S & M 2493

Fast Adaptive Low-density Parity Check Coding Based on Confidence Criteria

Xiue Gao,[1] Tianshu Zhang,[2] Shifeng Chen,[2]
Bo Chen,[1*] Chun-Chi Chen,[3] and Hsien-Wei Tseng[4**]

[1]College of Mechanical and Electronical Engineering, Lingnan Normal University, Zhanjiang 524048, China
[2]College of Information Engineering, Dalian University, Dalian 116622, China
[3]School of Life Sciences, Longyan University, Longyan, Fujian 364012, China
[4]School of Mathematics and Information Engineering, Longyan University, Longyan, Fujian 364012, China

(Received July 20, 2020; accepted January 26, 2021)

Keywords: LDPC code, confidence criteria, shortening, puncturing, IEEE802.16e

Low-density parity check (LDPC) codes have been gradually used in Wireless Sensor Networks (WSNs) owing to their strong error-correcting ability, low coding complexity, and high decoding efficiency. The shortcomings of the IEEE802.16e standard for LDPC coding algorithms are the fixed code length and code rates. These requirements are difficult to meet because LDPC code shortening and puncturing methods are currently underdeveloped. In this paper, we present an efficient adaptive LDPC coding method based on confidence criteria. First, a log likelihood ratio-based belief propagation (LLR-BP) algorithm iteratively generates a confidence value that is used to determine the priority ranking of shortening and puncturing positions across an information sequence. At transmission, the shortening position of the information sequence is set to 0 whenever the code length is less than one LDPC code word. Coding can be completed rapidly using the information sequence along with the original information bits. Faster coding can also be achieved by using the original information when the current code rate is lower than the rate demanded. The coding sequence can be sent after deleting the puncturing position corresponding to the verified bit. Knowing that this procedure has been applied enables BP decoding to be performed at the receiving end. Simulation results show that the method can adaptively adjust the code length and code rate and effectively improve the performance of LDPC coding algorithms. Thus, the anti-jamming ability of WSNs is enhanced.

1. Introduction

In modern wireless communication systems, the transmission channel varies over time, as does the amount of transmitted information. This has led to the use of adaptive low-density parity check (LDPC) coding algorithms to dynamically adjust the code length and code rate.

When the code length is less than one LDPC code word, a shortening technique is required. One way of going about this is to insert 0 or 1 in some parts of the original information to bring it to the required coding length. The inserted bits are deleted after encoding and the original

*Corresponding author: e-mail: chenbo20040607@126.com
**Corresponding author: e-mail: hsienwei.tseng@gmail.com
https://doi.org/10.18494/SAM.2021.3043

information and calculated parity coding bits are sent together. Upon receipt, the channel information for the deleted locations is set to infinity and the transmission can be decoded. A reverse-order shortening method has been adopted as part of IEEE802.11ac and other protocols.(1) The codes are shortened to the desired number of bits, working from the last bit to the first. However, this method cannot shorten codes to the optimal number of bits. Liu *et al.* have therefore proposed a shortening algorithm for irregular quasi-cyclic (QC) LDPC codes according to the distribution characteristics of the parity check matrix for the LDPC codes.(2) On the basis of this, the sum of the outer information in the variable nodes is defined as a priority selection criterion for shortening. Chen *et al.* have proposed a shortening method based on the destructive principle.(3) This quantifies the degree of damage to a ring and selects the shortened nodes with the least total destructive power. Xu *et al.* have suggested another shortening method that is based on smallest-row-variance priority (SRVP),(4) which improves the performance of the shortening process.

A puncturing technique is required when the current code rate is lower than the rate demanded. This allows the deletion of some bits, with the remaining bits sent in their original order after encoding. At the receiving end, the channel information regarding the position of each deleted bit is initialized to zero and the transmission is decoded. Ha *et al.* and Park *et al.* have used an optimization search algorithm to determine the optimal nodes for deletion.(5,6) The number of nodes that can be deleted, however, is limited by the structure of the encoding parity check matrix, limiting the dynamic range of the possible rate of change. Su and Shu have studied the role of multiple survivable verification nodes and have proposed a method that can maximize the number of surviving check nodes.(7) Meanwhile, Wu has developed a puncturing algorithm that is based on non-greedy ranking criteria for QC-LDPC codes with a value function that can search for high-value puncturing locations.(8) Xiao *et al.* have proposed a new hierarchical selective deletion algorithm to tackle the problem of the limited maximum punctured code rate.(9)

One of the shortcomings of existing methods is that they ignore the internal structure of LDPC codes and potential channel influence. Vellambi and Fekri have therefore designed a sampling procedure that uses a drilling program to assess the importance of these features.(10) It was found that disregarding the internal structure can impact upon the decoding performance of LDPC codes. Taking into account the structural characteristics of a network's LDPC code, Wang *et al.* have designed a network-based puncturing algorithm that is compatible with the LDPC code rate that can minimize the probability of recovery errors.(11) Zang and Chen have also proposed a puncturing algorithm based on rate-compatible LDPC codes,(12) where a greedy search algorithm is used to maximize the K-step recoverable sections, thus obtaining the maximum number of potential K-step recoverable nodes with the smallest K values. In a different approach, Fulvio has formally analyzed a previously defined class of punctured LDPC codes through their parity check matrices to try and locate an LDPC code rate that is compatible with the design of puncturing patterns.(13) Deka has adopted confidence propagation with edge scheduling,(14) which significantly improves the decoding convergence of punctured codes, and Li *et al.* have proposed a simple and effective LDPC code puncturing strategy based on a diagonal structure.(15) Zhou *et al.* have developed a non-greedy puncturing method to construct binary rate-compatible LDPC codes,(16) and Li and Zheng have introduced an effective puncturing method for irregular LDPC codes,(17) which is based on a serial scheduling program.

For measurement purposes, confidence reflects the reliability of a node and its value represents the amount of information it can receive from external nodes. Guo and Wang have therefore converted data items and confidence measures into priorities,(18) which take the form of weighted sums. This improves the credibility of priority assessments. He *et al*. have proposed a new confidence updating function that can suppress the excessive decay of confidence and improve image guidance and repair accuracy.(19) Nan and Xi have developed a golden ratio for the confidence and weight coefficients of a data item,(20) using conic equations to describe the confidence updating formula. This approach also helped to improve restoration effects. Meanwhile, Zhao and Zhang have proposed a residual confidence distribution,(21) and Fan *et al.* have developed a crossover algorithm for hierarchical confidence propagation.(22)

To some extent, the above-mentioned methods do solve the problem of the code length and code rate for LDPC codes. However, in real wireless communications, it is often necessary to dynamically adjust the code length and the code rate to ensure that LDPC codes are properly adapted to the channel conditions. A particular issue is that, once the base parity check matrix of the IEEE802.16e standard has been determined, the corresponding code rate is specified and the code length can only be one of 19 fixed code lengths. This makes it difficult to realize adaptive LDPC coding. In view of this, we propose in this paper a fast adaptive LDPC coding algorithm based on confidence criteria. The confidence criteria are introduced into the shortening and puncturing techniques and combined with a fast coding algorithm to meet the code length and bit rate requirements of the adaptive dynamic LDPC.

2. Materials and Methods

2.1 LDPC parity check matrix construction

The IEEE802.16e standard for the LDPC code adopts a QC structure. However, its code structure is different from the general structure of the QC-LDPC code because the right-hand side of its check matrix has a quasi-double diagonal structure. The LDPC code of this structure is only used for checking, so the matrix is effectively an extended check matrix.

The IEEE802.16e standard presents a six-base check matrix for the LDPC code that corresponds to the six code rates, i.e., 1/2, 2/3 (class A), 2/3 (class B), 3/4 (class A), 3/4 (class B), and 5/6. The base check matrix can be defined as $\boldsymbol{H_b}$ ($m_b \times n_b$), where n_b is 24 but m_b has a different value. $\boldsymbol{H_b}$ has a special structure and its elements correspond to the sub-matrices of the check matrix $\boldsymbol{H}$, which can be obtained by extending the elements in $\boldsymbol{H_b}$. Note also that $\boldsymbol{H_b} = [\boldsymbol{H_{b1}}\ \boldsymbol{H_{b2}}]$, where the dimensions of $\boldsymbol{H_{b1}}$ are $m_b \times k_b$ ($k_b = n_b - m_b$) and where, once the position of $\boldsymbol{H_b}$ is -1, the sub-matrix $\boldsymbol{H}$ at that position is a zero matrix. In cases where the position of $\boldsymbol{H_b}$ is a non-negative integer, the sub-matrix of $\boldsymbol{H}$ at that position is obtained by using an identity matrix that rotates a non-negative integer to the right. In the first column of $\boldsymbol{H_{b2}}$, $h(1)$, $h(r)$, and $h(m_b)$ are 0 or a non-negative integer, where $h(1) = h(m_b)$, $2 \le r \le m_b - 1$. Thus, the value of $\boldsymbol{H_{b2}}$ for the IEEE802.16e standard for different code rates has been identified.

By selecting z as a parameter, $\boldsymbol{H_b}$ can be extended to $m_b \cdot z \times n_b \cdot z$ for the parity check matrix $\boldsymbol{H}$. This selection of the extension parameter z is not arbitrary. According to the IEEE802.16e standard, z can be one of a series of 19 values, with a minimum of 24 and a maximum of 96. It should be noted that $m = m_b \cdot z$, $n = n_b \cdot z$, so the check matrix $\boldsymbol{H}$ can also be expressed as $m \times n$. The specific expansion method is as follows:

$$\boldsymbol{H_{b2}} = \begin{bmatrix} h(1) & 0 & & & & & & & \\ -1 & 0 & 0 & & & & & & \\ \vdots & & 0 & 0 & & -1 & & & \\ -1 & & & 0 & \ddots & & & & \\ h(r) & & & & \ddots & \ddots & & & \\ -1 & & & & & \ddots & 0 & & \\ \vdots & & & & & & 0 & 0 & \\ -1 & & & & & & & 0 & 0 \\ h(m_b) & & & & & & & & 0 \end{bmatrix}. \tag{1}$$

The elements of matrix $\boldsymbol{H_b}$ in position (i, j) are $q(i, j)$. First of all, each element of $q(i, j)$ can be updated if the rate of the LDPC code is one of 1/2, 2/3 (class A), 2/3 (class B), 3/4 (class A), and 3/4 (class B), and 5/6. The update is realized as follows:

$$q(i,j) = \begin{cases} q(i,j), & q(i,j) \le 0, \\ \left\lfloor \dfrac{q(i,j) \cdot z}{96} \right\rfloor, & q(i,j) > 0. \end{cases} \tag{2}$$

When the rate of the LDPC code is 2/3 (class A), $q(i, j)$ can be updated as follows:

$$q(i,j) = \begin{cases} q(i,j), & q(i,j) \le 0, \\ mod\big(q(i,j), 96\big), & q(i,j) > 0. \end{cases} \tag{3}$$

When $q(i, j)$ is −1, it can be replaced by a zero square matrix with the size of $z \times z$ of $\boldsymbol{H}$. When $q(i, j)$ is a non-negative integer, it can be replaced by a square matrix with the size of $z \times z$ of $\boldsymbol{H}$, which can be obtained by rotating the unit square matrix $q(i, j)$ to the right. The base check matrix $\boldsymbol{H_b}$ for a rate equal to 1/2 is given below:

$$\boldsymbol{H_{b,1/2}} = \begin{bmatrix} -1 & 94 & 73 & -1 & -1 & -1 & -1 & -1 & 55 & 83 & -1 & -1 & 7 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 27 & -1 & -1 & -1 & 22 & 79 & 9 & -1 & -1 & -1 & 12 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 24 & 22 & 81 & -1 & 33 & -1 & -1 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 61 & -1 & 47 & -1 & -1 & -1 & -1 & -1 & 65 & 25 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 39 & -1 & -1 & -1 & 84 & -1 & -1 & 41 & 72 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 46 & 40 & -1 & 82 & -1 & -1 & -1 & 79 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 95 & 53 & -1 & -1 & -1 & -1 & -1 & 14 & 18 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ -1 & 11 & 73 & -1 & -1 & -1 & 2 & -1 & -1 & 47 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ 12 & -1 & -1 & -1 & 83 & 24 & -1 & 43 & -1 & -1 & -1 & 51 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & 94 & -1 & 59 & -1 & -1 & 70 & 72 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 \\ -1 & -1 & 7 & 65 & -1 & -1 & -1 & -1 & 39 & 49 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\ 43 & -1 & -1 & -1 & -1 & 66 & -1 & 41 & -1 & -1 & -1 & 26 & 7 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

However, the base check matrix $\boldsymbol{H_b}$ for a rate equal to 2/3 is as given below:

$$\boldsymbol{H_{b,2/3}} = \begin{bmatrix} 3 & 0 & -1 & -1 & 2 & 0 & -1 & 3 & 7 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 & 36 & -1 & -1 & 34 & 10 & -1 & -1 & 18 & 2 & -1 & 3 & 0 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 12 & 2 & -1 & 15 & -1 & 40 & -1 & 3 & -1 & 15 & -1 & 2 & 13 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 \\ -1 & -1 & 19 & 24 & -1 & 3 & 0 & -1 & 6 & -1 & 17 & -1 & -1 & -1 & 8 & 39 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 \\ 20 & -1 & 6 & -1 & -1 & 10 & 29 & -1 & -1 & 28 & -1 & 14 & -1 & 38 & -1 & -1 & 0 & -1 & -1 & -1 & 0 & 0 & -1 & -1 \\ -1 & -1 & 10 & -1 & 28 & 20 & -1 & -1 & 8 & -1 & 36 & -1 & 9 & -1 & 21 & 45 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 \\ 35 & 25 & -1 & 37 & -1 & 21 & -1 & -1 & 5 & -1 & -1 & 0 & -1 & 4 & 20 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 \\ -1 & 6 & 6 & -1 & -1 & -1 & 4 & -1 & 14 & 30 & -1 & 3 & 36 & -1 & 14 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

2.2 Fast encoding algorithm for LDPC codes

We assume that the information code is $\boldsymbol{s} = [\boldsymbol{s_1^T} \ \ \boldsymbol{s_2^T} \ \cdots \ \boldsymbol{s_{k_b}^T}]$ and the check vector is $\boldsymbol{p} = [\boldsymbol{p_1^T} \ \ \boldsymbol{p_2^T} \ \cdots \ \boldsymbol{p_{m_b}^T}]$, where $\boldsymbol{s_i^T}$, $\boldsymbol{p_i^T}$ are $1 \times z$ vectors. If we also assume that the output of the encoder is the vector $\boldsymbol{c} = [\boldsymbol{s}\ \boldsymbol{p}]$, from $\boldsymbol{H_2} \times \boldsymbol{p^T} = \boldsymbol{H_1} \times \boldsymbol{s^T}$, we can obtain Eq. (4), where $\boldsymbol{H_1}$ and $\boldsymbol{H_2}$ are expansions of the matrices $\boldsymbol{H_{b1}}$ and $\boldsymbol{H_{b2}}$, respectively.

$$\begin{bmatrix} Z_{h(1)} & Z_0 & & & & & & & \\ Z_{-1} & Z_0 & Z_0 & & & & & & \\ \vdots & & Z_0 & Z_0 & & & Z_{-1} & & \\ Z_{-1} & & & Z_0 & Z_0 & & & & \\ Z_{h(r)} & & & & Z_0 & Z_0 & & & \\ Z_{-1} & & & & & Z_0 & Z_0 & & \\ \vdots & & & Z_{-1} & & & Z_0 & Z_0 & \\ Z_{-1} & & & & & & & Z_0 & Z_0 \\ Z_{h(m_b)} & & & & & & & & Z_0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{m_b} \end{bmatrix} = \begin{bmatrix} Z_{H_{b1(1,1)}} & Z_{H_{b1(1,2)}} & \cdots & Z_{H_{b1(1,k_b)}} \\ Z_{H_{b1(2,1)}} & Z_{H_{b1(1,2)}} & \cdots & Z_{H_{b1(2,k_b)}} \\ \vdots & \vdots & \cdots & \vdots \\ Z_{H_{b1(m_b,1)}} & Z_{H_{b1(1,2)}} & \cdots & Z_{H_{b1(m_b,k_b)}} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{k_b} \end{bmatrix} \tag{4}$$

The equations for m_b can be obtained if Eq. (4) is expanded:

$$\begin{cases} Z_{H_b(1)} \cdot p_1 + p_2 = \sum_{j=1}^{m} Z_{H_b(1,j)} \cdot s_j & (1) \\ p_2 + p_3 = \sum_{j=1}^{m} Z_{H_b(2,j)} \cdot s_j & (2) \\ \vdots & \vdots \\ Z_{H_b(n)} \cdot p_1 + p_n = \sum_{j=1}^{m} Z_{H_b(m,j)} \cdot s_j & (m) \end{cases} \tag{5}$$

By adding the equations for m_b together, the following can be obtained:

$$p_1 = \left(Z_{h(1)} + Z_{h(r)} + Z_{h(m_b)} \right)^{-1} \sum_{i=1}^{m_b} \sum_{j=1}^{k_b} Z_{H_{b1}(i,j)} \cdot s_j \tag{6}$$

It is assumed that

$$\lambda_i = \sum\nolimits_{j=1}^{k_b} Z_{H_{b1}(i,j)} \cdot s_j \, , \tag{7}$$

$$p_2 = \sum\nolimits_{j=1}^{k_b} Z_{H_{b1}(1,j)} \cdot s_j + Z_{h(1)} \cdot p_1 \, , \tag{8}$$

Equation (8) can be substituted into the second equation of Eq. (5), and so on. Eventually, the following equations are obtained:

$$p_i = p_{i-1} + \sum\nolimits_{j=1}^{k_b} Z_{H_{b1}(i-1,j)} \cdot s_j \, , \; i = 3,4, \ldots, r, r+2, r+3, m_b \tag{9}$$

$$p_{r+1} = p_r + \sum\nolimits_{j=1}^{k_b} Z_{H_{b1}(r,j)} \cdot s_j + Z_{h(r)} \cdot p_1 \tag{10}$$

3. Fast Adaptive LDPC Coding Technique Based on Confidence Criteria

3.1 Confidence criteria for ranking

We assume here that all of the nodes have the same possible initial logarithmic information when the decoding of the likelihood ratio-based belief propagation (LLR-BP) algorithm begins. After l iterations, the variable node of the hard-decision message $L^{(l)}(q_i)$ can be represented as follows:

$$L^{(l)}(q_i) = L(P_i) + \sum\nolimits_{j \in C_i} L^{(l)}(r_{ji}) \tag{11}$$

The results for $L^{(l)}(q_i)$ can differ because the decoded external information is not unique for each different node. The value of $L^{(l)}(q_i)$ can reflect the amount of information a node receives from the external nodes. Nodes that can receive more external information are more conducive to decoding. $|L^{(l)}(q_i)|$ is defined as the confidence of node i after l iterations. This confidence criterion is used to rank the priority for shortening and puncturing according to the confidence value obtained after the specified number of iterations. The confidence ranking process is as follows:

Step 1: Initialization—The initial probability ratio for message $L(P_i)$ that the channel is transferred to a variable node is set to a fixed value, which is given as

$$L^{(0)}(q_{ij}) = L(P_i) = \ln \frac{P_i(0)}{P_i(1)} = \frac{2y_i}{\sigma^2} \, . \tag{12}$$

Step 2: Iterative decoding—After l iterations using $\boldsymbol{H_b}$ from the IEEE802.16e standard, the confidence $|L^{(l)}(q_i)|$ of each variable node is calculated. The optimal number of iterations for the LLR-BP decoding algorithm is between 20 and 25. Each variable node should not converge until it reaches the optimal number of iterations to achieve the best decoding performance.

Step 3: Ranking—The confidence $|L^{(l)}(q_i)|$ is ranked from largest to smallest and, at the same time, the ranking of $|L^{(l)}(q_i)|$ is used to generate a ranked list of positions corresponding to all the bits.

3.2 Shortening and fast coding based on the confidence criteria

The confidence $|L^{(l)}(q_i)|$ of each variable node can be calculated from Eq. (11). The check matrix of the LDPC code to be analyzed is ranked according to the confidence criteria, with the number for shortening being N_0. The system bits represented by the N_0 serial numbers are shortened. The specific operation of this process can be summarized as follows:

(1) On the basis of the confidence characteristics of the absolute value of the log-likelihood ratio (LLR) of the iterative decoding, the LDPC codes are ranked according to the confidence criteria, from small to large. The ranked relationship is stored in table T as follows.

$$T = (t_1, t_2, ..., t_k) \tag{13}$$

(2) Depending on the value of N_0, the transmitter selects the first N_0 bits from confidence table T for the bits to be shortened for checking in the LDPC code. Their locations are recorded and these locations are set to 0 at the end of the transmission. The remaining locations are filled with valid information. All of the information is encoded, including the added 0s, by the fast encoding algorithm. In other words, $p_1, p_2, ..., p_i$ are calculated using Eqs. (5)–(10). After the encoding is complete, all of the 0s are simply deleted from the message to be sent. The shortening process for the transmitter is illustrated in Fig. 1.

(3) The receiving end checks the position numbers of the check bits before selecting N_0 from the list of confidence criteria T for the LDPC code bits to be shortened, depending on the bit rate of the currently used LDPC code and the corresponding N_0. 0s are inserted into the corresponding positions. The BP decoding algorithm is then used to decode the message, and the bits of information initially set to zero are reset to infinity. The process followed by the shortened message receiver is shown in Fig. 2.

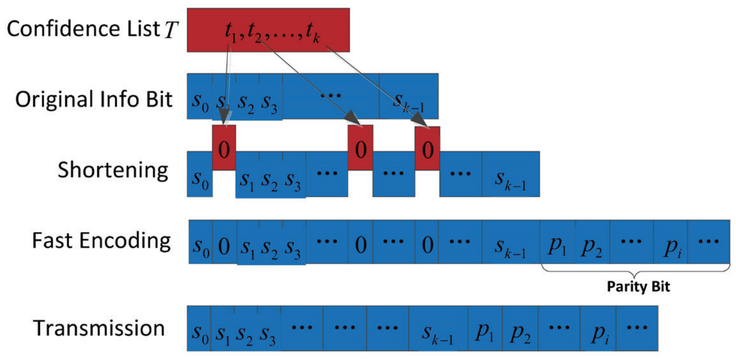
Confidence List T
$t_1, t_2, \ldots, t_k$
Original Info Bit
s_0 s_1 s_2 s_3 ⋯ s_{k-1}
Shortening
0 0 0
s_0 s_1 s_2 s_3 ⋯ ⋯ ⋯ s_{k-1}
Fast Encoding
s_0 0 s_1 s_2 s_3 ⋯ 0 ⋯ 0 ⋯ s_{k-1} p_1 p_2 ⋯ p_i ⋯
Parity Bit
Transmission
s_0 s_1 s_2 s_3 ⋯ ⋯ ⋯ s_{k-1} p_1 p_2 ⋯ p_i ⋯

Fig. 1. (Color online) Transmitter-based shortening.

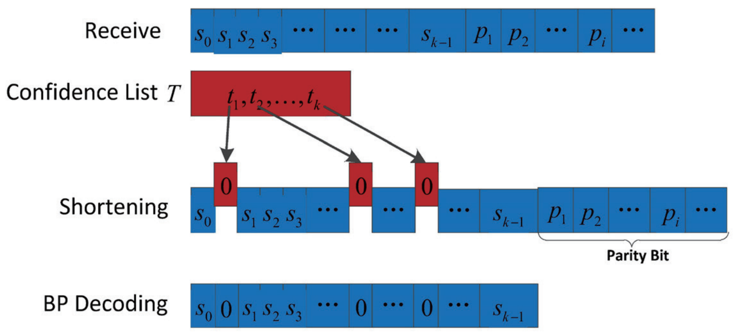
Receive
s_0 s_1 s_2 s_3 ⋯ ⋯ ⋯ s_{k-1} p_1 p_2 ⋯ p_i ⋯
Confidence List T
$t_1, t_2, \ldots, t_k$
Shortening
0
0
0
s_0 s_1 s_2 s_3 ⋯ ⋯ ⋯ s_{k-1} p_1 p_2 ⋯ p_i ⋯
Parity Bit
BP Decoding
s_0 0 s_1 s_2 s_3 ⋯ 0 ⋯ 0 ⋯ s_{k-1}

Fig. 2. (Color online) Receiver shortened message decoding.

3.3 Puncturing and fast coding based on the confidence criteria

We assume here that the rate of the R_m proxy is compatible with the bit rate of the LDPC group, with R_0 representing the code rate of the mother code and $R_0 < R_m$. For a mother code with a length of N, to obtain a code word with a code rate of R_m, the number of nodes to be punctured can be calculated as follows.

$$P_{num} = \frac{N\left(R_m - R_0\right)}{R_m} \tag{14}$$

The confidence $|L^{(l)}(q_i)|$ of each variable node is calculated using Eq. (11). The check matrix of the LDPC code to be analyzed is ranked according to the confidence criteria, and the required number of holes to be punctured is the check digit represented by the first N_p numbers. The specific way this works can be summarized as follows:

(1) The LDPC code corresponding to the selected parity check matrix is decoded and, according to the confidence characteristics of the absolute value of the LLR of the iterative decoding, the LDPC code parity bits are ranked from small to large. The ranked relationship is then stored in table T as follows.

$$T = (t_1, t_2, \ldots, t_{n-k}) \tag{15}$$

(2) According to the number of holes needing to be punched, the sending terminal selects the first N_p bits from table T and records their location. $p_1, p_2, \ldots, p_i$ are then calculated using Eqs. (5)–(10). The corresponding check bits of the LDPC code word are quickly encoded, then sent and deleted. This operation is shown in Fig. 3.

(3) According to the current length of the LDPC code, the receiving end selects the code rate and the corresponding number of punctured bits N_p. 0 is inserted into the parity bit position corresponding to the first N_p recorded numbers in T for the to-be-punctured LDPC code, and the BP algorithm decodes the message. This operation is shown in Fig. 4.

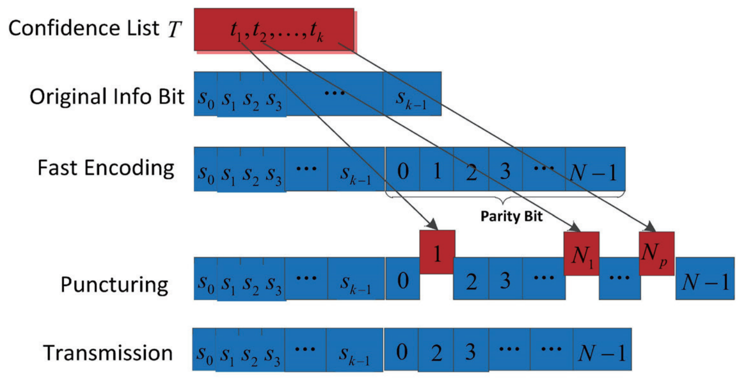

Fig. 3. (Color online) Transmitter-based puncturing.

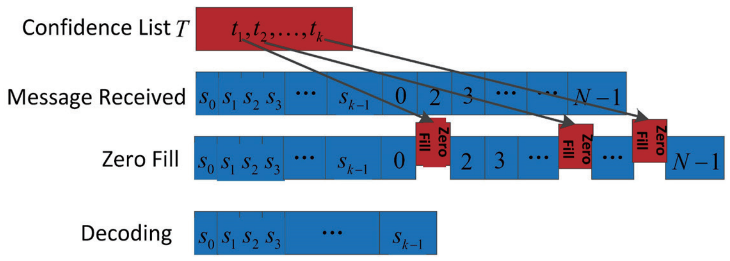

Fig. 4. (Color online) Receiver puncturing diagram.

4. Results and Discussion

To assess the viability of the method, the shortening and puncturing operations described above were simulated. When using a traditional BP decoding algorithm, the maximum number of iterations is 25.

To obtain an LDPC code for a code rate of 1/2 and a code length of 1056 according to the IEEE802.16e standard, LDPC codes with the same code rate and various code lengths were encoded using the shortening and confidence criteria techniques proposed above. The code length given by the encoding technique and the corresponding number of shortened nodes are shown in Table 1.

Taking the LDPC code produced according to the IEEE802.16e standard as the mother code requires a code length of 1536, a code rate of 1/2, and a median for the check matrix of 64. By the method proposed in this study, the code rate and the required number of nodes to be punctured are shown in Table 2.

Whenever the given code length is not within the 19 codes specified in the IEEE802.16e standard, i.e., if the length of the encoded information is less than one LDPC code word, our proposed shortening and fast coding method can be used instead.

It can be seen in Fig. 5 that the performance of the unshortened LDPC code is better than that of the shortened LDPC code. At the same time, the greater the number of nodes shortened,

Table 1
Code length and number of shortened nodes.

Code length	Number of shortened nodes
1008	48
1024	32
1040	16

Table 2
Code rate and number of nodes for puncturing.

Code rate	Number of nodes for puncturing
3/5	256
3/4	512
4/5	576
5/6	614

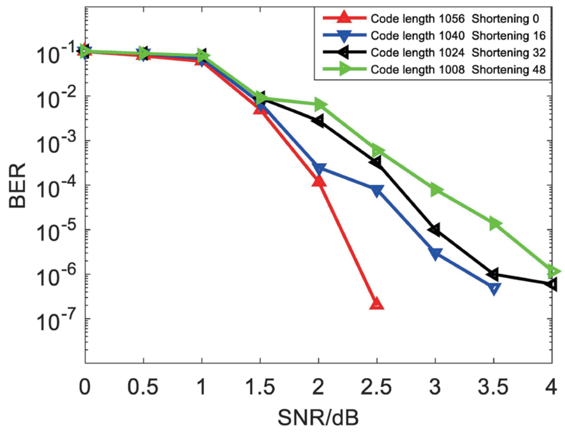

Fig. 5. (Color online) Shortened LDPC code performance.

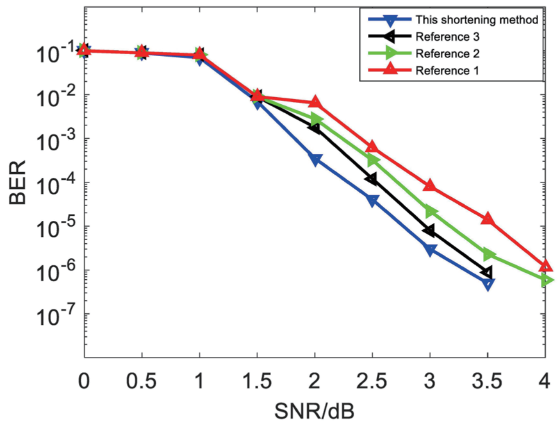

Fig. 6. (Color online) Comparison of the performance of the different shortening algorithms.

the faster the drop in the performance of the LDPC code. Note also that the greater the amount of shortening, the greater the difference will be between the given code length and the selected code length of 1056 in the IEEE802.16e standard. The code length given by the IEEE802.16e standard is closer to the fixed code length. We therefore sought to use this code length as the standard for fast coding.

To compare the performance of the proposed shortening method with existing shortening methods,(1–3) a code length of 1040 and a code rate of 1/2 were used for the mother code to shorten a code length of 1056 with the same code rate. Figure 6 shows the performance of the different shortening methods.

It can be seen that the performance of the other shortening methods was significantly better than that of the reverse-order shortening method.(1) Note especially that the performance of the shortening method proposed in this study was the best overall.

For situations where the required code rate is lower than the current fast code rate and this therefore needs to be improved, our proposed puncturing and fast code method can be used.

As illustrated in Fig. 7, the performance of the unpunctured LDPC code is better than that of the punctured LDPC code. The greater the amount of puncturing, the higher the code rate of the obtained LDPC code. However, the performance speed for the LDPC code decreases as the rate drops off. It is therefore advisable to use a higher code rate for the mother code to avoid an added loss of performance.

To compare the puncturing method proposed here with the performance of other puncturing methods,(3,5,7) we took a puncturing number of 256, a code length of 1536, and a code rate of

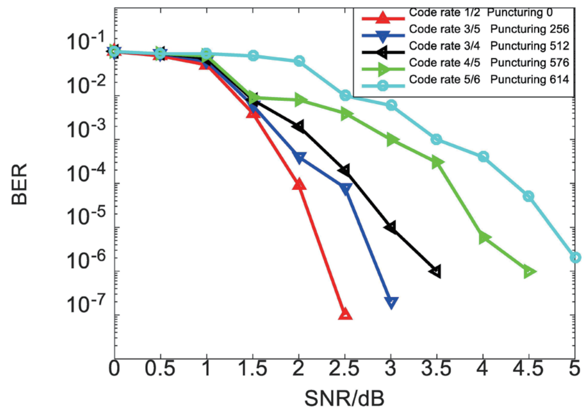

Fig. 7. (Color online) Punctured LDPC code performance.

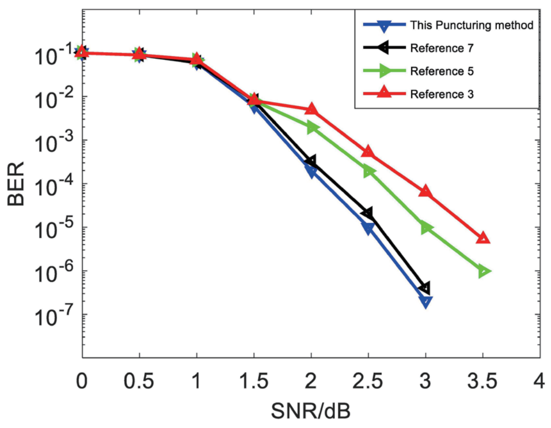

Fig. 8. (Color online) Comparison of the performance of different puncturing methods.

1/2, with a mother code rate of 3/5. Figure 8 shows the performance of the different puncturing methods.

It can be seen from Fig. 8 that the puncturing method proposed by Su and Shu achieved a similar performance to our approach.$^{(7)}$ However, these two approaches significantly outperform the other puncturing methods.

5. Conclusions

After an analysis of the shortcomings present in the IEEE802.16e standard relating to fixed code lengths and code rates, we have developed a fast adaptive LDPC coding method based on confidence criteria. It combines shortening and puncturing features based on the confidence criteria with the fast coding approach set out in the IEEE802.16e standards. The key objective was to improve the code lengths and code rates offered by LDPC adaptive dynamic adjustment. According to our simulation results, our approach improves the shortening performance for the LDPC code.

Acknowledgments

This work was supported by the Natural Science Foundation of Guangdong Province of China under Grant Nos. 2018A06001 and 2019A1515011164.

References

1 P802.11ma/D3.0—IEEE Draft Standard for Information Technology. https://doi.org/10.1109/IEEESTD.2012.6419735
2 X. Liu, X. Wu, and C. Zhao: IEEE Commun. Lett. **13** (2009) 612. https://doi.org/ 10.1109/LCOMM.2009.091026
3 Z. H. Chen, Y. Zhang, and G. Xie: J. Front. Comput. Sci. Technol. **9** (2015) 165. https://doi.org/10.3778/j.issn.1673-9418.1406025

4 Y. Xu, B. Liu, L. Gong, B. Rong, and L.Gui: IEEE Trans. Consum. Electron. **57** (2011) 105. https://doi.org/10.1109/TCE.2011.6018855

5 J. Ha, J. Kim, D. Klinc, and S. W. McLaughlin: IEEE Trans. Inf. Theory **52** (2006) 728. https://doi.org/10.1109/TIT.2005.862118

6 H. Y. Park, J. W. Kang, K. S. Kim, and K. C. Whang: IEEE Trans. Wireless Commun. **6** (2007) 3914. https://doi.org/10.1109/TWC.2007.060386

7 H. G. Su and S. T. Shu: J. Electron. Inf. Technol. **33** (2011) 2334. https://doi.org/10.3724/SP.J.1146.2010.01202

8 S. X. Wu: GLOBECOM 2009—2009 IEEE Global Telecommunications Conf. (2009) 1. https://doi.org/10.1109/GLOCOM.2009.5425627

9 M. Xiao, Y. Li, and L. Wang: J. Appl. Sci. **29** (2011) 385. https://doi.org/10.3969/j.issn.0255-8297.2011.04.009

10 B. N. Vellambi and F. Fekri. IEEE Trans. Commun. **57** (2009) 297. https://doi.org/10.1109/TCOMM.2009.02.060098

11 J. Y. Wang, Y. Li, and Y. Sun: J. Xidian Univ. **40** (2013) 13. https://doi.org/10.3969/j.issn.1001-2400.2013.02.003

12 L. Zang and Z. Q. Chen: Video Eng. **37** (2013) 105. https://doi.org/10.16280/j.videoe.2013.13.022.

13 F. Babich, M. Noschese, A. Soranzo, and F. Vatta: J. Commun. Software Syst. **14** (2018) 1. https://doi.org/10.24138/jcomss.v14i4.639

14 K. Deka, A. Rajesh, and P. K. Bora: 2016 Int. Conf. Signal Processing Communications (SPCOM) (2016) 1. https://doi.org/ 10.1109/SPCOM.2016.7746650

15 W. W. Li, J. Lei, E. Li, W. Yang, and Y. Q. Xiong: 2017 4th Int. Conf. Information Science Control Engineering (ICISCE) 1564. https://doi.org/10.1109/ICISCE.2017.372

16 L. Zhou, W. C. Huang, R. Zhao, and Y. He: J. Commun. Networks **19** (2017) 124. https://doi.org/10.1109/JCN.2017.000006

17 H. Li and L. Zheng: IEEE Commun. Lett. **19** (2015) 1508. https://doi.org/10.1109/LCOMM.2015.2453314

18 Y. Guo and M. Wang: Software Guide **12** (2013) 156. https://doi.org/CNKI:SUN:RJDK.0.2013-10-060

19 K. He, J. Q. Gao, and W. X. Lu: J. Tianjin Univ. **50** (2017) 399. https://doi.org/10.11784/tdxbz201603091

20 A. Nan, X. Xi: IEEE 2014 7th Int. Conf. Biomedical Engineering and Informatics, Dalian (IEEE, 2014) 885–889. https://doi.org/10.1109/BMEI.2014.7002897

21 M. Zhao and X. L. Zhang: J. Beijing Univ. Aeronaut. Astronaut. **40** (2014) 350. https://doi.org/10.13700/j.bh.1001-5965.2013.0254

22 Y. N. Fan, W. Zhang, L. C. Wang, and X. J. Yao: J. Xidian Univ. **44** (2017) 88. https://doi.org/10.3969/j.issn.1001-2400.2017.02.016