A Low Complexity Decoding Algorithm for NB-LDPC Codes over Quadratic Extension Fields

V. B. Wijekoon, Emanuele Viterbo, Yi Hong
Monash University, Australia

Abstract—NB-LDPC codes, a class of codes well-known for their exceptional error correcting performance, are not yet used widely in practice due to the high complexity of decoding algorithms. In this paper, we propose a low complexity decoder for these codes by means of a novel graph expansion. We view the finite field over which the code is constructed as the quadratic extension of one of its subfields, and then expand the Tanner graph of the code into a graph over that particular field. Decoding algorithm, which is tailored for this larger graph, presents significant complexity gains while the performance loss is minimal.

Index Terms—NB-LDPC codes, Graph expansion, Iterative decoding

I. INTRODUCTION

Non-binary low-density parity-check (NB-LDPC) codes were first introduced in 1998 [1], and it soon became apparent that, performance-wise, they surpassed even their binary counterparts, which were by then the code of choice in many practical applications. These performance gains are particularly significant for short-to-moderate code lengths, and are practical applications. These performance gains are particularly significant for short-to-moderate code lengths, and are consistent across many different types of channels [1],[2].

Even though performance of NB-LDPC codes are exceptional, high complexity of the decoding algorithms has so far restricted them from being widely used in practice. The best known decoder, Q-ary sum-product algorithm (QSPA), is of complexity order $O(q^2)$ [1], where $q$ is the size of the algebraic structure over which the code is defined. Fast Fourier transform based implementation of QSPA (FFT-QSPA) [3] manages to reduce this to $O(q \log q)$, but that too is quite high for practical applications, particularly since performance gains are more significant for larger $q$ [1],[3]. Over the past decade or so, numerous algorithms have been introduced in the hope of reducing the decoding complexity of NB-LDPC codes, such as extended min-sum (EMS) [4] and min-max [5] algorithms. But the performance-complexity trade-offs offered by these sub-optimal decoders are not very attractive, since the performance loss compared to QSPA is quite significant for high complexity gains.

Although they can be defined over a number of algebraic structures, finite fields are the overwhelmingly popular choice for constructing NB-LDPC codes. One approach in reducing decoding complexity of such codes is to decode them over smaller fields than the ones used for construction. This has been attempted quite a few times with codes over characteristic 2 fields and the base field $F_2$, for example in [6], [7], and [8], with encouraging results. Still, the performance-complexity trade-off offered by these schemes seems to be dependant on the code, and significant performance losses are noticeable in certain cases. Here we propose using an intermediate subfield, instead of the smallest possible one, for decoding. In particular, we focus on NB-LDPC codes constructed over finite fields of type $F_{p^2r}$, which can be viewed as quadratic extensions of $F_{p^r}$. Using the algebraic relations between $F_{p^2r}$ and $F_{p^r}$, we introduce a method to convert the Tanner graph of a code over $F_{p^2r}$ into a larger graph over $F_{p^r}$. This leads to a decoding algorithm over the second graph that offers significant complexity savings, and, as can be seen in the results section, minimal performance losses in comparison with QSPA.

We explain the graph expansion in detail in the next section, and present the decoding scheme in Section III. Section IV presents some simulation results for the new algorithm, while Section V analyzes and compares its complexity with existing algorithms. Section VI concludes the paper.

II. GRAPH EXPANSION

Expanding a graph over $F_{p^2r}$ into one over $F_{p^r}$ is based on certain properties of the subfield $F_{p^r} \subseteq F_{p^2r}$. We go through these in some detail in subsection A, and then present the expansion in subsection B.

A. Preliminaries

Since $r \mid 2r$, it is guaranteed that $F_{p^2r}$ contains $F_{p^r}$ as a subfield [9]. Following lemma presents the configuration of this subfield, specifically the elements of $F_{p^r}$ that form $F_{p^r}$.

Lemma 1. Let $\alpha$ be a primitive of $F_{p^2r}$, and $\beta = \alpha^{p^r+1}$. Then the subfield $F_{p^r}$ is formed by the set of elements generated by $\beta$, $S_\beta$, and the additive identity of $F_{p^2r}$, 0.

Proof: Let $K' = \{F_{p^2r}, \times\}$ and $K = \{F_{p^r}, \times\}$ be the multiplicative groups of the two fields. Note that both $K$ and $K'$ are cyclic, and that $K$ is a subgroup of order $(p^r - 1)$ in $K'$. A cyclic group can contain only one subgroup of a specific order [9]. $\beta = \alpha^{p^r+1}$ generates $(p^r - 1)$ elements in $F_{p^r}$, and the set of those elements, $S_\beta$, should be a subgroup of order $(p^r - 1)$ in $K'$. Thus, we can conclude $K = S_\beta$, which makes $F_{p^r} = S_\beta \cup \{0\}$. 

Now consider the additive group of $F_{p^r}$, $H = \{F_{p^r}, +\}$, which is a subgroup of $H' = \{F_{p^2r}, +\}$. Multiplying all the elements of $H$ by some $\alpha^i \in F_{p^2r}$ would create another subgroup of $H'$, since additive properties of $H$ are not altered. Both subgroups $H$ and $\alpha^i H$ are of order $p^r$. Although this is true for any value of $i = 0, 1, ..., p^2r - 2$, all subgroups $\alpha^i H$ may not be distinct. Lemma 2 gives the number of such distinct subgroups.

Lemma 2. Let $\alpha$ be a primitive of $F_{p^2r}$. Number of unique subgroups of $H' = \{F_{p^2r}, +\}$ of the form $\alpha^i H$, where $H = \{F_{p^r}, +\}$, is $(p^r + 1)$.  

ISIT 2020

Authorized licensed use limited to: Monash University. Downloaded on September 15,2020 at 13:00:24 UTC from IEEE Xplore. Restrictions apply.
Proof: As established in the proof of Lemma 1, $H = S_\beta \cup \{0\}$. Note that $\alpha_i \times 0 = 0$ for any $i$. Powers of $\alpha$ of the elements in $S_\beta$ form a subgroup of the ring of integers modulus $(p^{2r} - 1)$. Let that subgroup be $R_\beta$. Multiplying elements of $H$ by some $\alpha^i$ is akin to adding $i$ to elements of $R_\beta$, and the number of unique subgroups $\alpha^i H$ is equal to the number of unique sets $R_\beta \oplus i$, where $\oplus$ denotes addition modulus $(p^{2r} - 1)$.

If $R_\beta$ is arranged in increasing order, gap between each pair of consecutive elements (and first and last) would be $(p^r + 1)$. Therefore, $R_\beta \oplus (p^r + 1) = R_\beta$. Then, it becomes clear that for any $i, k \in \mathbb{Z}^\times, R_\beta \oplus (i + k(p^r + 1)) = R_\beta \oplus i$, and $R_\beta \oplus i$ can only be unique for $i = 0, 1, ..., p^r$. This means there can only be $(p^r + 1)$ unique subgroups $\alpha^i H$.

We denote this set of subgroups of $H'$ by $T_H = \{\alpha^0 H, \alpha H_1, ..., \alpha^p H\} = \{H_0, H_1, ..., H_{p-1}\}$, which has a unique property that leads to the graph expansion. Multiplying any $H_i \in T_H$ by some power of $\alpha$ results in some $H_{i_2} \in T_H$. Expansion uses the quotient groups of these $H_i$.

Quotient group $Q_i = H'/H_i$ consists of the $p^r$ cosets of $H_i$, and we represent this as $Q_i = \{C_0^i, C_1^i, ..., C_{p^r-1}^i\}$, where $Q_i$ can be thought of as a division, or a mapping, of elements in $H'$ into $p^r$ disjoint sets that preserves the additive property, i.e., a homomorphism from $H'$ to $H_i$. Therefore, each $Q_i$ is isomorphic to $H = \{\mathbb{F}_{p^r}, +\}$.

Denote the set of quotient groups by $T_Q = \{Q_0, Q_1, ..., Q_{p^r}\}$, and define the operation $\alpha^k Q_i$ as multiplying elements of all cosets of $H_i$ by $\alpha^k$, or $\alpha^k Q_i = \{\alpha^k C_0^i, \alpha^k C_1^i, ..., \alpha^k C_{p^r-1}^i\}$. Here, $Q_i$'s in $T_Q$ are related through this multiplication operation in a similar way to $H_i$'s in $T_H$, and this relationship is presented by Lemma 3.

Lemma 3. Let $H_{i_1}, H_{i_2} \in T_H$ be related as $H_{i_1} = \alpha^{i_1-i_2} H_{i_2}$. Then, $\alpha^{i_1-i_2} Q_{i_2}$ is a permutation of $Q_{i_1}$.

Proof: From the definition of operation $\alpha^k Q_i$;

$\alpha^{i_1-i_2} Q_{i_2} = \{\alpha^{i_1-i_2} C_0^i_{i_2}, \alpha^{i_1-i_2} C_1^i_{i_2}, ..., \alpha^{i_1-i_2} C_{p^r-1}^i_{i_2}\}$

We can represent cosets using the subgroup $H_{i_2}$ and coset leader terms, $l_0^i, C_0^i = \{H_{i_2}, l_0^i\}$. $C_0^i$ is the trivial coset, the subgroup itself.

$\alpha^{i_1-i_2} Q_{i_2} = \{\alpha^{i_1-i_2} H_{i_2}, ..., \alpha^{i_1-i_2} (H_{i_2} + l_0^i)\}$

Since $Q_{i_2}$ contains the cosets of $H_{i_2}$, which are disjoint, all $\alpha^{i_1-i_2} (H_{i_2} + l_0^i)$ should be disjoint as well. Using the relation $H_{i_1} = \alpha^{i_1-i_2} H_{i_2}$ we obtain;

$\alpha^{i_1-i_2} Q_{i_2} = \{H_{i_1}, H_{i_1} + \alpha^{i_1-i_2} l_0^i, ..., H_{i_1} + \alpha^{i_1-i_2} l_{p^r-1}^i\}$

Clearly, $\alpha^{i_1-i_2} Q_{i_2}$ contains the subgroup $H_{i_1}$ and its proper cosets. Thus, it contains the same elements as $Q_{i_1}$. If we consider the quotient groups in some specific order, it is possible that $\alpha^{i_1-i_2} Q_{i_2}$ and $Q_{i_1}$ are ordered differently, although the trivial coset, $H_{i_1}$, should be in the same position. Thus, $\alpha^{i_1-i_2} Q_{i_2}$ is a permutation of $Q_{i_1}$.

Our proposal is to construct probability vectors for each quotient group in $T_Q$, and use those instead of the symbol probability vector for decoding. In order to form $i$'th such probability vector for some variable, probabilities of that variable belonging to each coset in $Q_i$ have to be computed, and therefore, we call these ‘coset probability vectors’. With this approach, it is possible to significantly reduce the complexity of check node operations in decoding NB-LDPC codes.

Traditionally, check node operations are viewed as a combination of two distinct steps: permutation, and convolution of symbol probability vectors. Permutation step is necessary since a symbol turns into another when multiplied with some edge weight in the non-binary Tanner graph. From Lemma 3, it is clear quotient groups in $T_Q$ behave similarly; $Q_{i_1} \in T_Q$ turns into a permutation of $Q_{i_2} \in T_Q$ when multiplied with some edge weight. Thus, coset probability vectors also have to be permuted, and there is no significant change to the permutation step.

Main reason for the high decoding complexity of NB-LDPC codes is the convolution step [4], [5]. In order to understand how the proposed approach changes this step, we consider the simple case of a degree 3 check node of a code over $\mathbb{F}_{p^r}$, where the parity-check equation is $v_1 + v_2 + v_3 = 0$. Assume we need to calculate $i$'th coset probability vector of $v_1, P_i v_1$. Note that this calculation is not directly impacted by any coset probability vector other than the $i$'th ones of $v_1$ and $v_2$, $P_i v_1$ and $P_i v_2$. As briefly noted earlier, the $i$'th quotient group $Q_i$ is isomorphic to the additive group of $\mathbb{F}_{p^r}$. Then, calculation of $P_i v_3$ should be the same as the convolution step at a check node of a code over $\mathbb{F}_{p^r}$. Thus, $P_i v_3 = P_i v_1 \odot P_i v_2$, where $\odot$ represents convolution.

With symbol probabilities, convolution is of $O(p^{2r})$ complexity, since vectors are of length $p^{2r}$. But with coset probability vectors, this is reduced to $O(p^{r^2})$, as lengths are now $p^r$. Note that $T_Q$ contains $(p^r + 1)$ quotient groups, and therefore, that many coset probability vectors have to be computed. This results in an overall complexity of $O(p^{r^2})$, which is still a significant reduction, especially for large fields.

Above insights suggest that decoding using coset probability vectors can offer substantial complexity gains. In subsection B, we present how to expand a code’s Tanner graph to enable this new approach.

B. Graph Expansion

Observations in subsection A lead us to the first step in expanding a graph over $\mathbb{F}_{p^r}$ into a graph over $\mathbb{F}_{p^r}$. Since decoding is to be done using coset probability vectors, and there are $(p^r + 1)$ of these, each variable and check node over $\mathbb{F}_{p^r}$ should first be replaced by that many nodes over $\mathbb{F}_{p^r}$. Each such variable node would represent some coset probability vector, whereas a check node would calculate estimates of these. Connections between the nodes over $\mathbb{F}_{p^r}$ of a single neighboring variable-check node pair over $\mathbb{F}_{p^r}$ depends on the original edge weight.

As Lemma 3 states, a quotient group in $T_Q$ turns into a permutation of another when multiplied with some $\alpha \in \mathbb{F}_{p^r}$. Therefore, permutation of coset probability vectors is a two-fold process. First, these $(p^r + 1)$ vectors are divided between the $(p^r + 1)$ check nodes in the expansion. Each vector then has to be permuted (within itself) once more. This whole process
can be interpreted as connecting the \((p^r + 1)\) variable and check nodes using edges labeled with elements from \(F_{p^r}\). First, coset probability vectors are permuted, and then each vector is individually permuted due to the new edge weights. Fig. 1 shows this initial step in expanding a graph over \(F_{2^2}\). There, the pair of shaded nodes are from the original graph, while the others are from the one over \(F_{2^2}\), with \(a \in F_{2^2}\) and \(b_i \in F_{2^2}\).

![Fig. 1. Initial step in the expansion](image)

This straight-forward expansion would be sufficient if coset probability vectors of a single variable node over \(F_{p^r}\) were independent of each other. But each vector contains some information about all others, and the expansion should account for these dependencies. In order to visualize them, we propose an alternate representation of elements in \(F_{p^r}\).

As each \(Q_i \in T_Q\) is isomorphic to \(H = \{F_{p^r}, +\}\), every coset in one such \(Q_i\) maps with some element of \(H\). We define the value of some \(\gamma \in F_{p^r}\) with respect to some \(Q_i \in T_Q\) as the element of \(H\) that maps with the coset containing \(\gamma\). Since \(|T_Q| = (p^r + 1)\), using values defined with respect to each \(Q_i\), \(\gamma\) can be uniquely represented as a vector of \((p^r + 1)\) elements of \(H\). As an example, consider \(F_{2^2}\), the quadratic extension field of \(F_2\). Trivial cosets of the \((2^2 + 1)\) quotient groups (in \(T_Q\)) are given in Table I. Alternate representations of \(F_{2^4}\) elements are vectors of length \((2^2 + 1)\) over \(F_{2^2}\), which are listed in Table II. \(\alpha\) and \(\omega\) denote primitives of \(F_{2^4}\) and \(F_{2^2}\) respectively.

**TABLE I. TRIVIAL COSETS OF QUOTIENT GROUPS IN \(F_{2^4}\)**

| 00000 | 00000ω | 00000ω² | 1ω²0ω² |
| 10111 | 00000ω¹ | 0ω²0ω² | 1ω²0ω² |
| 011ω² | ω²0ω² | ω²0ω² | 0ω²0ω² |
| 110ω² | 0ω²0ω² | ω²0ω² | 0ω²0ω² |

**TABLE II. ALTERNATE REPRESENTATIONS OF SYMBOLS IN \(F_{2^4}\)**

Close observation of Table II shows that the 16 vectors constitute a 2-dimensional vector space over \(F_{2^2}\). In other words, they are the codewords of a \((2, 5)\) linear code over \(F_{2^2}\). So it is evident that values of some \(\gamma \in F_{2^4}\), with respect to some \(Q_i\)’s in \(T_Q\), contain information about the values with respect to other \(Q_i\)’s. These dependencies between coset probability vectors can easily be captured through the parity-check matrix of the code.

For the general case of \(F_{p^r}\), alternate representation vectors would form a \((2, p^r + 1)\) code over \(F_{p^r}\). Since this code only involves the \((p^r + 1)\) variable nodes of a single node in the original graph, we refer to it as the ‘local code’. Dependencies between coset probability vectors can be represented with the parity-check matrix of this local code. As \((2, p^r + 1)\) code over \(F_{p^r}\) is from the family of non-binary simplex codes, the dual is the \((p^r - 1, p^r + 1)\) Hamming code over \(F_{p^r}\) [10]. \((p^r - 1)\) codewords of this Hamming code forms the ‘local’ parity-check matrix, \(H_L\). For example, \(H_L\) for the case of \(F_{2^2}\), given below, contains 3 codewords of the \((3, 5)\) Hamming code over \(F_{2^2}\).

\[
H_L^{2^4} = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
1 & \omega & 0 & 1 & 0 \\
1 & \omega^2 & 0 & 0 & 1
\end{bmatrix}
\]

From the perspective of the graph expansion, these new parity-check equations are actually a set of additional check nodes, which have to be added for each variable node in the original graph. Each new check node will only be connected with a subset of the \((p^r + 1)\) nodes of one variable node over \(F_{p^r}\). Adding \((p^r - 1)\) ‘local’ check nodes per variable node of the original graph might seem to increase complexity, but any such increase is significantly offset by their low degrees. As parity-check equations are codewords of a Hamming code, it should always be possible to reduce the degree to just 3.

In LDPC codes, short cycles in the Tanner graph are known to negatively impact decoding. \(H_L^{2^4}\) we presented earlier contains a number of such cycles. But, in NB-LDPC codes, particularly troublesome are cycles that do not satisfy the full-rank condition [11]-[12], and the ones created here do not fall in that category. Also, the decoding scheme we propose in Section IV employs a technique to mitigate possible negative effects of cycles among ‘local’ check nodes.

Fig. 2 presents the final form of the graph expansion initiated in Fig. 1. Here, the three local check nodes are represented with hexagons, and \(c_i \in F_{2^2}\).

![Fig. 2. Final expanded graph](image)

**III. DECODING SCHEME**

For decoding, we propose using the Q-ary sum-product algorithm [1]-[3] over the expanded graph. Basic steps of QSPA would remain unchanged, but special characteristics of the expansion require a couple of simple modifications. These modifications and the reasons for them being necessary are explained in the following.

1) Initialization of the Decoder:

Each variable node in the expanded graph represents some ‘coset probability vector’ of a node in the original graph,
defined with respect to some quotient group. Initial symbol probability vector of this node (over \(F_{p^r}\)) is available from the channel. For each coset in the quotient group, initial probability estimate can be calculated as the sum of probabilities of the symbols belonging to that coset. These estimates together would form the initial coset probability vector.

Equation (1) represents initializing \(t^{th}\) node in the expansion of \(n^{th}\) variable node of the original graph. \(P_n^{c_i}\) and \(P_{n,i}\) represent symbol and coset probability vectors, and \(a_k\) represents symbols in coset \(C_i^{j}\).

\[
P_{n,i}(j) = \sum_{a_k \in C_j^{i}} P_n^{c_i}(a_k); \quad j = 0, ..., p^r - 1 \tag{1}
\]

But there are some issues with this seemingly straightforward approach. Even though a node over \(F_{p^r}\) is replaced by \((p^r + 1)\) nodes over \(F_{p^r}\), only a single symbol in \(F_{p^r}\) is transmitted for all of them. Such a symbol can be viewed as two \(F_{p^r}\) symbols. Therefore, channel information received should only be enough to initialize two nodes in the expanded graph, and initializing \((p^r + 1)\) nodes does not seem correct. Errors in channel information may get duplicated, and can propagate rapidly through the graph, leading to performance degradation.

One approach in reducing error propagation is to initialize only 2 nodes out of the \((p^r + 1)\) with channel information, and set all other initial estimates to zero. But simulations show that this also does not result in good decoding performance. We therefore propose a slightly different method. After initializing 2 of the nodes, we calculate initial estimates for the remaining \((p^r - 1)\), and use a scaling factor \(\delta\) \((0 < \delta < 1)\) with those. \(\delta\) has to be optimized per code.

Equation (2) shows how scaling is used in initialization. Note that two of the nodes are initialized as in (1).

\[
P_{n,i}^{c}(j) = \delta \sum_{a_k \in C_j^{i}} P_{n}^{c_i}(a_k); \quad j = 0, ..., p^r - 1 \tag{2}
\]

Concept of the ‘local code’ we introduced in Section II B allows us to view a set of \((p^r+1)\) nodes as representing symbols of a \((2, p^r+1)\) code over \(F_{p^r}\). Then, two of the nodes can be considered to represent information symbols, and the rest, parity symbols. Our method of initializing is appropriate in this sense as well. Scaling factor \(\delta\) is used with the \((p^r - 1)\) parity symbols, which are not actually transmitted. Also, any pair of nodes that can be considered as the information symbols of the local code may be chosen to initialize without scaling factors.

Since decoders more often operate on log or log-likelihood ratio (LLR) domains, probability vectors \(P_{n,i}^{c}\), calculated as in (1) and (2) have to be converted to one of those domains prior to other decoding steps.

2) Probability estimates of ‘local’ check nodes

There are two types of check nodes in the expanded graph: those resulting from the ones in the original graph, and those added to represent dependencies between coset probability vectors. We refer to the two types as ‘global’ and ‘local’ check nodes. Local check nodes are only concerned with relationships between the \((p^r + 1)\) nodes of a single variable node over \(F_{p^r}\).

As was seen in Section II B, local parity-check matrix, which can be constructed with low-weight codewords of a \((p^r - 1, p^r + 1)\) Hamming code, may contain some short cycles, although not of the type considered most harmful in NB-LDPC codes [11]. Still, these cycles can create correlations between messages, and that can negatively impact decoding. Therefore, probability estimates of local check nodes may not be as reliable as those of global nodes. To reflect this, we propose using another scaling factor \(\psi\) \((0 < \psi < 1)\) with local check nodes’ estimates. These estimates will always be scaled by \(\psi\) before being sent to neighboring variable nodes. Optimum value for \(\psi\) should be determined per code, although in majority of codes tested, this was found to be 0.25.

3) Tentative decision at each iteration

In QSPA, a tentative decision is taken by every variable node in each iteration to test whether the decoder has converged to a valid codeword. When decoding on the expanded graph, only ‘information symbols’ of the local code need to be considered for this purpose. Highest probable \(P_{p^r}\) values at the two ‘information symbol’ nodes together would map to an element in \(F_{p^r}\), via a codeword in the local code. That element would be the tentative decision for the corresponding variable node in the original graph.

Apart from these minor modifications, decoding algorithm is the same as QSPA (used over \(F_{p^r}\)) [1]-[3].

IV. SIMULATION RESULTS

In this section, we evaluate performance of the new decoding scheme against existing decoding algorithms for NB-LDPC codes, specifically the best-known decoder QSPA [1], and two simplifications, min-max algorithm [5] and max-log-SP algorithm [13]. Max-log-SP algorithm is a special case of extended min-sum (EMS) algorithm [4], where the two parameters \(n_m\) and \(n_c\) are set to the maximum values possible, respectively the size of the field and check-node degree. Proposed scheme was implemented in LLR-domain, as in [13], and simulations were carried out over the BI-AWGN channel. Maximum decoding iterations was set to 50 for all algorithms. In the following figures, we label the curves of QSPA over the original graph as ‘\(F_{p^r}-\text{qspa}\)’, and those of the new scheme as ‘\(F_{p^r}-\text{qspa}\)’.

![Fig. 3. FER Perf. with (1998,1776) code over \(F_{2^4}\) (C1)](image)

Fig. 3 presents FER performance of algorithms with \(C_1\), a \((1998,1776)\) code over \(F_{2^4}\). This code was constructed by relabeling a regular binary matrix of column weight 4, from [15].
Here the proposed scheme, $F_{22}$-qspa, performs within 0.2dB of the best decoder, $F_{24}$-qspa, at a FER of $10^{-4}$, while offering significant reductions in complexity as will be highlighted in the next section. It has a gap of about 0.05dB with min-max algorithm, and outperforms max-log-SP by close to 0.4dB at similar FERs. It should be noted that min-max decoding [5] is of the same complexity order as QSPA, although it uses comparisons in place of more complex operations. Optimum values for $\delta$ and $\psi$ in the proposed scheme were, respectively, 0.75 and 0.25.

We let $d_c$ and $d_v$ denote average degrees of a check node and a variable node in the original graph. As explained in Section II B, local check nodes are taken to be of degree 3. We consider number of operations of each different type, and assume look-up tables are used for max* operation [13].

![Fig. 4. FER Perf. with (361,288) code over $F_{26}$ ($C_2$)](image)

Performance of algorithms with $C_2$, a (361,288) code over $F_{26}$, is shown in Fig. 4. This is again a regular code of column width 4, generated by re-labeling a binary graph constructed with the progressive edge growth algorithm [16]. Here also, the gap between our scheme and QSPA over the original graph is only about 0.2dB, while with min-max decoding, it is now around 0.13dB. Once more, the proposed scheme outperforms max-log-SP algorithm, this time by 0.3dB. Optimum values for $\delta$ and $\psi$ here were 0.5 and 0.25.

![TABLE III. Decoding Complexity](image)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$d_c$</th>
<th>$d_v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT-QSPA</td>
<td>34.1ms</td>
<td>37.4ms</td>
</tr>
<tr>
<td>LLR-QSPA</td>
<td>47.4ms</td>
<td>94.2ms</td>
</tr>
<tr>
<td>Min-Max</td>
<td>32.3ms</td>
<td>48.2ms</td>
</tr>
<tr>
<td>New Sch.</td>
<td>13ms</td>
<td>13.4ms</td>
</tr>
</tbody>
</table>

V. Decoding Complexity

Major complexity advantage of the proposed scheme is due to decoding being carried out over a field considerably smaller than the one used to define the code. In existing decoding algorithms for NB-LDPC codes [1]-[5], [13], complexity increases with field size, and for most, this increase is quadratic. Thus, complexity gains offered by the proposed scheme would be particularly attractive, especially for larger fields.

In the following, we compare the complexity of our approach with QSPA and min-max algorithms. We consider LLR-domain implementations of both QSPA and the new scheme. It should be noted that LLR-QSPA is considered more suitable for hardware implementations than other variants of QSPA, due to better numerical stability [14].

An NB-LDPC code over $F_{p^2}$ is considered in the comparison. Complexities of the two major steps in decoding, check node and variable node operations, are compared separately, in Table III. Since the expansion replaces each node over $F_{p^2}$ with $(p^2 + 1)$ nodes over $F_p$, complexity of that many nodes in the new approach is compared with the complexity of a single node in existing algorithms. Complexity of operations at ‘local’ check nodes is added to that of variable nodes, since those nodes are added per variable node of the original graph.

![TABLE IV. Decoding Latency per Iteration](image)

Table IV shows that the proposed scheme is several times faster than any existing algorithm, in both cases considered. In particular, it is almost 3 times faster than the fastest implementation of QSPA, both with $C_1$ and $C_2$. With performance loss less than 0.2dB, performance-complexity trade-off offered by the new scheme is particularly attractive. We refer the reader to the longer version of the paper [17] for a more in-depth study of this performance-complexity trade-off.

VI. Conclusion

In this paper, we have proposed a new decoding algorithm for NB-LDPC codes defined over fields of type $F_{p^2}$ that manages to offer significant complexity advantages, while performance losses compared to the best known algorithms are minimal. Different variations of this scheme, such as a FFT-based implementation, promises further complexity gains. New algorithm is based on a novel graph expansion, which could be useful in other applications as well.
REFERENCES


