Article

Minimum Distance Optimization with Chord Edge Growth for High Girth Non-Binary LDPC Codes

Changcai Han , Hui Li and Weigang Chen *

School of Microelectronics, Tianjin University, Tianjin 300072, China; cchan@tju.edu.cn (C.H.);
hui_li@tju.edu.cn (H.L.)

* Correspondence: chenwg@tju.edu.cn; Tel.: +86-186-2264-0107

Received: 11 November 2020; Accepted: 12 December 2020; Published: 17 December 2020

Abstract: Short or moderate-length non-binary low-density parity-check (NB-LDPC) codes have the potential applications in future low latency and high-reliability communication thanks to the strong error correction capability and parallel decoding. Because of the existence of the error floor, the NB-LDPC codes usually cannot satisfy very low bit error rate (BER) requirements. In this paper, a low-complexity method is proposed for optimizing the minimum distance of the NB-LDPC code in a progressive chord edge growth manner. Specifically, each chord edge connecting two non-adjacent vertices is added to the Hamiltonian cycle one-by-one. For each newly added chord edge, the configuration of non-zero entries corresponding to the chord edge is determined according to the so-called full rank condition (FRC) of all cycles that are related to the chord edge in the obtained subgraph. With minor modifications to the designed method, it can be used to construct the NB-LDPC codes with an efficient encoding structure. The analysis results show that the method for designing NB-LDPC codes while using progressive chord edge growth has lower complexity than traditional methods. The simulation results show that the proposed method can effectively improve the performance of the NB-LDPC code in the high signal-to-noise ratio (SNR) region. While using the proposed scheme, an NB-LDPC code with a quite low BER can be constructed with extremely low complexity.

Keywords: Hamiltonian graph; cycle; full rank condition; non-binary low-density parity-check codes

1. Introduction

The design of efficient short codes has attracted widespread interest due to the emergence of applications requiring high reliability and low latency data transmission [1,2]. The low-density parity check (LDPC) codes, which were first proposed by Gallager et al. [3] and rediscovered by MacKay and Neal [4,5], have performance close to the Shannon limit as the code length approaches infinity [6]. For short or moderate code lengths, non-binary low-density parity-check (NB-LDPC) codes exhibit better error-correction capabilities than binary LDPC codes in certain cases [7,8]. The construction of NB-LDPC codes can adopt algebraic or graph theory methods. The algebraic methods for constructing non-binary quasi-cyclic LDPC codes that are based on finite Euclidean geometric planes and array masks were studied by Zhou et al. [9], Huang et al. [10] and Peng et al. [11]. The codes that were constructed by these methods can be effectively encoded and they have excellent performance on the Additive White Gaussian Noise (AWGN) channel. The design methods of non-binary protograph-based LDPC codes with excellent performance were studied by Divsalar et al. [12], Dolecek et al. [13] and Wang et al. [14].

The design of an LDPC code with excellent performance in the high signal-to-noise ratio (SNR) region is challenging due to the presence of the error floor. It has been found that the error floor is largely attributed to the effect of error-prone graph structures in the Tanner graph of the code, such as stopping sets and trapping sets [15–17]. Therefore, much work has been devoted to the

construction of NB-LDPC codes that can avoid these harmful substructures [14–16,18,19]. In order to avoid small absorbing sets, a design guide of NB-LDPC codes with the column weight greater than two was proposed by Amiri et al. [16], which can significantly reduce the error floor of the graph-based non-binary codes. Because these small substructures are related to short cycles, the error floor can be mitigated by manipulating the parameters of the cycles [14,15]. The designing scheme of the NB-LDPC codes by maximizing the quasi girth and the minimum distance of the subgraph was studied by Wang et al. [14]. However, this method needs to continuously expand the tree to determine the local girth and calculate the minimum distance upper limit, which increases the complexity of the algorithm.

The ultra-sparse NB-LDPC codes with the column weight of two defined on the high-order finite field or general linear group have excellent performance at finite length [7,8,15,20]. For the optimization of this type of ultra-sparse NB-LDPC codes, a method that is based on the algebraic properties of the binary image of the parity-check matrix that is associated with the code to ensure the performance of the waterfall region and the error floor region was introduced by Poulliat et al. [15], where the algebraic properties of the cycles and the stopping sets were linked to the global minimum distance of the code. However, it does not take advantage of the characteristics of ultra-sparse codes to simplify the optimization method for the numerous cycles in the corresponding graph. Because ultra-sparse codes can be described by simple graphs [21], the sub-optimal optimization methods combining the characteristics of graphs can be used to design the NB-LDPC codes with the outstanding performance efficiently. Based on the studies by Chen et al. [22,23,24], the design method can be further modified to obtain the NB-LDPC code that can be encoded by a simple recursive method, thereby achieving efficient encoding.

In this paper, a progressive chord edge growth strategy (PCEG) for ultra-sparse NB-LDPC codes with high girth, which can effectively lower the error floor of NB-LDPC codes, is proposed. The chord edges are added to the Hamiltonian cycle in turn, and the candidate non-zero entries corresponding to each chord edge are searched. The selection criterion is to ensure that all of the cycles in the current subgraph satisfy the full rank condition (FRC) [15]. The proposed method has the advantage of low complexity. For one thing, the number of cycles to be subjected to the FRC check in the subgraph is effectively reduced, benefiting from the application of the progressive chord edge growth. For another thing, the number of candidate configurations of non-zero entries does not increase exponentially with the increase of the chord edges, due to the FRC constraint. Through such a concise scheme for allocating non-zero elements, a large number of short cycles in the graph that is associated with the code can be cancelled, thereby obtaining an NB-LDPC code with a low error floor. Based on the proposed method, a flexible modification is further developed to construct NB-LDPC codes with an efficient encoding structure. For the efficient encoding structure, good error correction performance can still be provided while using the proposed construction method. Therefore, the contribution of this paper is to develop a construction method for the NB-LDPC codes with a column weight of 2, which can obtain comparable performance and it has a lower complexity than the scheme proposed by Poulliat et al. [15].

The rest of this paper is organized, as follows. Section 2 gives a brief review of the Hamiltonian representation of NB-LDPC codes and the FRC of cycles and analyzes the method for improving the performance in the error floor region. In Section 3, a method for progressive chord edge growth that was developed for optimizing the minimum distance is presented, and the proposed method is slightly modified to construct easy-to-encode NB-LDPC codes. In Section 4, the complexity analysis of the proposed method and the simulation results of several design examples are provided to verify the proposed design method. Finally, Section 5 provides the conclusions.

2. Non-Binary LDPC Codes Based on High Girth Hamiltonian Graphs

In this section, the Hamiltonian representation of NB-LDPC codes, the FRC of cycles, and a conventional optimization method of NB-LDPC codes are reviewed. First, a class of NB-LDPC codes

with variable nodes of the degree of two and the Hamiltonian graph representation of these codes are described. Subsequently, the cycles and the FRC of the cycles are introduced, which are the foundations of the proposed design method. Finally, the method proposed by Poulliat et al. [15] that improves the error floor while using the algebraic properties of cycles and stopping sets is briefly presented.

2.1. Parity-Check Matrix and Hamiltonian Graph

The NB-LDPC code that is defined on the high-order Galois field with a sparse parity-check matrix is generally described by a Tanner graph. Furthermore, an NB-LDPC code with a variable node degree of two can be represented using a simple graph obtained by omitting the variable nodes in its Tanner graph. Consider a graph $G(V, E)$ with neither loops nor multiple edges, where V represents a non-empty set of vertices and E represents a set of edges. The Hamiltonian cycle in graph G is a cycle that passes once through all of the vertices in graph G , and the Hamiltonian graph denotes a graph with at least one Hamiltonian cycle. The Hamiltonian graph forms of an NB-LDPC code are presented in Figure 1a. For a three-regular graph with eight vertices, the original vertex is taken as a check node, a vertex is inserted as a variable node on each edge of the graph, and the non-zero elements of the Galois field $GF(q)$, where q is the order of the field, are added to the edges connecting these two types of nodes. Subsequently, the parity-check matrix and Tanner graph of the NB-LDPC code that is shown in Figure 1b,c, respectively, can be obtained. If the Hamiltonian graph has the regular vertex degree of d_c , then a $(2, d_c)$ -regular LDPC code can be constructed with the variable node degree of 2 and check node degree of d_c in the associated Tanner graph.

Figure 1. (a) The mapping from an eight-vertex Hamiltonian graph to an non-binary low-density parity-check (NB-LDPC) code, where $v_1, v_2, \dots, v_8$ correspond to the check node, and $\alpha_{11}, \alpha_{12}, \dots, \alpha_{83}$ are non-zero elements over the Galois field. (b) The parity-check matrix of the code associated with the Hamiltonian graph. (c) The Tanner graph of the code associated with the Hamiltonian graph.

In Figure 1, the representation of the cycle is given. In the Tanner graph, a cycle represents a closed-loop that consists of different vertices and edges, and the cycle length is the number of edges or vertices that it contains. The girth g denotes the length of the shortest cycle in the entire graph. The vertices in the Hamiltonian graph correspond to check nodes, and the actual length of the cycle is twice the number of passed vertices. The shortest cycle length in the Hamiltonian graph that is shown in Figure 1 is four, and the corresponding Tanner graph has a girth of eight. Section 2.2 describes the influence of girth and cycles on the performance of the code in detail.

The NB-LDPC code that is constructed based on the Hamiltonian graph can not only obtain good girth characteristics, but also be implemented with the linear encoding. Many efficient encoding structures have been developed while using the recursive encoding methods that are suitable for particular graphs [22–24]. In these methods, the matrix mapped from the Hamiltonian cycle can be modified, as shown in Equation (1).

$$\mathbf{H}_P = \begin{bmatrix} \alpha^0 & 0 & \cdots & \cdots & 0 \\ \alpha^{k_{1,0}} & \alpha^0 & & & \vdots \\ 0 & \alpha^{k_{2,0}} & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha^{k_{m-1,0}} & \alpha^0 \end{bmatrix}. \quad (1)$$

Let $[\mathbf{s}, \mathbf{p}]$ be a legal codeword, where $\mathbf{s} = [s_0, s_1, \dots, s_{n-m-1}]$ and $\mathbf{p} = [p_0, p_1, \dots, p_{m-1}]$ are the information symbol vector and check symbol vector, respectively. The NB-LDPC code with $[\mathbf{H}_I, \mathbf{H}_P]$ as the parity-check matrix can be encoded by the recursive method that includes the following steps:

Step1: $w_i = \alpha^{k_{i,0}}, 1 \leq i \leq m-1$;

Step2: $p_0 = u_0, p_i = u_i - w_i \cdot p_{i-1}, 1 \leq i \leq m-1$.

In the above steps, $\mathbf{u} = [u_0, u_1, \dots, u_{m-1}]$ is the intermediate result, and $\mathbf{u} = \mathbf{H}_P \cdot \mathbf{p}^T = \mathbf{H}_I \cdot \mathbf{s}^T$. The vector $\mathbf{u}$ can be calculated according to the matrix $\mathbf{H}_I$ and the information symbol vector $\mathbf{s}$. The $(m-1)$ non-zero elements $\alpha^{k_{1,0}}, \alpha^{k_{2,0}}, \dots, \alpha^{k_{m-1,0}}$ in the matrix $\mathbf{H}_P$ are stored to calculate each check symbol p_i in the described recursive steps. The method described requires less calculation and storage resources when compared with traditional encoding methods. Therefore, the NB-LDPC code with the parity-check matrix in the form shown in Equation (1) can be efficiently encoded in a recursive manner.

2.2. Cycle and Cycle Cancellation

The cycle in the Tanner graph is related to the sub-optimality of the iterative decoding. The length of the shortest cycle where a node is located represents the shortest path or minimum number of iterations a node's message needs to travel back to the node itself. When the number of iterations increases, the message propagates to other parts of the graph. Therefore, enlarging the girth of an LDPC code helps to improve the iterative decoding performance. Figure 2a shows a Hamiltonian graph that is composed of 26 vertices and 26 chord edges, and it can be used to derive a parity-check matrix with a code rate of 1/2. In this figure, the shortest cycle length is six, i.e., the girth of the derived parity-check matrix is 12.

Although there are no short cycles in the parity-check matrix constructed while using the high-girth Hamiltonian graph, in the case of an improper configuration of non-zero entries, the existence of cycles can generate low-weight codewords, thereby reducing the minimum distance and introducing a higher error floor. Fortunately, it is possible to prevent the cycle from generating low-weight codewords by ensuring that the determinant value of the matrix corresponding to the cycle is not equal to zero, which means that the cycle is cancelled [15].

Figure 2. (a) The Hamiltonian graph with 26 vertices. (b) A cycle in the graph and its full rank condition (FRC).

Let $\alpha_1, \alpha_2, \dots, \alpha_l$ be the non-zero elements of Galois field $\text{GF}(q)$ related to a cycle of length l , and the matrix form of the cycle is equivalent to the $\frac{l}{2} \times \frac{l}{2}$ square matrix that is expressed in Equation (2).

$$\mathbf{C}_d = \begin{bmatrix} \alpha_2 & \alpha_3 & 0 & \cdots & \cdots & 0 \\ 0 & \alpha_4 & \alpha_5 & & & \vdots \\ \vdots & & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & & & & \alpha_{l-2} & \alpha_{l-1} \\ \alpha_1 & 0 & \cdots & \cdots & 0 & \alpha_l \end{bmatrix}. \quad (2)$$

The determinant of $\mathbf{C}_d$ is given by

$$\det(\mathbf{C}_d) = \prod_{i=1}^{l/2} \alpha_{2i} - \prod_{i=1}^{l/2} \alpha_{2i-1} \quad (3)$$

The multiplication and addition operations are all defined on Galois field $\text{GF}(q)$. Subsequently, the FRC of the cycle can be expressed as

$$\prod_{i=1}^{l/2} \alpha_{2i} - \prod_{i=1}^{l/2} \alpha_{2i-1} \neq 0. \quad (4)$$

Furthermore, by applying the addition property of the finite field, the FRC is equivalent to

$$\prod_{i=1}^{l/2} \alpha_{2i} \neq \prod_{i=1}^{l/2} \alpha_{2i-1}. \quad (5)$$

Figure 2b shows the FRC of a cycle of length 12. Therefore, a cycle can avoid the generation of low-weight codewords by appropriately configuring its corresponding non-zero elements to satisfy the FRC.

2.3. Optimization Method of Minimum Distance Based on the Binary Image

A method of selecting and configuring non-zero elements that is based on binary images for optimizing NB-LDPC codes with the column weight of two was proposed by Poulliat et al. [15]. Based on the minimum distance of the equivalent binary code, the non-zero entries in each row were carefully selected, which proved to improve the performance in the waterfall region. Furthermore,

the structure of the relevant Tanner graph was considered to lower the error floor. For the parity-check matrix of a code, the columns defining the minimum distance are located on symbols belonging to the shortest cycles and the associated stopping sets. Therefore, a method using the algebraic properties of cycles and stopping sets was given for improving the minimum distance of the binary image of the code.

Specifically, an iterative and row-wise optimization procedure was performed for the purpose of the cycle cancellation and the stopping set mitigation. Assume that R and S denote the set of optimized non-zero entries for a row and the set of the smallest stopping sets with good minimum distance, respectively. The length of the cycle was assumed to lie in the range from the girth g to a given length l_{max} . The cycle optimization process is as follows.

Initialization: randomly select elements from set R and its permutations as the rows of the check matrix, and define A as the row label set of the parity-check matrix H to be optimized.

Step 1: initialize A as all row labels of matrix H .

Step 2: randomly select a row label a in A .

Step 3: select the element from R and its permutation for a th row that maximizes the number of cancelled cycles of length l , where l indicates the length of the shortest cycles that have not been cancelled.

Step 4: remove a from A . If all cycles of length l satisfy the FRC, then $l = l + 2$, and go to Step 1; otherwise, judge whether there are elements in set A ; and, if there are some of them, go to Step 1; otherwise, go to Step 2.

End of cycle optimization: when no more cycles can be cancelled, the cycle optimization process ends.

After the cycle cancellation process, the process that is described above was used to optimize the cycles with length greater than l_{max} continuously from the perspective of the stopping set. In order to optimize the stopping set, the selection criterion of non-zero elements for a row was to maximize the number of cancelled loops and the minimum distance of the stopping set.

In the cycle optimization method that was proposed by Poulliat et al. [15], a row of non-zero elements are temporarily determined based on the number of cycles associated with the row that do not meet the FRC. The idea of iterative optimization can cancel cycles to a large extent. However, all cycles related to the selected row need to perform the FRC check, which means that a large number of cycles should be checked repeatedly, increasing the complexity. According to the characteristics of ultra-sparse graphs, a method for configuring non-zero entries with the progressive chord edge growth is proposed in this paper, which can effectively reduce the number of FRC checks.

3. Minimum Distance Optimization Using Progressive Chord Edge Growth

Based on the high-girth graph [22–24] and a set of optimized non-zero entries that are associated with a row [15], a method of configuring non-zero entries in the manner of progressive chord edge growth is proposed to increase the minimum distance of NB-LDPC codes. The non-zero elements are selected for each chord edge based on the FRC of the cycles. Only those cycles containing the newly-added chord edges are checked, which effectively reduces the number of cycles that need to perform a FRC check. Furthermore, a simple variant of the proposed method is applied to optimize the NB-LDPC codes with an efficient encoding structure.

3.1. Proposed Progressive Chord Edge Growth (PCEG) Method

Assume that, in the Hamiltonian graph G that is composed of m vertices and n edges, the associated parity-check matrix is expressed as $H = [H_I, H_P]$. Let the sets of vertices and edges composing the Hamiltonian graph be $V = \{v_1, v_2, \dots, v_m\}$ and $E = \{e_1, e_2, \dots, e_n\}$ respectively, where $v_i (i = 1, 2, \dots, m)$ and $e_j (j = 1, 2, \dots, n)$ correspond to the i th row and j th column in matrix H , respectively. The first m edges in E , which construct set E_C , form a Hamiltonian cycle with a length of $2 \times m$. The remaining edges in E denote chord edges that connect two non-adjacent vertices

in the Hamiltonian graph in turn. The process of progressive chord edge growth of a 26-vertex Hamiltonian graph is illustrated in Figure 3, where $M_a^{27}, M_a^{28}, \dots, M_a^{52}$ represent the total number of cycles, and $M_c^{27}, M_c^{28}, \dots, M_c^{52}$ represent the number of the cycles associated with the newly added chord edge.

Figure 3. The process of progressive chord edge growth and related cycles. $M_a^{27}, M_a^{28}, \dots, M_a^{52}$ represent the total number of cycles, and $M_c^{27}, M_c^{28}, \dots, M_c^{52}$ represent the number of the cycles that are introduced by the newly added chord edge.

Although the cycles included in the Hamiltonian graph are complex globally, the cycles related to the newly added chord edge are limited from the perspective of the progressive chord edge growth. As mentioned before, this paper considers cycles with a length l having the value from girth g to a given length $l_{\max}$. In the process of progressive chord edge growth, the number of cycles that do not satisfy the FRC in the cycles introduced by the newly added chord edge is denoted as $N_j^{l \leq l_{\max}}$. The proposed non-zero element configuration method for making the cycles meet the FRC will be described in detail in the following.

According to the order of growing chord edges, the allocation of non-zero elements is divided into two main steps. One is the selection of non-zero elements in set E_C , and the other is the selection of non-zero elements that correspond to each chord edge. Figure 4 shows the specific process of configuring non-zero entries with progressive chord edge growth.

During the initialization process, the non-zero elements on the Hamiltonian cycle are determined. Because there is no short cycle in E_C , the non-zero elements chosen from set R of optimal row entries [15] can be randomly configured. Specifically, two randomly selected elements from R are assigned to two non-zero positions that correspond to vertex v_i on the Hamiltonian cycle. The set of unselected non-zero elements corresponding to vertex v_i is denoted as $R(v_i)$, and the number of elements in $R(v_i)$ is denoted as $n(v_i)$.

After selecting the corresponding non-zero elements on the Hamiltonian cycle, the non-zero elements of each chord edge are determined, in turn, according to the remaining optional non-zero elements set $R(v_i)$ for each vertex. Starting from edge e_{m+1} , the two vertices that are connected by edge e_{m+1} are denoted as $v_{e_{m+1}}^{(1)}$ and $v_{e_{m+1}}^{(2)}$, where $v_{e_{m+1}}^{(1)}$ is the vertex where edge e_m and edge e_{m+1} intersect, and $v_{e_{m+1}}^{(2)}$ represents the vertex where edge e_{m+1} and edge e_{m+2} intersect. First, all combinations of the optional non-zero elements of chord edge e_{m+1} are recorded by traversing the elements in $R(v_{e_{m+1}}^{(1)})$ and $R(v_{e_{m+1}}^{(2)})$. Subsequently, the number $N_{m+1}^{l \leq l_{\max}}$ of cycles not satisfying the FRC among all of the cycles related to edge e_{m+1} are determined. Finally, the configurations that satisfy $N_{m+1}^{l \leq l_{\max}} = 0$ are retained, and the configuration of non-zero entries that correspond to the next chord edge e_{m+2} is performed. This process continues until the configuration of the non-zero entries of the last chord edge is completed.

Based on the Hamiltonian graph consisting of m vertices and n edges, the design process of the NB-LDPC codes in the proposed progressive chord edge growth manner is summarized, as follows.

Initialization: randomly select non-zero elements from the optimized row set R , assign them to the non-zero positions corresponding to the Hamiltonian cycle, and initialize the index of the chord edge $j = m + 1$.

Step 1: add the chord edge e_j to the Hamiltonian cycle, and obtain the subgraph G_j composed of the Hamiltonian cycle and the added $(j - m)$ chord edges.

Step 2: according to the remaining optional non-zero elements of the vertices connected to the chord edge e_j , select the non-zero elements corresponding to the chord edge e_j , in order that the cycles with the length not greater than l_{max} in the subgraph G_j related to the chord edge e_j all satisfy the FRC.

Step 3: if there is no non-zero element that meets the condition, go to the initialization process; otherwise, judge whether the chord edge index j is equal to n , and if $j = n$, go to Step 4; otherwise, $j = j + 1$, and go to Step 1;

Step 4: from all of the non-zero element configurations that meet the conditions, select the configuration that violates less FRC criterion, and finally obtain the optimized parity-check matrix of the NB-LDPC code.

Figure 4. The scheme of configuring non-zero entries based on the progressive chord edge growth.

In the configuration of the non-zero entries for each chord edge, there are some subtle points that need further explanation. First, if there is no configuration that meets $N_j^{l \leq l_{max}} = 0$ after traversing all optional non-zero elements of edge e_j , the non-zero elements that correspond to E_C are re-allocated and the non-zero elements are re-configured from the first chord edge. Second, if there are multiple choices for configuring the non-zero entries corresponding to the last chord edge, that is, multiple parity-check matrices satisfying $N_n^{l \leq l_{max}} = 0$ are obtained, then the FRC check will be performed on the cycles with a cycle length that is greater than l_{max} , and the matrix that has the minimum number of cycles not satisfying the FRC is selected. Third, if there are a large number of candidate configurations of non-zero entries for a chord edge, then one candidate configuration is randomly reserved, which is, the non-zero elements that are related to the added chord edges are fixed. This effectively prevents excessive memory usage due to the rapid increase in the number of optional configurations.

3.2. Progressive Chord Edge Growth Optimization for Efficient Encoding

The proposed method for configuring non-zero entries in a progressive chord edge growth manner can be used to generate NB-LDPC cycle codes with an arbitrary size, as long as a Hamiltonian graph is provided. After slight modification, the proposed method can also be applied to construct NB-LDPC codes with linear-time-encoding.

Based on the study by Chen et al. [24], a parity-check matrix with a bidiagonal structure in the last m columns and α^0 in the principal diagonal can be efficiently encoded. Therefore, the proposed method can be used to construct NB-LDPC codes with low encoding complexity after two minor modifications. The first one is to modify the Hamiltonian graph, so that it can be mapped to a matrix with a bidiagonal structure, which is, the Hamiltonian graph after disconnecting the first vertex and edge e_m is mapped to be the parity-check base matrix. The second is to modify the non-zero element configuration corresponding to the Hamiltonian cycle. When selecting non-zero elements of the Hamiltonian cycle, the non-zero entry corresponding to edge e_j and vertex v_j is fixed as α^0 , and the non-zero entries that correspond to edge e_j and vertex v_{j+1} are randomly selected from R . These two modifications are applied to the initialization step in the method described in Section 3.1, thereby enabling the designed code to have an efficient encoding structure.

4. Complexity Analysis and Design Examples

In this section, the method for constructing NB-LDPC codes while using the progressive chord edge growth strategy is comprehensively analyzed. The complexity of the method is first analyzed in terms of the number of cycles and the number of candidate configurations for each chord edge. Subsequently, several examples of NB-LDPC codes constructed using the proposed method are shown. The distribution of cycles that do not meet the FRC is counted, and the performance of the designed codes is evaluated through computer simulation.

4.1. Complexity Analysis

This paper aims to design a method for configuring non-zero entries that are based on the FRC of a cycle, thus optimizing the global minimum distance of the NB-LDPC codes. For finite length codes, optimizing the allocation of non-zero entries in the parity-check matrix is a complicated permutation and combination problem. In addition, the number of cycles in the graph associated with the code is large, even for the ultra-sparse code, which increases the time complexity of the optimization process. In this subsection, the Hamiltonian graph shown in Figure 2 is first taken as an example to analyze the complexity of the proposed method in detail. The number of check nodes and number of chord edges are both 26, and the maximum cycle length considered is 16. Furthermore, when considering the design complexity of high code rate NB-LDPC codes, a more general discussion is provided to evaluate the complexity for designing NB-LDPC codes with any code rate.

First, with the proposed method, the number of candidate non-zero entries for each chord edge is limited. The proposed method traverses the optional non-zero entries of each chord edge in a spanning tree manner and deletes nodes that do not satisfy the conditions. As the chord edge increases, a large number of branches that do not meet the criteria are deleted, thus avoiding the exponential growth of the number of branches. The number of branches is further reduced by applying the idea of grouping truncation. That is, after completing the configuration of non-zero entries for a certain chord edge, one of the candidate configurations of non-zero entries is randomly reserved.

Figure 5 shows the number of candidate configurations for each chord edge in the Hamiltonian graph mentioned above. In the figure, the abscissa indicates the index of the newly added chord edge in the progressive chord edge growth, and the ordinate indicates the number of all candidate configurations of non-zero entries for the newly added chord edge. The Hamiltonian graph used has 26 chord edges and 26 vertices, and the truncation is performed after completing the configuration of non-zero entries for the sixth chord edge, due to the large number of candidate configurations.

For comparison, Figure 5 also shows the number of candidate configurations for each row while using the iterative and row-wise optimization method that was proposed by Poulliat et al. [15]. Because the degree of the check node is four, a set of non-zero entries derives 24 different permutations. In the case of using three different sets of non-zero entries, there are 72 different permutation options for each row. Obviously, the number of considered configurations in the proposed method is much smaller than that by Poulliat et al. [15].

Figure 5. The number of candidate configurations for each chord edge/row.

Second, the proposed method for allocating non-zero entries in the progressive chord edge growth manner can effectively reduce the number of cycles that need FRC check compared with similar optimization methods. Figure 6 shows the number of associated cycles for processing each chord edge. The number of cycles that require FRC check gradually increases to about 140 as the chord edge grows while using the proposed method, while the number of considered cycles for each row using the scheme designed by Poulliat et al. [15] is about 240. Applying the progressive chord edge growth strategy, the FRC check is only performed on the cycles that contain the newly added chord edge in the current sub-graph instead of the cycles containing the selected chord edge in the entire graph. Therefore, the number of associated cycles increases as the chord edge grows. Using the iterative and row-wise method, the cycles containing the selected check node in the entire graph are FRC checked, which results in the uniform distribution of the number of considered cycles for optimizing each row. For the last chord edge, the FRC check is performed on the cycles, including the chord edge in the entire graph, and the number of the cycles is still smaller than the number of cycles that are related to a row. This is due to the fact that the column weight of the LDPC code is generally smaller than the row weight, so the number of cycles that are related to a column is smaller than the number of cycles related to a row. The total number of cycles that need FRC check is shown in the legend part of Figure 6. It can be seen that, when compared with the iterative and row-wise optimization method, the proposed method reduces the number of cycles requiring the FRC check from 6634 to 884.

Above, the complexity of the proposed method is analyzed in terms of the number of candidate configurations for each chord edge and the number of cycles that require the FRC check. Combining these two aspects, the number of FRC checks is counted, as shown in the Figure 7. The number of FRC checks performed while using the proposed method is quite small. For the example given, a total of 477,688 FRC checks are required for completing one time optimization of the entire parity-check matrix using the scheme proposed by Poulliat et al. [15], while only 11,810 FRC checks are required using the proposed progressive chord edge growth method. Therefore, the proposed method has much lower complexity.

Figure 6. The number of cycles that require the FRC check for each chord edge/row.

Figure 7. The number of FRC checks for each chord edge/row.

Using the above analysis steps, the design complexity of the proposed method for higher code rate NB-LDPC codes is further analyzed. Table 1 shows the complexity of the optimization for the NB-LDPC code with a code rate of 3/4 and code length of 96 symbols. The row weight d_c and column weight d_v of the adopted base matrix are eight and two, respectively. First, using the method proposed by Poulliat et al. [15], the number of candidate configurations for non-zero elements in each row is the number of permutations of d_c elements. For the case of $d_c = 8$, there are 40,320 permutations. The number of rows of the base matrix is 24, and the number of permutations for all rows is $40,320 \times 24$, which is, 967,680. The non-zero elements of 72 chord edges need to be configured while using the proposed scheme; however, the total number of candidate configurations is less than the iterative and row-wise method. Subsequently, the number of cycles that need to perform FRC check is compared. In the manner of progressive chord edge growth, the cycles for each added chord edge are different. Using the row-wise optimization, there are a large number of cycles that are repeatedly processed when considering that the cycles are in the entire graph. Therefore, fewer cycles need to be checked

using the proposed method. Finally, Table 1 lists the number of FRC checks. The number of FRC checks for each row or each chord edge is the result of multiplying the number of cycles to be processed and the number of candidate configurations for each row or each chord edge. It can be seen that more FRC checks are required for the case of a larger check node degree, and the number of FRC checks using the proposed method is lower than that of the iterative and row-wise optimization.

Table 1. The complexity of the optimization for the NB-LDPC code with the code rate of 3/4.

Methods	The Total Number of Candidate Configurations	The Total Number of Related Cycles	The Total Number of FRC Checks
Progressive chord edge growth	676,598	2330	$\approx 1.32 \times 10^7$
Iterative and row-wise optimization	967,680	11,188	$\approx 4.51 \times 10^8$

In general, for the design of NB-LDPC codes with arbitrary code length and code rate, it is difficult to accurately calculate the number of candidate configurations for non-zero elements in each row or chord edge without specific operations. Nevertheless, the application of truncation can prevent a sharp increase in the number of candidate configurations. As far as the cycles are concerned, the number of related cycles while using the proposed scheme is less than that of the iterative and row-wise optimization, due to the idea of progressive chord edge growth avoiding the repeated processing of cycles. Therefore, the proposed method has lower complexity, even if the number of chord edges is larger than that of rows in the case of high code rate.

4.2. Design Examples

Several examples of finite-length NB-LDPC codes constructed using the proposed method are presented. All of the codewords are simulated on a binary input additive white Gaussian noise (AWGN) channel while using binary phase shift keying (BPSK) modulation, and decoded by the fast Fourier transform belief propagation (FFT-BP) algorithm [6]. The maximum number of iterations is set to be 50. In addition, all of the simulations terminated with 50 erroneous codewords (frames).

Example 1. Based on the Hamiltonian graph with 26 vertices that are shown in Figure 2, the NB-LDPC codes with a code length of 312 bits and a code rate of 1/2 defined on $GF(64)$ are constructed. The local minimum Hamming distance that is associated with a check node is used as a criterion to determine the set of non-zero entries for each row. One set of non-zero entries over $GF(64)$ provided by Poulliat et al. [15] is applied to the proposed method. The non-zero entry set $R = \{\alpha^{37}, \alpha^{22}, \alpha^9, \alpha^0\}$, with a minimum distance of 3 and a weight enumerator coefficient of 20, is selected. Subsequently, the proposed method is performed to configure the non-zero entries for the parity-check matrix. The proposed method and a simplified version of the proposed method are, respectively, used to construct NB-LDPC codes. These two versions are called PCEG and S-PCEG, respectively. For the design example using the S-PCEG method, after completing the search for the non-zero entries of the sixth chord edge, one candidate configuration is reserved. In addition, the proposed method is applied to the optimization of the NB-LDPC code with an efficient encoding structure that was proposed by Chen et al. [24]. Based on the same basic graph and non-zero elements, NB-LDPC codes are randomly constructed for comparison.

Table 2 shows the cycle distributions of the constructed codes. The girths of these NB-LDPC codes are all 12, and the numbers of cycles with lengths from 12 to 20 that do not satisfy FRC are listed. Using the proposed method, the cycles less than 20 in length are cancelled, which improves the performance to a certain extent. For cycles with a length of 20, the number of cycles not meeting FRC is further reduced. The similar effect of cycle cancellation is obtained in the parity-check matrix with an efficient encoding structure [24].

Table 2. Cycle distributions of NB-LDPC codes over GF(64) with the length of 312 bits.

Cycle Length		12	16	20
(2,4)-Regular NB-LDPC Codes	Total number of cycles	219	665	5416
	Random	8	20	197
	Poulliat et al. [15]	0	0	102
	PCEG	0	0	94
	S-PCEG	0	0	109
NB-LDPC Codes with an Efficient Encoding Structure	Total number of cycles	192	557	4336
	Random	9	15	84
	PCEG	0	0	68

Figures 8 and 9 shows the performance simulation results of different constructed codes. Because there is no short cycle not satisfying the FRC, the codes constructed by the proposed method have performance advantages in the high signal-to-noise ratio (SNR) region when compared with the codes constructed randomly with the same code parameters. The proposed method has the potential to construct NB-LDPC codes that meet the requirements of low bit error rate (BER) or frame error rate (FER). Applying the proposed method to the NB-LDPC code with an efficient encoding structure proposed by Chen et al. [24], an NB-LDPC code with low encoding complexity and superior performance can be constructed.

Furthermore, the performance of the code constructed while using the proposed method under different decoding iterative number is analyzed, as shown in Figure 10. It can be observed that, when the iteration number is larger than 20, the code can achieve significant error correction performance. Therefore, 20 iterations may be proper for our designed codes and 10 iterations is not so enough. Thus, in this design, we suggest use 20 iterations, which can achieve a relatively good trade-off of performance and latency. When considering the trade-off between decoding complexity (or decoding latency) and error correction performance, 20 iterations are feasible for the proposed codes.

Figure 8. Performance of (2,4)-regular NB-LDPC codes over GF (64) with the code length of 312 bits.

Figure 9. Performance of NB-LDPC codes with an efficient encoding structure.

Figure 10. Performance of the (2, 4)-regular NB-LDPC code under different number of iterations.

Example 2. In order to analyze the feasibility of the proposed method for constructing NB-LDPC codes with different code rates, the NB-LDPC codes with the code length of 576 bits are constructed, and their code rates are 1/2 and 3/4, respectively. The check node degree of the codes with the code rate of 1/2 is four, and the check node degree of codes with the code rate of 3/4 is eight. For codes with the code rate of 1/2, based on the positions of non-zero entries in the parity-check matrix that was provided by Helmling et al. [25], the proposed method is performed to configure a non-zero value for each position. The NB-LDPC code that was provided by Helmling et al. [25], which has been designed using the cycle optimization proposed by Poulliat et al. [15], is compared with the designed code. In addition, two codes with the same parameters are randomly constructed for comparison. For codes with the code rate of 3/4, one set of optimized non-zero element set $\{\alpha^0, \alpha^6, \alpha^{13}, \alpha^{21}, \alpha^{28}, \alpha^{36}, \alpha^{44}, \alpha^{54}\}$ [26] is used for the non-zero entries that are associated with each check node. The non-zero positions of codes with the same code rate are the same, and the non-zero values are different.

The cycle distributions of the NB-LDPC codes with the two code rates are given in Tables 3 and 4. It can be seen that the proposed method cancels all the cycles less than 20 in length for the codes with the code rate of 1/2. The girth of the codes with a code rate of 3/4 is six and the maximum considered cycle length is 12. The proposed method cancels all cycles with the length of less than 12 and greatly reduces the number of cycles with the length of 12 that do not satisfy the FRC criterion. Figures 11 and 12 show the corresponding performance simulation results. It can be observed that the codes with more cancelled cycles can obtain better performance.

Table 3. Cycle distributions of NB-LDPC codes over GF(64) with the code length of 576 bits and the code rate of 1/2.

Cycle Length		6	8	10	12	14	16	18	20
Total Number of Cycles		0	4	24	56	172	476	1192	2820
Number of Cycles Not Satisfying the FRC	Random 1	0	0	0	1	4	9	14	53
	Random 2	0	1	1	3	2	16	17	67
	Helmling et al. [25]	0	0	0	0	6	8	26	47
	PCEG	0	0	0	0	0	0	0	56

Table 4. Cycle distributions of NB-LDPC codes over GF(64) with the code length of 576 bits and the code rate of 3/4.

Cycle Length		6	8	10	12
Total Number of Cycles		24	414	1892	9452
Number of Cycles Not Satisfying the FRC	Random 1	1	10	27	146
	Random 2	1	9	36	166
	Poulliat et al. [15]	0	0	0	107
	PCEG	0	0	0	103

Figure 11. Performance of NB-LDPC codes over GF(64) with the code length of 576 bits and the code rate of 1/2.

Figure 12. Performance of NB-LDPC codes over GF(64) with the code length of 576 bits and the code rate of 3/4.

5. Conclusions

A low-complexity method that is based on progressive chord edge growth is developed for the configuration of non-zero entries, which can construct NB-LDPC codes with a low error floor. All of the candidate configurations of non-zero entries are traversed for each chord edge in turn, and the configurations that make the cycles in the current subgraph meet the FRC are retained. For a chord edge with a large number of eligible configurations, the limitation on the number of these configurations effectively reduces the memory usage. In addition, a slightly modified version of the proposed method is used to construct the NB-LDPC code with an efficient encoding structure. The analysis results show that the proposed method requires fewer FRC checks than the traditional cycle cancellation method, thus achieving lower complexity. Several design examples prove that our method can construct NB-LDPC codes with a large number of cancelled short cycles, thus effectively lowering the error floor. For future investigation, the idea of the proposed progressive chord edge growth may be applied to the optimization of other structures in NB-LDPC codes.

Author Contributions: Conceptualization, C.H., H.L. and W.C.; methodology, C.H. and W.C.; software, H.L.; formal analysis, C.H., H.L. and W.C.; investigation, C.H., H.L. and W.C.; writing—original draft preparation, H.L.; writing—review and editing, C.H. and W.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by National Natural Science Foundation of China (61671324), and the Director's Funding from Pilot National Laboratory for Marine Science and Technology (Qingdao) (QNL201712).

Conflicts of Interest: The authors declare no conflict of interest.

References

- Shirvanimoghaddam, M.; Mohammadi, M.S.; Abbas R.; Minja, A.; Yue, C.; Matuz, B.; Han G.; Lin, Z.; Liu, W.; Li, Y.; et al. Short block-length codes for Ultra-Reliable Low-Latency Communications. *IEEE Commun. Mag.* **2018**, *57*, 130–137. [\[CrossRef\]](#)
- Coskun M.C.; Durisi, G.; Jerkovits, T.; Liva, G.; Ryan, W.; Stein, B.; Steiner, F. Efficient error-correcting codes in the short blocklength regime. *Phys. Commun.* **2019**, *34*, 66–79. [\[CrossRef\]](#)
- Gallager, R.G. Low-density parity-check codes. *IRE Trans. Inf. Theory* **1962**, *8*, 21–28. [\[CrossRef\]](#)
- Mackay, D.J.C.; Neal, R.M. Near Shannon limit performance of low density parity check codes. *Electron. Lett.* **1996**, *32*, 1645–1646. [\[CrossRef\]](#)
- MacKay, D.J.C. Good error-correcting codes based on very sparse matrices. *IEEE Trans. Inf. Theory* **1999**, *45*, 399–431. [\[CrossRef\]](#)
- Richardson, T.J.; Urbanke, R.L. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inf. Theory* **2001**, *47*, 599–618. [\[CrossRef\]](#)

7. Davey M.C.; MacKay, D. Low-density parity-check codes over $GF(q)$. *IEEE Commun. Lett.* **1998**, *2*, 165–167. [[CrossRef](#)]
8. Hu, X.-Y.; Eleftheriou, E. Binary representation of cycle Tanner-graph $GF(2^b)$ codes. In Proceedings of the 2004 IEEE International Conference on Communications, Paris, France, 20–24 June 2004; pp. 528–532.
9. Zhou, B.; Kang, J.; Song, S.; Lin, S.; Abdel-Ghaffar, K.; Xu, M. Construction of non-binary quasi-cyclic LDPC codes by arrays and array dispersions. *IEEE Trans. Commun.* **2009**, *57*, 1652–1662. [[CrossRef](#)]
10. Huang, Q.; Diao, Q.; Lin, S.; Abdel-Ghaffar, K. Cyclic and quasi-cyclic LDPC codes on constrained parity-check matrices and their trapping sets. *IEEE Trans. Inf. Theory* **2012**, *58*, 2648–2671. [[CrossRef](#)]
11. Peng R.-H.; Chen, R.-R. Design of nonbinary quasi-cyclic LDPC cycle codes. In Proceedings of the 2007 IEEE Information Theory Workshop, Lake Tahoe, CA, USA, 2–6 September 2007; pp. 13–18.
12. Divsalar, D.; Dolecek, L. Graph cover ensembles of non-binary protograph LDPC codes. In Proceedings of the 2012 IEEE International Symposium on Information Theory Proceedings (ISIT), Cambridge, MA, USA, 1–6 July 2012; pp. 2526–2530.
13. Dolecek, L.; Divsalar, Y.; Sun, D.; Amiri, B. Non-binary protograph based LDPC codes: Enumerators, analysis, and designs. *IEEE Trans. Inf. Theory* **2014**, *60*, 3913–3941. [[CrossRef](#)]
14. Wang R.; Li Y.; Zhao H. Nonbinary protograph-based LDPC codes based on additive group of finite field. *IEEE Commun. Lett.* **2016**, *20*, 636–639. [[CrossRef](#)]
15. Poulliat, C.; Fossorier, C. M.; Declercq, D. Design of regular $(2, d_c)$ LDPC codes over $GF(q)$ using their binary images. *IEEE Trans. Commun.* **2008**, *56*, 1626–1635. [[CrossRef](#)]
16. Amiri, B.; Kliewer, J.; Dolecek, L. Analysis and enumeration of absorbing sets for non-binary graph-based codes. *IEEE Trans. Commun.* **2014**, *62*, 398–409. [[CrossRef](#)]
17. Hashemi, Y.; Banihashemi, A.H. Characterization and efficient search of non-elementary trapping sets of LDPC codes with applications to stopping sets. *IEEE Trans. Inf. Theory* **2019**, *65*, 1017–1033. [[CrossRef](#)]
18. Hu, X.-Y.; Eleftheriou, E.; Arnold, D.M. Regular and irregular progressive edge growth Tanner graphs. *IEEE Trans. Inf. Theory* **2005**, *51*, 386–398. [[CrossRef](#)]
19. Cho, S.; Cheun, K.; Yang, K. Design of nonbinary LDPC codes based on message-passing algorithms. *IEEE Trans. Commun.* **2018**, *66*, 5028–5040. [[CrossRef](#)]
20. Chen, W.; Poulliat, C.; Declercq, L.; Conde-Canencia, D.; Al-Ghouwayel, A.; Boutillon, E. Non-binary LDPC codes defined over the general linear group: Finite length design and practical implementation issues. In Proceedings of the VTC Spring 2009—IEEE 69th Vehicular Technology Conference, Barcelona, Spain, 26–29 April 2009; pp. 1–5.
21. Malema G.; Liebelt, M. High girth column-weight-two LDPC codes based on distance graphs. *EURASIP J. Wirel. Commun. Netw.* **2007**, *2007*, 48158. [[CrossRef](#)]
22. Chen, W.; Yin, L.; Lu, J. Efficient encoding of cycle codes on graphs with large girths. In Proceedings of the 2008 International Conference on Communications, Circuits and Systems, Xiamen, China, 25–27 May 2008; pp. 11–15.
23. Chen, W. Efficient encoder structures for non-binary cycle codes. In Proceedings of the International Conference on Wireless Communications Networking & Mobile Computing, Chengdu, China, 23–25 September 2010; pp. 1–5.
24. Chen, W.; Liang, C.; Guo, T.; Ding, Y. Encoder implementation with FPGA for non-binary LDPC codes. In Proceedings of the 2012 18th Asia-Pacific Conference on Communications (APCC), Jeju Island, Korea, 15–17 October 2012; pp. 980–984.
25. Helmling, M.; Scholl, S.; Gensheimer, F.; Dietz, T.; Kraft, K.; Ruzika, S.; Wehn, N. Database of Channel Codes and ML Simulation Results. Available online: <https://www.uni-kl.de/channel-codes/channel-codes-database/non-binary-ldpc/> (accessed on 2 March 2020).
26. Boutillon, E. Optimization of non binary parity check coefficients. *IEEE Trans. Inf. Theory* **2019**, *65*, 2092–2100. [[CrossRef](#)]

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).