IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 30, 2018, accepted August 30, 2018, date of publication September 10, 2018, date of current version September 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2869374

Decoding Optimization for 5G LDPC Codes by Machine Learning

XIAONING WU, (Student Member, IEEE), MING JIANG, (Member, IEEE), AND CHUNMING ZHAO, (Member, IEEE)

National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China

Corresponding author: Ming Jiang (jiang_ming@seu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61771133 and Grant 61521061, and in part by the National Science & Technology Projects of China under Grant 2018ZX03001002.

ABSTRACT In this paper, we propose a generalized minimum-sum decoding algorithm using a linear approximation (LAMS) for protograph-based low-density parity-check (PB-LDPC) codes with quasi-cyclic (QC) structures. The linear approximation introduces some factors in each decoding iteration, which linearly adjust the check node updating and channel output. These factors are optimized iteratively using machine learning, where the optimization can be efficiently solved by a small and shallow neural network with training data produced by the LAMS decoder. The neural network is built according to the parity check matrix of a PB-LDPC code with a QC structure which can greatly reduce the size of the neural network. Since, we optimize the factors once per decoding iteration, the optimization is not limited by the number of the iterations. Then, we give the optimized results of the factors in the LAMS decoder and perform decoding simulations for PB-LDPC codes in fifth generation mobile networks (5G). In the simulations, the LAMS algorithm shows noticeable improvement over the normalized and the offset minimum-sum algorithms and even better performance than the belief propagation algorithm in some high signal-to-noise ratio regions.

INDEX TERMS Iterative decoding, parity check codes, neural networks, optimization, machine learning.

I. INTRODUCTION

Low-density parity-check (LDPC) codes have been aroused much attention during the past decades. Protograph-based LDPC (PB-LDPC) codes [1] gradually become a mainstream encoding technology for various communication systems. The landmark event is that the enhanced mobile broadband (EMBB) scenario in 5th generation mobile networks (5G) adopts two sets of PB-LDPC codes defined by the parity check matrices, BG1 and BG2 [2]. The BG1 and the BG2 matrices are constructed from the raptor-like protographs [3] with quasi-cyclic (QC) structures, which can support the rate compatible coding and scalable information transmission from dozens to thousands.

Although the belief propagation (BP) algorithm can achieve near-optimal performance for the 5G LDPC codes, the simplified versions of the BP algorithm, such as the minimum-sum (MS), normalized MS (NMS), and offset MS (OMS) algorithms, are widely used in practical applications thanks for their easy hardware implementation [4]–[6].

Here, the NMS and the OMS algorithms can achieve noticeable performance improvement on the MS decoding with the normalized and the offset factors, respectively. Since the tanh functions are replaced by the comparison operations in the calculations of the check node updating, there are substantial performance losses for the simplified algorithms compared with the BP algorithm, especially for the low rate, moderate and short length, and finite iterations conditions.

Usually, these BP-based simplified algorithms involve many factor settings which are optimized by the threshold analysis or simulations. In [4] and [6]–[11], the threshold analysis is used to optimize the factors under the assumptions of infinite code length and cycle-free, which is not in line with actual conditions. In Xu *et al.* [10], Oh and Parhi [12], and Wu *et al.* [13] also optimize the factors by simulations, but that makes the optimization very time consuming and inefficient.

Recently, machine learning provides a new perspective for decoding optimization. In [14]–[16], the BP and the MS decoders for short length high-density parity-check (HDPC) codes are optimized by deep recurrent neural networks. The standard decoders are significantly improved by optimizing different factors for the output of each variable node and check node in each iteration. However, the size of the network

is increased with the code length and the maximum number of the iterations. For the longer code length and more iterations, the network may be too complicated to be optimized. In [17], the idea of using deep neural networks for decoding of the random and the polar codes is revisited. In [18], a method from the field of machine learning is applied to construct the discrete message passing decoders for regular LDPC codes. In [19], a joint quantizer is optimized by a neural network for the BP decoder. In [20], the authors extend the syndrome decoding and propose two deep neural network architectures for decoding of linear codes. In [21], a novel iterative architecture concatenating a trained convolutional neural network with a standard BP decoder for channel decoding is proposed.

In this paper, we propose a method based on machine learning to optimize the simplified BP-based algorithms. Firstly, we propose a generalized MS algorithm using a linear approximation (LAMS), where the magnitudes of the check node output are modified not only by a normalized or an offset factor like the NMS or the OMS algorithm but by a linear function containing both the factors. Besides, the LAMS algorithm also modifies the output from the channel with a similar linear function. Then, we build a neural network through the graph of a PB-LDPC code with a QC structure and configure the parameters of the LAMS algorithm into the corresponding network for optimization.

Our neural network is not deep because it only runs one iteration process of decoding and gets the corresponding optimized factors. When the current iteration is finished, the optimized factors are saved to calculate the input of the network for the next iteration. The optimization of the next iteration is started when the neural network is reinitialized and the input is sent back. We use the local optimal solution of each iteration instead of the global optimal solution of all iterations to reduce the network depth and computational complexity. With help of the QC structures of 5G LDPC codes, the complexity of our neural network will not significantly increase with the code length, where multiple bits can be processed in parallel. The number of operation units in each layer of the neural network is also directly proportional to the scale of a protograph which is much smaller than that of the parity-check matrix.

Since the neural network is not too much complex, we can optimize the LAMS decoder for LDPC codes with thousands of code lengths and dozens of iteration numbers. We use the LAMS algorithm on the 5G BG2 matrix and perform the simulations for different code lengths, code rates, and numbers of the iterations. The simulation results show that the LAMS algorithm has obvious advantages over the NMS and the OMS algorithms. As the signal-to-noise ratio (SNR) increases and the block error rate (BLER) decreases, the gap between the LAMS and the BP algorithms gradually decreases. For some conditions with high SNRs, the BLER performance of the LAMS algorithm exceeds that of the BP algorithm.

II. GENERALIZED MS DECODING USING A LINEAR APPROXIMATION

An LDPC code is defined by a parity-check matrix $\mathbf{H}$, which has M parity-check constraints among the N coded bits. The matrix $\mathbf{H}$ is an $M \times N$ sparse binary matrix, where the number of the 1-elements is $J \ll MN$. The matrix can be represented by a bipartite graph, which is formed by N variable nodes $v_n, n \in \mathcal{N} = \{0, 1, \cdots, N-1\}$, M check nodes $c_m, m \in \mathcal{M} = \{0, 1, \cdots, M-1\}$, and J edges $e_j, j \in \mathcal{J} = \{0, 1, \cdots, J-1\}$. The j-th edge connects the m-th check node and the n-th variable node if the m-th row and the n-th column of the matrix $\mathbf{H}$ is a 1-element, which is the j-th 1-element from left to right and from top to bottom in $\mathbf{H}$. The neighbors of the variable node v_n are the check nodes $c_m, m \in \mathcal{A}(n)$, where $\mathcal{A}(n)$ is the index set of the check nodes connected to the node v_n. Similarly, the neighbors of the check node c_m are the variable nodes $v_n, n \in \mathcal{B}(m)$, where $\mathcal{B}(m)$ is the index set of the variable nodes participating in the check node c_m.

The K information bits are coded by the matrix $\mathbf{H}$, and N coded bits $w_n, n \in \mathcal{N}$ are generated. Among the N coded bits, $\tilde{N}$ coded bits are BPSK modulated and transmitted, which means $N-\tilde{N}$ coded bits are punctured. The $\tilde{N}$ bits are sent to a receiver through an additive white Gaussian noise (AWGN) channel with the mean 0 and the variance σ^2. The receiver takes the $\tilde{N}$ received signals and $N-\tilde{N}$ zero signals as channel outputs, defined by $r_n, n \in \mathcal{N}$, into an LDPC decoder, where the $N-\tilde{N}$ zero signals are corresponding to the punctured bits that are not transmitted through the channel.

The BP decoder gets the channel output signal r_n and calculates the corresponding log-likelihood ratio (LLR), defined as l_n^{ch}, for the n-th coded bit w_n (or the variable node v_n) as follows,

$$l_n^{ch} = \log \frac{\Pr(w_n = 0|r_n)}{\Pr(w_n = 1|r_n)} = \frac{2r_n}{\sigma^2} \tag{1}$$

where $\Pr(w_n = 0|r_n)$ and $\Pr(w_n = 1|r_n)$ are conditional probabilities of $w_n = 0$ and $w_n = 1$ given r_n, respectively. The channel LLR l_n^{ch} represents how likely the coded bit w_n is 0 or 1. The larger the magnitude of l_n^{ch}, the higher the certainty of that the code bit w_n is 0 or 1. If $l_n^{ch} \geq 0$, that $\Pr(w_n = 0|r_n) \geq \Pr(w_n = 1|r_n)$, the original code bit is more likely 0, otherwise, 1 is more likely for the code bit. Based on the channel LLRs l_n^{ch} and the parity-check matrix $\mathbf{H}$, the BP decoder iteratively updates the LLRs between the variable nodes and the check nodes in each iteration to obtain the LLR output for each bit.

At the beginning of the BP decoding, the 0-th iteration, the LLR sent from the variable node $v_n, n \in \mathcal{N}$ to the neighbor check node $c_m, m \in \mathcal{A}(n)$ is denoted as follows,

$$l_{n\to m}^0 = l_n^{ch} \tag{2}$$

For each iteration $i = 0, 1, \cdots, I-1$, the check node $c_m, m \in \mathcal{M}$ receives $l_{n\to m}^i$ from its neighbors $v_n, n \in \mathcal{B}(m)$ and gives backward LLRs to the corresponding variable

nodes, which can be formulated as follows,

$$l^i_{m\to n} = 2\tanh^{-1}\left(\prod_{n'\in\mathcal{B}(m)\backslash n}\tanh\left(\frac{l^i_{n'\to m}}{2}\right)\right) \tag{3}$$

where $\mathcal{B}(m)\backslash n$ is the index set of the variable nodes $\mathcal{B}(m)$ excluding n.

When $l^i_{m\to n}$ comes back to the variable node v_n, the output LLR for current iteration, denoted as l^i_n, can be formulated as follows,

$$l^i_n = l^{ch}_n + \sum_{m\in\mathcal{A}(n)} l^i_{m\to n}. \tag{4}$$

Here, l^i_n can be used to obtain the tentative hard decision of coded bit w^i_n for the i-th iteration, where the N coded bits w^i_n form a column vector $\mathbf{w}^i$. If all the tentative hard decision of LLR outputs match the check equations in $\mathbf{H}$, namely $\mathbf{H}\mathbf{w}^i = 0$ in the Galois field GF(2), the decoding is successful, otherwise, the BP decoding is carried out for the next iteration until the maximum number of iteration I is reached.

For the next iteration $i+1$, the updated LLR should be sent from the variable node $v_n, n \in \mathcal{N}$ to the check node $c_m, m \in \mathcal{A}(n)$ again, which is different from the beginning of the decoding in Equation (2). It can be represented as follows,

$$l^{i+1}_{n\to m} = l^i_n - l^i_{m\to n} \tag{5}$$

where $l^i_{m\to n}$ and l^i_n are the outputs of the check nodes and the variable nodes in the previous iteration, respectively. Then, the iterative decoding is carried on until the decoding is finished.

To reduce the complexity of the BP algorithm, a simplified BP-based decoding, the MS algorithm, is widely used in many practical applications. The tanh function in Equation (3) can be approximately replaced with the min-sum simplification as follows,

$$l^i_{m\to n} = \left(\prod_{n'\in\mathcal{B}(m)\backslash n}\operatorname{sign}\left(l^i_{n'\to m}\right)\right)\cdot\min_{n'\in\mathcal{B}(m)\backslash n}\left(\left|l^i_{n'\to m}\right|\right) \tag{6}$$

where the function sign() gets the sign of the parameter. Note that sign(0) is defined as 0.

Since the MS decoding only includes the additions and the comparisons, it is unnecessary to estimate the channel variance σ^2. The initialization of the channel LLR is changed from Equation (1) as follows,

$$l^{ch}_n = \frac{\Pr(w_n = 0|r_n)}{\Pr(w_n = 1|r_n)} = r_n \tag{7}$$

where the noise variance σ^2 is omitted in the calculation and the MS decoding is free from the estimation error of the variance.

However, the MS approximation incurs considerable performance loss compared with the BP algorithm, especially for the low-rate LDPC codes. To enhance the performance of the MS algorithm, the normalized MS (NMS) algorithm is proposed as follows,

$$l^i_{m\to n} = \alpha\cdot\left(\prod_{n'\in\mathcal{B}(m)\backslash n}\operatorname{sign}\left(l^i_{n'\to m}\right)\right)\cdot\min_{n'\in\mathcal{B}(m)\backslash n}\left(\left|l^i_{n'\to m}\right|\right) \tag{8}$$

where the normalized factor α is introduced to minimize the approximation error of the min-sum simplification.

Similar with the NMS algorithm, the OMS algorithm improves the performance of the MS algorithm by an offset factor shown as follows,

$$l^i_{m\to n} = \left(\prod_{n'\in\mathcal{B}(m)\backslash n}\operatorname{sign}\left(l^i_{n'\to m}\right)\right)\cdot\max\left(\min_{n'\in\mathcal{B}(m)\backslash n}\left(\left|l^i_{n'\to m}\right|\right)+\beta, 0\right) \tag{9}$$

where β is the offset factor and the function max() is for guaranteeing the sign of the check output.

The two factors, α and β, only make multiplicative and additive corrections to the amplitude $\left|l^i_{n'\to m}\right|$, which lacks room for approximation. Combining the two factors, we propose a linear approximation of the amplitude $\left|l^i_{n'\to m}\right|$, which introduces the factors α_i, β_i for each iteration. Based on Equation (8) and (9), the approximation is described as follows,

$$l^i_{m\to n} = \left(\prod_{n'\in\mathcal{B}(m)\backslash n}\operatorname{sign}\left(l^i_{n'\to m}\right)\right)\cdot\max\left(\alpha_i\cdot\min_{n'\in\mathcal{B}(m)\backslash n}\left(\left|l^i_{n'\to m}\right|\right)+\beta_i, 0\right) \tag{10}$$

where the factors α_i, β_i only have a linear impact on the magnitude of the LLR and do not change the sign of that.

Besides, the factors α^{ch}_i, β^{ch}_i are also used for linear adjusting the magnitude of the channel LLR as follows,

$$l^i_n = \operatorname{sign}\left(l^{ch}_n\right)\cdot\max\left(\alpha^{ch}_i\left|l^{ch}_n\right|+\beta^{ch}_i, 0\right)+\sum_{m\in\mathcal{A}(n)} l^i_{m\to n} \tag{11}$$

where the equations are changed from Equation (4). The channel LLR l^{ch}_n is calculated by Equation (7). Here, we keep the sign of the channel LLR and only linearly change the magnitude of that.

The complete LAMS algorithm is shown in Algorithm 1, where the channel output r_n is processed to get the final coded bits $\mathbf{w} = \mathbf{w}^i$. Sine the four factors are very difficult to be optimized for each iteration by the traditional methods of threshold analysis and simulations, we use machine learning for optimization in the next section.

III. THE DECODING OPTIMIZATION BY MACHINE LEARNING

The factors α_i, β_i, α^{ch}_i, and β^{ch}_i of each iteration $i = 0, 1, \cdots, I-1$ in the LAMS algorithm can be optimized iteratively using machine learning as shown in Fig. 1 (a).

Algorithm 1 The LAMS Decoding Algorithm

{Initialization:}
for all $n \in \mathcal{N}$ **do**
  for all $m \in \mathcal{A}(n)$ **do**
    Set $l^0_{n\to m} = l^{ch}_n = r_n$
  end for
end for
Set $i = 0$
repeat
  {Check Nodes Update:}
  for all $m \in \mathcal{M}$ **do**
    for all $n \in \mathcal{B}(m)$ **do**
      Get $l^i_{m\to n}$ by Equation (10)
    end for
  end for
  {Output Calculation:}
  for all $n \in \mathcal{N}$ **do**
    Get l^i_n by Equation (11)
    Get $w^i_n = (1 - \text{sign}(l^i_n))/2$
  end for
  Get $\mathbf{w}^i = [w^i_0, w^i_1, \cdots, w^i_{N-1}]^T$
  {Variable Nodes Update for Next Iteration:}
  for all $n \in \mathcal{N}$ **do**
    for all $m \in \mathcal{A}(n)$ **do**
      Get $l^{i+1}_{n\to m}$ by Equation (5)
    end for
  end for
  Get $i = i + 1$
until $i = I$ or $\mathbf{Hw}^i = \mathbf{0}$
return $\mathbf{w}^i$

The machine learning uses a neural network with only three layers in Fig. 1 (b), which is fed by a database of LLRs for training. In the first layer, called the input layer, the received LLRs of the neural network are assigned to the next layer like the message passing from variable nodes to check nodes. In the second layer, called the hidden layer, the LLRs are calculated and passed forward to the last layer, where the procedure is like the check node updating by Equation (10) in the LAMS algorithm. In the last layer, called the output layer, the LLRs are finally updated according to Equation (11). Similar with one iteration of the LDPC decoding, the neural network controls the flow of LLRs and optimizes the four factors in the second and last layer. With large amounts of the LLR data, the neural network can train the factors in the i-th iteration, α_i, β_i, α^{ch}_i, β^{ch}_i, to minimize the gap between the output and the target codes by the stochastic gradient descent. When the optimization of the factors is finished, the four optimized factors are saved and used to calculate the LLRs for the next iteration by the LAMS decoder described in Algorithm 1. Then, the LLRs are sent back as the training data to start the optimization for the next iteration, where the neural network should have been initialized.

Since the LDPC codes in 5G new radio (NR) belong to the family of PB-LDPC codes with QC structures, an operation

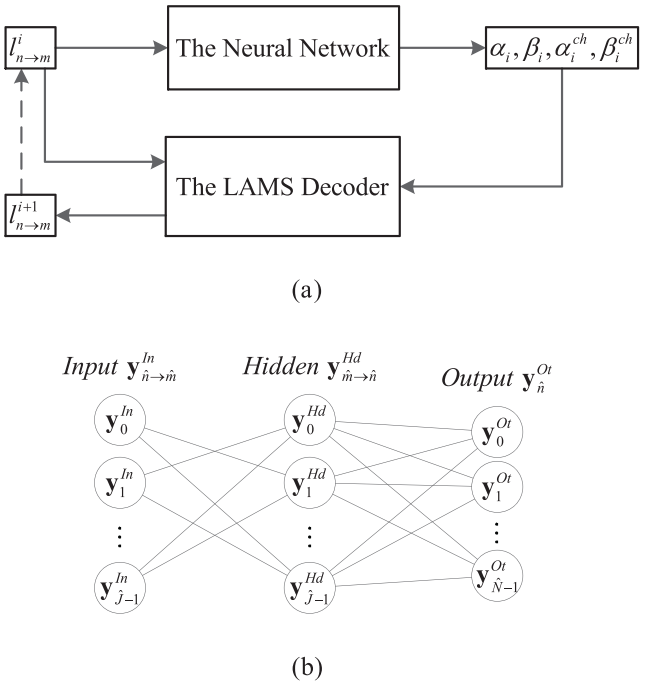

FIGURE 1. (a) The procedure of the optimization using the neural network and the LAMS decoder iteratively (b) The neural network for optimization with the input, the hidden, and the output layers which own $\hat{J}$, $\hat{J}$, and $\hat{N}$ operation units connected according to the protograph, respectively.

unit of the neural network in Fig. 1 (b) can deal with a set of Z bits simultaneously, where Z is the lifting factor of a QC matrix. That makes the number of operation units equal to the number of edges or nodes in a protograph. The protograph is usually considered as a small bipartite graph containing $\hat{N}$ variable nodes and $\hat{M}$ check nodes which are interconnected by $\hat{J}$ edges. To form a derived graph corresponding to an LDPC code, the protograph should be lifted Z times, where each edge is copied Z times and permuted in the same edge type across the Z different copies of the variable nodes and the check nodes in the same node type. In one edge type or one node type, Z edges or Z nodes form an edge set or a node set in the derived graph, which is corresponding to an edge or a node in the protograph. Therefore, the type or the set numbers of the edges, the variable nodes, and the check nodes in the derived graph are also $\hat{J}$, $\hat{N}$ and $\hat{M}$, respectively.

Note that, the PB-LDPC codes in our paper are all considered to be QC-LDPC codes, where the cyclic permutation is used in the lifting process. Since the graphs of the PB-LDPC codes are lifted from a small protograph with cyclic permutations, the parity check matrix of a PB-LDPC code can be represented with a QC structure as follows,

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{0,0} & \mathbf{H}_{0,1} & \cdots & \mathbf{H}_{0,\hat{N}-1} \\ \mathbf{H}_{1,0} & \mathbf{H}_{1,1} & \cdots & \mathbf{H}_{1,\hat{N}-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_{\hat{M}-1,0} & \mathbf{H}_{\hat{M}-1,1} & \cdots & \mathbf{H}_{\hat{M}-1,\hat{N}-1} \end{bmatrix} \quad (12)$$

where $\mathbf{H}_{\hat{m},\hat{n}}, \hat{m} = 0, 1, \cdots, \hat{M}-1, \hat{n} = 0, 1, \cdots, \hat{N}-1$ is a $Z \times Z$ binary circulant permutation matrix (CPM) with

only one non-zero element in each row or the all-zero sub-matrix. In the CPM, each row is cyclically right shifted by one position compared with the previous row. If the non-zero element in the first row is the p-th element with $p = 0, 1, \cdots, Z-1$, the shift value of this CPM is defined as p. The matrix $\mathbf{H}$ with $M = \hat{M}Z$ rows and $N = \hat{N}Z$ columns are divided into $\hat{M} \times \hat{N}$ blocks formed by the CPMs and the all-zero sub-matrices, which can be reduced to a simplified matrix $\mathbf{P}$ of the shift values. The matrix $\mathbf{P}$ with the size $\hat{M} \times \hat{N}$ is filled with the values of p and -1 which represent the CPMs and the all-zero sub-matrix in the matrix $\mathbf{H}$, respectively. According to the matrix $\mathbf{P}$, we can easily build the matrix $\mathbf{H}$.

With help of the QC structures in the PB-LDPC codes, we can process a block of data of the size Z in an operation unit of the neural network. Furthermore, considering the batch size S in the training step, we set the tensors in the neural network with the shape of the first dimension S and the second dimension Z.

The output of the operation unit in the input layer with the shape size 2 from the $\hat{n}$-th variable node set to the $\hat{m}$-th check node set, defined by $\mathbf{y}^{In}_{\hat{n}\to\hat{m}}$, can be described by a matrix as follows,

$$\mathbf{y}^{In}_{\hat{n}\to\hat{m}} = \begin{bmatrix} y_{0,0} & y_{0,1} & \cdots & y_{0,Z-1} \\ y_{1,0} & y_{1,1} & \cdots & y_{1,Z-1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{S-1,0} & y_{S-1,1} & \cdots & y_{S-1,Z-1} \end{bmatrix} \tag{13}$$

where $y_{s,z}, s = 0, 1, \cdots, S-1, z = 0, 1, \cdots, Z-1$ is the abbreviated form of the symbol $y^{In}_{\hat{n}\to\hat{m}}(s,z)$ representing the z-th element from the $\hat{n}$-th variable node set to the $\hat{m}$-th check node set in the s-th sample of the batch for the optimization in the input layer. Note that, the operation unit $\mathbf{y}^{In}_{\hat{n}\to\hat{m}}$ corresponds to the $\hat{j}$-th operation unit $\mathbf{y}^{In}_{\hat{j}}, \hat{j} = 0, 1, \cdots, \hat{J}-1$ of the neural network with the size $\hat{J}$ in Fig. 1 (b). When training, the input layer is fed by the corresponding LLR in the i-th iteration as the equation $y^{In}_{\hat{n}\to\hat{m}}(s,z) = l^{i}_{\hat{n}\to\hat{m}}(s,z)$.

In the hidden layer, the elements of the tensor from the input layer should be circularly shifted along the second axis to make them conducted in parallel. The shift function with the right shift value p for the matrix $\mathbf{y}^{In}_{\hat{n}\to\hat{m}}$ is defined as $\text{Roll}(\mathbf{y}^{In}_{\hat{n}\to\hat{m}}, p)$, where each element of the shifted matrix is denoted by $y^{p}_{s,z}$. The relationship between $y^{p}_{s,z}$ and $y_{s,z}$ can be formulated as follows,

$$y^{p}_{s,z} = y_{s,\ \mathrm{mod}\ (z-p,Z)} \tag{14}$$

where $y_{s,\ \mathrm{mod}\ (z-p,Z)}$ is the s-th row and $\mathrm{mod}\ (z-p, Z)$-th column element in $\mathbf{y}^{In}_{\hat{n}\to\hat{m}}$. For example, the matrix after circular right shift with the value $p = 1$ can be described by $y_{s,z}$ as follows,

$$\text{Roll}\left(\mathbf{y}^{In}_{\hat{n}\to\hat{m}}, 1\right) = \begin{bmatrix} y_{0,Z-1} & y_{0,0} & \cdots & y_{0,Z-2} \\ y_{1,Z-1} & y_{1,0} & \cdots & y_{1,Z-2} \\ \vdots & \vdots & \ddots & \vdots \\ y_{S-1,Z-1} & y_{S-1,0} & \cdots & y_{S-1,Z-2} \end{bmatrix} \tag{15}$$

where all the columns of the matrix are circularly shifted one place to the right. Suppose that the shift values of the $\hat{j}'$-th edge set from $\hat{n}'$ to $\hat{m}$ and the $\hat{j}$-th edge set from $\hat{n}$ to $\hat{m}$ are p' and p, respectively, the elements of the tensor from the input layer should be circularly right shifted as the following equation,

$$\mathbf{y}^{In,Rl}_{\hat{n}'\to\hat{m}} = \text{Roll}\left(\mathbf{y}^{In}_{\hat{n}'\to\hat{m}}, p-p'\right) \tag{16}$$

where the right shift $p-p'$ is equivalent to the left shift $p'-p$.

Then, the output of the $\hat{j}$-th operation unit in the hidden layer from the $\hat{m}$-th check node set to the $\hat{n}$-th variable node set, defined as $\mathbf{y}^{Hd}_{\hat{m}\to\hat{n}}$, can be calculated by processing the corresponding elements in the matrix $\mathbf{y}^{In,Rl}_{\hat{n}'\to\hat{m}}$ in parallel.

Each element of $\mathbf{y}^{Hd}_{\hat{m}\to\hat{n}}$ is formulated as follows,

$$y^{Hd}_{\hat{m}\to\hat{n}}(s,z) = \left(\prod_{\hat{n}'\in\mathcal{B}(\hat{m})\backslash\hat{n}} \text{sign}\left(y^{In,Rl}_{\hat{n}'\to\hat{m}}(s,z)\right)\right) \cdot \max\left(\alpha \cdot \min_{\hat{n}'\in\mathcal{B}(\hat{m})\backslash\hat{n}} \left(\left|y^{In,Rl}_{\hat{n}'\to\hat{m}}(s,z)\right|\right) + \beta, 0\right) \tag{17}$$

where $y^{In,Rl}_{\hat{n}'\to\hat{m}}(s,z)$ is the s-th row and z-th column element in the matrix $\mathbf{y}^{In,Rl}_{\hat{n}'\to\hat{m}}$. Here, the variables α and β in the neural network will be optimized by training data and assigned to α_i and β_i in the i-th iteration of the LAMS decoder when the training is finished.

The final output $\mathbf{y}^{Ot}_{\hat{n}}$ in the last layer is an $S \times Z$ tensor matrix for the $\hat{n}$-th variable node set. With the data from the hidden layer, the result of each element in the output $\mathbf{y}^{Ot}_{\hat{n}}$ is as follows,

$$y^{Ot}_{\hat{n}}(s,z) = \text{sign}\left(l^{ch}_{\hat{n}}(s,z)\right) \cdot \max\left(\alpha^{ch} \cdot \left|l^{ch}_{\hat{n}}(s,z)\right| + \beta^{ch}, 0\right) + \sum_{\hat{m}\in\mathcal{A}(\hat{n})} y^{Hd}_{\hat{m}\to\hat{n}}(s,z) \tag{18}$$

where $l^{ch}_{\hat{n}}(s,z)$ is the s-th row and z-th column element of the corresponding channel LLR matrix with size $S \times Z$ for the $\hat{n}$-th variable node set. Similar with α_i and β_i, $\alpha^{ch}_i = \alpha^{ch}$ and $\beta^{ch}_i = \beta^{ch}$ are assigned when the training are finished.

The output of the third layer is the estimation of the neural network for the $\hat{N}$ variable node sets. Before a loss function is calculated, they should be preprocessed as follows,

$$o_{\hat{n}} = \frac{1}{SZ} \sum_{s=0}^{S-1} \sum_{z=0}^{Z-1} \frac{1}{1 + e^{-y^{Ot}_{\hat{n}}(s,z)}} \tag{19}$$

where we use the sigmoid function $\text{sigmoid}\,(x) = \frac{1}{1+e^{-x}}$ to process each element of $\mathbf{y}^{Ot}_{\hat{n}}$, and then, the mean function $\text{mean}(\mathbf{x}) = \frac{1}{SZ} \sum_{s=0}^{S-1} \sum_{z=0}^{Z-1} x(s,z), \mathbf{x} = [x(s,z)]^{S\times Z}$ to output a value averaging them.

The loss between the estimation and the target should be minimized. The loss function uses the cross entropy function

shown as follows,

$$Loss = -\frac{1}{\hat{N}} \sum_{\hat{n}=0}^{\hat{N}-1} \left(g_{\hat{n}} \log \left(o_{\hat{n}} \right) + \left(1 - g_{\hat{n}} \right) \log \left(1 - o_{\hat{n}} \right) \right) \quad (20)$$

where $g_{\hat{n}}$ is the target of the output and set to 1 for conveniences [16].

To minimize the loss function, we use the stochastic gradient descent to train the four factors, α, β, α^{ch}, and β^{ch}. When Equation (17) and (18) are non-differentiable at some points, the sub-gradient is chosen during back-propagation. Finally, the optimized values of the four factors are assigned to the factors in the i-th iteration of our LAMS algorithm, α_i, β_i, α_i^{ch}, and β_i^{ch}. For the optimization of the next iteration $i+1$, the neural network is initialized and fed with the LLRs calculated by equation (5). In the first iteration with $i = 0$, the neural network is fed with the LLRs from channel shown as Equation (7).

TABLE 1. The table of the normalized and the offset factors optimized by the neural network for each iteration.

i	α_i	β_i	α_i^{ch}	β_i^{ch}	i	α_i	β_i	α_i^{ch}	β_i^{ch}
0	0.8	0	1.5	0.1	15	0.9	-0.2	1.3	0.2
1	0.7	-0.2	1.5	0.1	16	0.8	-0.2	1.3	0.2
2	0.7	-0.2	1.5	0.1	17	0.9	-0.2	1.3	0.2
3	0.8	-0.2	1.5	0.1	18	0.8	-0.2	1.3	0.2
4	0.8	-0.2	1.5	0.1	19	0.8	-0.2	1.3	0.2
5	0.8	-0.2	1.5	0.1	20	0.8	-0.2	1.3	0.2
6	0.8	-0.2	1.4	0.1	21	0.8	-0.2	1.2	0.2
7	0.9	-0.2	1.4	0.1	22	0.8	-0.2	1.2	0.2
8	0.9	-0.2	1.4	0.1	23	0.8	-0.2	1.2	0.1
9	0.9	-0.2	1.4	0.1	24	0.9	-0.2	1.2	0.1
10	0.9	-0.3	1.4	0.2	25	0.8	-0.2	1.2	0.1
11	0.9	-0.3	1.4	0.2	26	0.8	-0.1	1.2	0.1
12	0.9	-0.3	1.4	0.2	27	0.8	-0.2	1.2	0.1
13	0.9	-0.3	1.3	0.2	28	0.8	-0.1	1.2	0.1
14	0.9	-0.3	1.3	0.2	29	0.8	-0.1	1.2	0.1

IV. SIMULATION RESULTS

We use Tensorflow to train the neural network with the learning rate 0.1, the number of training 500, the batch size 500. The initial values of the four factors, α, β, α^{ch}, and β^{ch} are set to 0.7, 0, 1, and 0, respectively. To make the factors more practical for implementation, we set the precision of the four factors to 0.1, which means we round the value after training. The neural network is built according to the rate-1/3 parity check matrix of BG2 [2] with the lifting size $Z = 52$ and the information length $K = 520$. The initial training data of LLRs are collected by taking the BPSK modulated signals from the all-zero coded bits through the AWGN channel with the SNR, $E_s/N_0 = -4.0$ (dB), where the first $2Z = 104$ bits of the codeword are punctured. If a large number of intermediate variables do not have sufficient memory space to store, we will use a binary file to read and write the variables on the hard disk. The training results are shown in Table 1,

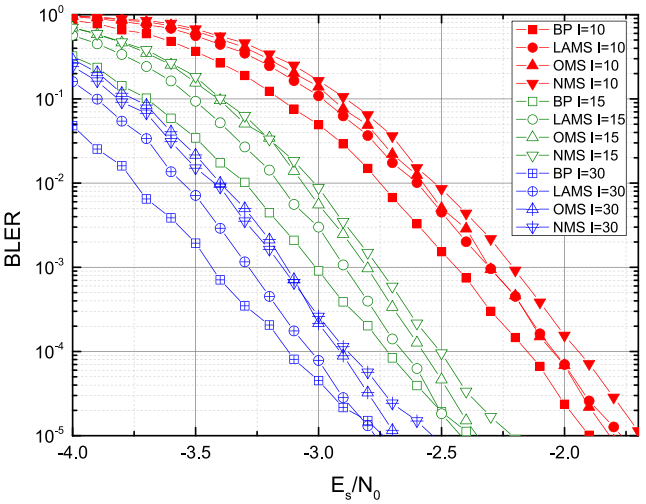

FIGURE 2. The BLER performance of the 5G codes with the information length $K = 520$, the code rate $R = 1/3$, and the different iterations $I = 10, 15, 30$.

where the four factors are listed according to the iteration $i = 0, 1, \cdots, 29$.

We use the BG2 [2] of 3GPP as the parity-check matrix of the decoder. We compare different decoders, the BP, the NMS, and the OMS decoders, to illustrate the advantages of our optimized LAMS decoder. The NMS and the OMS algorithms set the normalized and the offset factors to 0.7 and -0.2, respectively, which are optimized by simulations within 0.1 accuracy range like the factors of the LAMS algorithm. The coded bits are modulated by BPSK and transmitted through the AWGN channel.

In Fig. 2, four decoding schemes, the BP, LAMS, OMS, and NMS algorithms, with the different iterations $I = 10, 15, 30$ are simulated. The decoding performance gaps between the LAMS algorithm and the BP algorithm with more iterations are narrower than that with fewer iterations. When the number of the iterations is $I = 10$, the performance of the LAMS algorithm is similar to that of the OMS algorithm and does not show significant advantages. However, for the iteration $I = 15$, our LAMS decoder is obviously better than the OMS and the NMS algorithms. When the SNR is larger than -2.5 (dB), the LAMS algorithm even shows better performance than the BP algorithm, which may result from the better error floor of the LAMS algorithm. A similar situation can be observed when the number of the iterations is 30. The result that the LAMS algorithm performs better in the high SNR region is not surprising, because the BP algorithm is not optimal for short code lengths and limited numbers of iterations, which is caused by the increasing correlations during message passing [4]. Hence, by reducing the negative effect of correlations, the optimized LAMS decoders can outperform the BP decoder in some specific cases.

Fig. 3 shows the performance of the rate-1/3 codes with different information lengths for the four decoders, where the maximum number of the iterations is set to $I = 15$. As can be

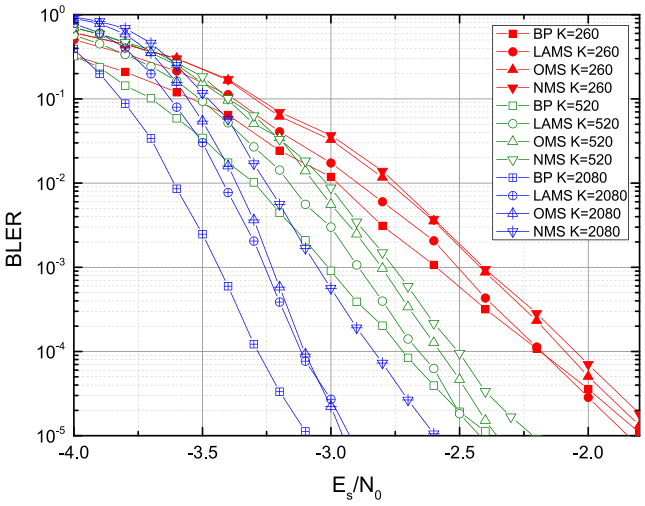

FIGURE 3. The BLER performance of the 5G codes with the code rate $R = 1/3$, the number of iterations $I = 15$, and the different information lengths $K = 260, 520, 2080$.

seen from the figure, our LAMS algorithm performs better at short information lengths. For the information lengths $K = 260$ and $K = 520$, the LAMS decoders even exceed the BP decoders, when the SNRs are larger than about -2.2 (dB) and -2.5 (dB), respectively.

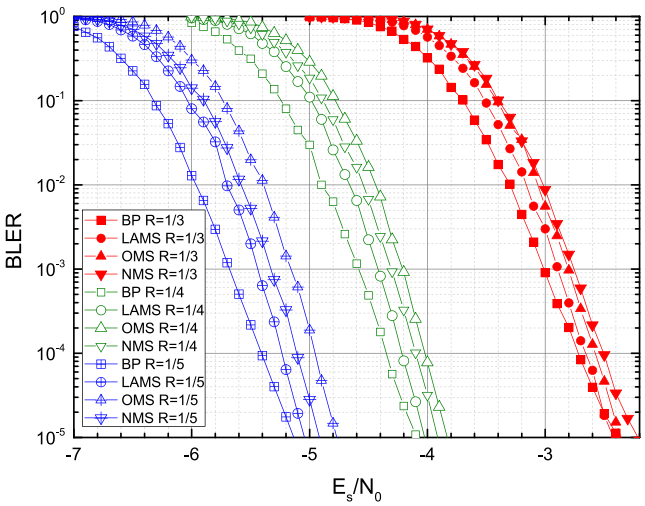

FIGURE 4. The BLER performance of the 5G codes with the information length $K = 520$, the number of iterations $I = 15$, and the different code rates $R = 1/3, 1/4, 1/5$.

As shown in Fig. 4, we also compare the four algorithms for the LDPC codes with different code rates. Although the protographs of the LDPC codes with lower rates $1/4$ and $1/5$ are quite different for the rate $1/3$ code, the LAMS decoding uses the same optimized factors in each iteration shown in Table 1. For these three code rates, the LAMS algorithm can achieve better performance than the NMS and OMS algorithm. The higher the code rate, the more the advantage of the performance for the LAMS decoder. As the SNR increases, the gap between the LAMS decoder and the BP decoder is gradually narrowing. Compared with the OMS algorithm, the LAMS algorithm has better robustness to maintain excellent performance under the same optimized factors for different code rates.

V. CONCLUSION

Inspired by the simplified BP-based algorithms, we propose the LAMS algorithm, which uses a linear approximation instead of a single normalization or offset to modify the check node updating. Moreover, we also modify the LLR calculation from the AWGN channel with the linear approximation. We use a small and shallow neural network to train the corresponding factors of each iteration, where the factors are set to the same values for all the message passing edges in the decoder. Using QC structures, we make the size of the neural network comparable to the size of the protograph. Since we only optimize the parameters of one iteration at a time, the neural network has only three layers, which allows the training to be done without requiring extensive computing resources. Finally, we train a set of factors for 30 iterations and conduct simulations for the LAMS decoder. The simulation results show that the proposed LAMS decoder can perform better than the NMS and OMS decoders and even at high SNR than the BP decoder.

REFERENCES

[1] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 876–888, Aug. 2009.

[2] *WF on LDPC Parity Check Matrices*, document R1-1711982, 3GPP TSG RAN WG1 NR AH #2, Qingdao, China, Jun. 2017. [Online]. Available: http://www.3gpp.org/ftp/TSG_RAN/WG1_RL1/TSGR1_AH/NR_AH_1706/Docs/R1-1711982.zip

[3] T. Y. Chen, K. Vakilinia, D. Divsalar, and R. D. Wesel, "Protograph-based raptor-like LDPC codes," *IEEE Trans. Commun.*, vol. 63, no. 5, pp. 1522–1532, May 2015.

[4] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[5] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[6] S. Myung, S.-I. Park, K.-J. Kim, J.-Y. Lee, S. Kwon, and J. Kim, "Offset and normalized min-sum algorithms for ATSC 3.0 LDPC decoder," *IEEE Trans. Broadcast.*, vol. 63, no. 4, pp. 734–739, Dec. 2017.

[7] J. Chen and M. P. C. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Commun. Lett.*, vol. 6, no. 5, pp. 208–210, May 2002.

[8] J. Zhang, M. Fossorier, D. Gu, and J. Zhang, "Two-dimensional correction for min-sum decoding of irregular LDPC codes," *IEEE Commun. Lett.*, vol. 10, no. 3, pp. 180–182, Mar. 2006.

[9] M. Jiang, C. Zhao, L. Zhang, and E. Xu, "Adaptive offset min-sum algorithm for low-density parity check codes," *IEEE Commun. Lett.*, vol. 10, no. 6, pp. 483–485, Jun. 2006.

[10] Y. Xu *et al.*, "Variable LLR scaling in min-sum decoding for irregular LDPC codes," *IEEE Trans. Broadcast.*, vol. 60, no. 4, pp. 606–613, Dec. 2014.

[11] F. Alberge, "Min-Sum decoding of irregular LDPC codes with adaptive scaling based on mutual information," in *Proc. Int. Symp. Turbo Codes Iterative Inf. Process.*, Sep. 2016, pp. 71–75.

[12] D. Oh and K. K. Parhi, "Min-sum decoder architectures with reduced word length for LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 1, pp. 105–115, Jan. 2010.

[13] X. Wu, Y. Song, M. Jiang, and C. Zhao, "Adaptive-normalized/offset min-sum algorithm," *IEEE Commun. Lett.*, vol. 14, no. 7, pp. 667–669, Jul. 2010.

[14] E. Nachmani and Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2016, pp. 341–346.
[15] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2017, pp. 1361–1365.
[16] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119–131, Feb. 2018.
[17] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning-based channel decoding," in *Proc. 51st Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2017, pp. 1–6.
[18] J. Lewandowsky and G. Bauch, "Information-optimum LDPC decoders based on the information bottleneck method," *IEEE Access*, vol. 6, pp. 4054–4071, 2018.
[19] T. Wadayama and S. Takabe. (2018). "Joint quantizer optimization based on neural quantizer for sum-product decoder." [Online]. Available: https://arxiv.org/abs/1804.06002
[20] A. Bennatan, Y. Choukroun, and P. Kisilev. (2018). "Deep learning for decoding of linear codes—A syndrome-based approach." [Online]. Available: https://arxiv.org/abs/1802.04741
[21] F. Liang, C. Shen, and F. Wu, "An iterative BP-CNN architecture for channel decoding," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 144–159, Feb. 2018.

XIAONING WU (S'15) received the B.Eng. degree in communication and information engineering from Southeast University, Nanjing, China, in 2014, where he is currently pursuing the Ph.D. degree in communication and information engineering. His research interests include code design, encoding/decoding, and machine learning.

MING JIANG (S'06–M'07) received the B.Sc., M.S., and Ph.D. degrees in communication and information engineering from Southeast University, Nanjing, China, in 1998, 2003, and 2007, respectively.

He is currently an Associate Professor with the National Mobile Communications Research Laboratory, Southeast University. His research interests include coding and modulation technology.

CHUNMING ZHAO (M'93) received the B.Sc. and M.S. degrees from the Nanjing Institute of Posts and Telecommunications in 1982 and 1984, respectively, and the Ph.D. degree from the Department of Electrical and Electronic Engineering, University of Kaiserslautern, Germany, in 1993.

He has been a Post-Doctoral Researcher with the National Mobile Communications Research Laboratory, Southeast University, where he is currently a Professor and the Vice Director of the Laboratory. He has managed several key projects of Chinese Communications High-Tech Program. His research interests include communication theory, coding/decoding, mobile communications, and VLSI design.

Dr. Zhao received the First Prize of National Technique Invention of China in 2011. He was a recipient of the Excellent Researcher Award from the Ministry of Science and Technology, China.