = 0)$ is

$$\Pr\left(\mathbf{c} \times \mathbf{h}^T = 0\right) = \Pr\left(\mathbf{c} \times \mathbf{h}^T = 1\right) = \frac{1}{2}. \tag{27}$$

Therefore, the expectation and variance of $wt(\mathbf{q})$ are

$$\begin{aligned} E_2 &= E\left(wt\left(\mathbf{q}\right)\right) = \frac{1}{2} L, \\ D_2 &= D\left(wt\left(\mathbf{q}\right)\right) = \frac{1}{4} L. \end{aligned} \tag{28}$$

So we propose the threshold δ based on λ-standard deviation principle as follows:

$$\begin{aligned} \delta &= \frac{\left(E_1 + \lambda\sqrt{D_1}\right) + \left(E_2 - \lambda\sqrt{D_2}\right)}{2} \\ &= \Bigg(\left[\left(\left(1 - (1-2\tau)^w \right)/2 \right) L + \lambda \sqrt{\left(\left(1 - (1-2\tau)^{2w} \right)/4 \right) L} \right] \\ &\qquad + \left((1/2) - \lambda\sqrt{(1/4)\, L} \right) \Bigg) (2)^{-1} \\ &= \frac{L}{2} \left[1 - \frac{(1-2\tau)^w}{2} \right] + \frac{\lambda\sqrt{L}}{4} \left[\sqrt{1 - (1-2\tau)^{2w}} - 1 \right]. \end{aligned} \tag{29}$$

Experimentally, we suggest choosing the parameter λ between 6 and 8 to get a good recognition result.

In (29), we must ensure that

$$\left(E_1 + \lambda\sqrt{D_1}\right) < \left(E_1 - \lambda\sqrt{D_2}\right). \tag{30}$$

So the number of rows of the observed data matrix $\mathbf{R}_l$ should meet the following condition:

$$L > \left[\frac{\lambda\sqrt{1 - (1-2\tau)^{2w}} + 1}{(1-2\tau)^w} \right]^2. \tag{31}$$

The threshold δ and corresponding condition of L are based on the prior information about the noise level, that is, the channel transition probability τ. If such prior information is unknown, the calculation of (29) and (31) cannot be done. In this situation, we propose to set the threshold δ to be $L/10$ and let

$$\left(\frac{1}{2} L - \lambda \sqrt{\frac{1}{4} L} \right) > \delta = \frac{L}{10}. \tag{32}$$

Therefore, L should meet the following condition:

$$L > \left(\frac{5}{4} \lambda \right)^2. \tag{33}$$

The threshold $\delta = L/10$ is just an experimental value. If no parity check vector can be found based on such a threshold, we can increase the value of δ and redo the searching procedure. If $\delta = \mu/\theta$ $(\theta > 2)$, we must choose L such that

$$\left(\frac{1}{2} L - \lambda \sqrt{\frac{1}{4} L} \right) > \delta = \frac{L}{\theta}; \tag{34}$$

that is,

$$L > \left(\frac{\theta}{\theta - 2} \lambda \right)^2. \tag{35}$$

To implement the algorithm automatically by a computer program, we propose the following procedure to recognize the parameter ω defined by (19) and the codeword synchronization position t_0 as follows.

Step 1. Set the searching range of ω; that is, set $\omega_{\min}$ and $\omega_{\max}$.

Step 2. List all the composite numbers between $\omega_{\min}$ and $\omega_{\max}$ to form a set

$$\{\omega_1 \ \ \omega_2 \ \cdots \ \omega_\varepsilon\}, \tag{36}$$

where $\omega_1 < \omega_2 < \cdots < \omega_\varepsilon$, ε is the number of composite numbers between $\omega_{\min}$ and $\omega_{\max}$.

Step 3. Let $i = 1$.

Step 4. Let $l = \omega_i$.

Step 5. Let $t = 1$; fill the data matrix $\mathbf{R}_l$ with the received data. The first bit filled into $\mathbf{R}_l$ is the tth bit in the received stream.

Step 6. Let $j = 1$.

Step 7. Create a vector:

$$\mathbf{h} = [h_1 \ \ h_2 \ \cdots \ h_{l-1} \ \ h_l], \tag{37}$$

where $h_1 = h_l = 1$ and $h_2 h_3 \cdots h_{l-1}$ is the binary form of j with length $l - 2$.

Step 8. Calculate the vector $\mathbf{q}$ by (22) and denote by $wt(\mathbf{q})$ the Hamming weight of $\mathbf{q}$. If $wt(\mathbf{q}) < \delta$, stop the searching and output t and l to be the estimation of t_0 and ω; that is, let $\widehat{t}_0 = t$ and $\widehat{\omega} = l$. Besides, record $\mathbf{h}$ as $\mathbf{h}_x$.

Step 9. If $j < 2^{l-2} - 1$, let $j = j + 1$ and go back to Step 7. Otherwise, execute Step 10.

Step 10. If $t < n_{\max}$, let $t = t + 1$ and go back to Step 6. Otherwise, execute Step 9. $n_{\max}$ is the maximum of possible codeword length. $n_{\max}$ can be set to the maximum factor of l (except l itself).

Step 11. If $i < \varepsilon$, go back to Step 4. Otherwise, execute Step 12.

Step 12. End the searching.

If we can stop the searching from Step 8, we can recognize the codeword length n based on $\widehat{t}_0$, $\widehat{\omega}$, and $\mathbf{h}_x$. According to (19), the code length n is a factor of ω. So we propose to estimate n by the following steps.

Step 1. List all the factors of ω (expect 1 and ω itself) to form a set

$$\{n_1 \ \ n_2 \ \cdots \ n_\sigma\}, \tag{38}$$

where $n_1 < n_2 < \cdots < n_\sigma$, σ is the number of the factors of ω (expect 1 and ω itself); let $l = \omega$.

Step 2. Let $i = 1$.

Step 3. Let $n = n_i$.

Step 4. Fill the data matrix $\mathbf{R}_l$ as follows:

$$\mathbf{R}_l = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_L \end{pmatrix}, \tag{39}$$

where

$$\begin{aligned}
\mathbf{c}_1 &= [c_1 \ \cdots \ c_n \mid c_{n+1} \ \cdots \ c_{2n} \mid \cdots \mid c_{\mu^\perp n+1} \ \cdots \ c_{(\mu^\perp+1)n}], \\
\mathbf{c}_2 &= [c_{n+1} \ \cdots \ c_{2n} \mid c_{2n+1} \ \cdots \ c_{3n} \mid \cdots \mid c_{(\mu^\perp+1)n+1} \ \cdots \ c_{(\mu^\perp+2)n}], \\
\mathbf{c}_3 &= [c_{2n+1} \ \cdots \ c_{3n} \mid c_{3n+1} \ \cdots \ c_{4n} \mid \cdots \mid c_{(\mu^\perp+2)n+1} \ \cdots \ c_{(\mu^\perp+3)n}], \\
&\vdots
\end{aligned} \tag{40}$$

where c_i $(i \geq 1)$ is the $(\widehat{t}_0 + i - 1)$th bit in the received data stream and $\mu^\perp = \widehat{\omega}/n - 1$.

Step 5. Calculate the vector $\mathbf{q}$ by $\mathbf{q} = \mathbf{R}_l \times \mathbf{h}_x{}^T$ and get $wt(\mathbf{q})$, the Hamming weight of $\mathbf{q}$.

Step 6. If $wt(\mathbf{q}) < \delta$, stop the searching and output the current n to be the estimated codeword length: $\widehat{n} = n$. Otherwise, execute Step 7.

Step 7. If $i < \sigma$, let $i = i + 1$ and go back to Step 3. Otherwise, execute Step 8.

Step 8. End the searching.

After the estimation of n, we can simply get the estimation of $\mu^\perp$ by

$$\widehat{\mu}^\perp = \frac{\widehat{\omega}}{\widehat{n}} - 1. \tag{41}$$

Finally, we search all the basic check vectors to recover the basic check matrix $\mathbf{H}_0$. According to (13) and (14), we can estimate the encoder parameter k and the $n - k$ parity check vectors by the following steps.

Step 1. Let $k = 1$.

Step 2. Let $s = 1$.

Step 3. Create a vector $\mathbf{x}_s$ as follows:

$$\mathbf{x}_s = \left[c_1 \ \cdots \ c_k \ \ c_{k+s} \ \ c_{\hat{n}+1} \ \cdots \ c_{\hat{n}+k} \ \ c_{\hat{n}+k+s} \ \ c_{2\hat{n}+1} \ \cdots\right], \tag{42}$$

where the definition of c_i $(i \geq 1)$ is the same as (40).

Step 4. Let $l = (k+1)(\mu^{\perp}+1)$ and fill the elements in $\mathbf{x}_s$ into the matrix $\mathbf{R}_l$ with row length l.

Step 5. Enumerate all vectors $\mathbf{h}$ with length l, where the last element of $\mathbf{h}$ is 1, and calculate $wt(\mathbf{q})$, where $\mathbf{q} = \mathbf{R}_l \times \mathbf{h}_x{}^T$. If any vector $\mathbf{h}$ can make $wt(\mathbf{q}) < \delta$, record $\widehat{\mathbf{h}}_s = \mathbf{h}$ and execute Step 6. Otherwise, execute Step 8.

Step 6. If $s < \hat{n} - k$, let $s = s + 1$ and go back to Step 3. Otherwise, execute Step 7.

Step 7. Stop the searching, and output $\hat{k} = k$ and the recorded $\hat{n} - \hat{k}$ vectors $\widehat{\mathbf{h}}_1, \widehat{\mathbf{h}}_2, \ldots, \widehat{\mathbf{h}}_{\hat{n}-\hat{k}}$.

Step 8. If $k < \hat{n} - 1$, let $k = k + 1$ and go back to Step 2. Otherwise, execute Step 9.

Step 9. End the searching.

If such recognition procedure can successfully output $\hat{n} - \hat{k}$ vectors $\widehat{\mathbf{h}}_1, \widehat{\mathbf{h}}_2, \ldots, \widehat{\mathbf{h}}_{\hat{n}-\hat{k}}$, we can finally recover the basic parity check matrix $\mathbf{H}_0$. We write the vectors $\widehat{\mathbf{h}}_1, \widehat{\mathbf{h}}_2, \ldots, \widehat{\mathbf{h}}_{\hat{n}-\hat{k}}$ as follows:

$$\begin{aligned}
\widehat{\mathbf{h}}_1 &= \left[h_{1,1} \cdots h_{1,\hat{k}} h_{1,\hat{k}+1} \ \mid \ h_{1,(\hat{k}+1)+1} \cdots h_{1,(\hat{k}+1)+\hat{k}} h_{1,2(\hat{k}+1)} \ \mid \ \cdots \ \mid \ h_{1,\hat{\mu}^{\perp}(\hat{k}+1)+1} \cdots h_{1,\hat{\mu}^{\perp}(\hat{k}+1)+\hat{k}} h_{1,(\hat{\mu}^{\perp}+1)(\hat{k}+1)}\right], \\
\widehat{\mathbf{h}}_2 &= \left[h_{2,1} \cdots h_{2,\hat{k}} h_{2,\hat{k}+1} \ \mid \ h_{2,(\hat{k}+1)+1} \cdots h_{2,(\hat{k}+1)+k} h_{2,2(\hat{k}+1)} \ \mid \ \cdots \ \mid \ h_{2,\mu^{\perp}(\hat{k}+1)+1} \cdots h_{2,\hat{\mu}^{\perp}(\hat{k}+1)+\hat{k}} h_{2,(\hat{\mu}^{\perp}+1)(\hat{k}+1)}\right], \\
&\vdots \\
\widehat{\mathbf{h}}_{\hat{n}-\hat{k}} &= \left[h_{\hat{n}-\hat{k},1} \cdots h_{1,\hat{k}} h_{1,\hat{k}+1} \ \mid \ h_{\hat{n}-\hat{k},(\hat{k}+1)+1} \cdots h_{\hat{n}-\hat{k},(\hat{k}+1)+\hat{k}} h_{\hat{n}-\hat{k},2(\hat{k}+1)} \ \mid \ \cdots \ \mid \ h_{\hat{n}-\hat{k},\hat{\mu}^{\perp}(\hat{k}+1)+1} \cdots h_{\hat{n}-\hat{k},\hat{\mu}^{\perp}(\hat{k}+1)+\hat{k}} h_{\hat{n}-\hat{k},(\hat{\mu}^{\perp}+1)(\hat{k}+1)}\right],
\end{aligned} \tag{43}$$

and then the parity check matrix $\mathbf{H}_0$ can be estimated as follows:

$$\widehat{\mathbf{H}}^0 = \begin{bmatrix}
\hat{h}_{1,1} & \cdots & \hat{h}_{1,k} & \hat{h}_{1,k+1} & & & \hat{h}_{1,k+2} & \cdots & \hat{h}_{1,2k+1} & \hat{h}_{1,2k+2} & & & \hat{h}_{1,k+2} & \cdots \\
\vdots & \ddots & \vdots & & \ddots & & \vdots & \ddots & \vdots & & \ddots & & \vdots & \cdots \\
\hat{h}_{n-k,1} & \cdots & \hat{h}_{n-k,k} & & & \hat{h}_{n-k,k+1} & \hat{h}_{n-k,k+2} & \cdots & \hat{h}_{n-k,2k+1} & & & \hat{h}_{n-k,2k+2} & \hat{h}_{n-k,k+2} & \cdots
\end{bmatrix}. \tag{44}$$

5.2. Soft-Decision Situations. In hard-decision situations, we calculate $wt(\mathbf{q})$ by evaluating how many vectors in $\mathbf{R}_l$ are correlated with a candidate parity check vector $\mathbf{h}$. In soft decisions, we can utilize the soft output from the receiver to evaluate the reliability of each decision bit. Therefore, we can calculate the probability of each row in $\mathbf{R}_l$ to be correlated with $\mathbf{h}$. For example, if a communication system uses BPSK demodulation in an additive-white-Gaussian-noise (AWGN) channel, according to the analysis in [14, 15], we can calculate the probability of $\mathbf{c} \times \mathbf{h}^T = 1$ and $\mathbf{c} \times \mathbf{h}^T = 0$ for given vectors $\mathbf{c}$ and $\mathbf{h}$ as follows:

$$\begin{aligned}
P_r\left(\mathbf{c} \times \mathbf{h}^T = 0\right) &= \frac{1}{2} + \frac{1}{2}\prod_{j=1}^{w} \tanh\left(\frac{c_{u_j}}{\sigma^2}\right) \\
P_r\left(\mathbf{c} \times \mathbf{h}^T = 1\right) &= \frac{1}{2} - \frac{1}{2}\prod_{j=1}^{w} \tanh\left(\frac{c_{u_j}}{\sigma^2}\right),
\end{aligned} \tag{45}$$

where σ is the noise variance, w is the Hamming weight of $\mathbf{h}$, u_j is the jth nonzero position in $\mathbf{h}$, and c_{u_j} is the u_jth element in vector $\mathbf{c}$. Based on this, $wt(\mathbf{q})$ is calculated as follows:

$$wt\left(\mathbf{q}\right) = \sum_{i=1}^{L} P_r\left(\mathbf{c}_i \times \mathbf{h}^T = 1\right), \tag{46}$$

where $\mathbf{c}_i$ denotes the ith row in $\mathbf{R}_l$.

6. Simulation Results

In this section we show the simulation results of the blind recognition of the convolutional coding parameters by utilizing the method introduced in this paper. The simulations include three parts corresponding to our proposed recognition algorithm on different noise level and different observed matrix size L. In the simulations we assume that the signal

is transmitted on a binary symmetry channel (BSC) which is corrupted by AWGN.

For the GJETP-based recognition method, we show the false recognition ratios (FRR) in Figure 5 for several convolutional codes in different channel conditions when the observed window size $L = 200$. We compare the proposed soft-decision based method with the hard-decision based algorithm proposed in [7]. We can see from the simulation results that when the soft information is introduced into the recognition algorithm, the recognition performances can be improved obviously. And as the SNR rises, the FRR curves descend more rapidly on soft-decision algorithm.

Figure 6 shows the recognition performance of two convolutional codes for different observed matrix size, when the SNR $E_s/N_0 = 2.36$ dB. It shows that the soft information can help to make the FRR descend more rapidly when L is rising. That is to say, in soft-decision situations, we can improve the recognition performance by increasing the number of rows in the data matrix of $\mathbf{R}_l$, while the hard-decision based algorithm cannot.

In Figure 7, we compare the performances of convolutional encoder recognition by GJETP method and correlation attack while recognizing the coding parameters of $C(2, 1, 3)$. GJETP method is based on Gauss elimination on $\mathbf{R}_l$, so the influences of error bits are easily diffused during Gauss eliminating. Therefore, the fault-tolerance of GJETP method is limited. Correlation attack method can avoid this problem, so the recognition performance can be improved in low SNR situations.

7. Conclusions

This paper proposes the methods of utilizing soft information to improve the recognition performance of convolutional encoder parameters. And we propose a formalized estimation of the parameter n and synchronization positions. When introducing the soft information the recognition performance can be obviously improved and the simulations show the efficiency of the proposed methods. Besides, we propose a new algorithm to recover the basic parity check matrix by correlation attack. Although this method takes longer elapsed time, it can push the SNR limits of the GJETP method, and some principles are proposed to reduce the searching space. If the channel quality is well, the GJETP method has advantages on short computational delay. For a worse channel quality such that the GJETP does not work, correlation attack method has a significant advantage on its higher fault-tolerance.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] B. Rice, "Determining the parameters of a rate 1/n convolutional encoder," in *Proceedings of the International Conference on Finite Fields and Applications*, Glasgow, UK, 1995.

[2] E. Filiol, "Recognition of convolutional encoders over GF(p)," in *Proceedings of the 6th IMA Conference on Cryptogrophy and Coding*, Cirencester, UK, December 1997.

[3] M. Marazin, R. Gautier, and G. Burel, "Blind recovery of the second convolutional encoder of a turbo-code when its systematic outputs are punctured," *Military Technical Academy Review XIX*, vol. 2, pp. 213–232, 2009.

[4] F. Wang, Z. Huang, and Y. Zhou, "A method for blind recognition of convolution code based on euclidean algorithm," in *Proceedings of the IEEE International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '07)*, pp. 1414–1417, Shanghai, China, September 2007.

[5] J. Dingel and J. Hagenauer, "Parameter estimation of a convolutional encoder from noisy observations," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '07)*, pp. 1776–1780, Nice, France, June 2007.

[6] M. Marazin, R. Gautier, and G. Burel, "Dual code method for blind identification of convolutional encoder for cognitive radio receiver design," in *Proceedings of the IEEE Globecom Workshops (GLOBECOM '09)*, Honolulu, Hawaii, USA, November-December 2009.

[7] M. Marazin, R. Gautier, and G. Burel, "Blind recovery of k/n rate convolutional encoders in a noisy environment," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, article 168, pp. 1–9, 2011.

[8] S. Lin and D. J. Costello, "Reliability-based soft-decision decoding algorithms for linear block codes," in *Error Control Coding*, J. M. Horton and D. W. Riccardi, Eds., pp. 395–452, Pearson Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2004.

[9] L. Ni, F. Yao, and L. Zhang, "A rotated quasi-othogonal space-time block code for asynchronous cooperative discovery," *Entropy*, vol. 14, pp. 654–664, 2012.

[10] X. Liu and X. Geng, "A convolutional code-based sequence analysis model and its application," *International Journal of Molecular Sciences*, vol. 14, pp. 8393–8405, 2013.

[11] J. Hagenauer and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 429–445, 1996.

[12] Z. Huang, M. Chen, and C. Diao, "Low compexity weighted reliability-based iterative decoding of LDPC codes," *IEICE Transactions on Communications*, vol. 95, pp. 3572–3575, 2012.

[13] J. Jiang and K. R. Narayanan, "Iterative soft-input soft-output decoding of reed-solomon codes by adapting the parity-check matrix," *IEEE Transactions on Information Theory*, vol. 52, no. 8, pp. 3746–3756, 2006.

[14] J. Zhou, Z. Huang, C. Liu, S. Su, and Y. Zhang, "Information-dispersion-entropy-based blind recognition of binary BCH codes in soft decision situations," *Entropy*, vol. 15, pp. 1705–1725, 2013.

[15] J. Zhou, Z. Huang, S. Su, and S. Yang, "Blind recognition of binary cyclic codes," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, article 218, pp. 1–17, 2013.

[16] G. D. Forney Jr., "Structural analysis of convolutional codes via dual codes," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 512–518, 1973.