Multiple-votes parallel symbol-flipping algorithm for non-binary LDPC codes

Nhat Quang Nhan, Telex M.N. Ngatched, Octavia Dobre, Philippe Rostaing, Karine Amis Cavalec, Emanuel Radoi

Multiple-Votes Parallel Symbol-Flipping Decoding Algorithm for Non-Binary LDPC Codes

Nhat-Quang Nhan, Telex M. N. Ngatched, Octavia A. Dobre, Philippe Rostaing, Karine Amis, and Emanuel Radoi

***Abstract*—A novel decoding algorithm for non-binary low density parity check (NB-LDPC) codes is proposed. The algorithm builds on the recently designed parallel symbol-flipping decoding (PSFD) algorithm and combines a technique of error estimation and a method of multiple voting levels from each unsatisfied check-sum to the corresponding variable nodes. Simulations results, performed on a number of NB-LDPC codes of various lengths and column weights constructed using several methods, show that the new algorithm not only avoids using code-dependent voting threshold but also improves the error rate performance of the PSFD algorithm, particularly for low column weight parity-check matrices.**

***Index Terms*—NB-LDPC codes, low complexity decoding algorithm, symbol flipping.**

I. Introduction

FIRST introduced by Gallager in [1], and subsequently rediscovered by Davey and Mackay in [2], non-binary low density parity check (NB-LDPC) codes outperform their binary counterparts, especially for short-to-moderate length codes. Just like with binary LDPC codes, the algorithms for decoding NB-LDPC codes can be classified into three general categories: hard-decision decoding [3], soft-decision decoding [4]–[7], and hybrid decoding (also known as reliability-based decoding) [8]–[11]. From an implementation point of view, hard-decision decoding is the simplest in complexity. However, its simplicity results in a relatively poor performance that can be as far away as a few decibels from that of soft-decision decoding. Soft-decision decoding provides the best performance but requires the highest computational complexity. Hybrid decoding is in between the two extremes and provides a good trade-off between performance and complexity.

N.-Q. Nhan is with the Université Européenne de Bretagne, Université de Brest, 29238 Brest, France, with Telecom Bretagne, Signal and Communications Department, Institut Mines-Télécom, 29238 Brest, France and also with CNRS, UMR 6285 Lab-STICC (e-mail: nhat-quang.nhan@univ-brest.fr).

T. M. N. Ngatched and O. A. Dobre are with the Department of Electrical and Computer Engineering, Memorial University of Newfoundland, St. John's, NL A1B 3X9, Canada (e-mail: tngatched@grenfell.mun.ca; odobre@mun.ca).

P. Rostaing and E. Radoi are with the Université Européenne de Bretagne, Université de Brest, 29238 Brest, France, and also with CNRS, UMR 6285 Lab-STICC (e-mail: philippe.rostaing@univ-brest.fr; emanuel.radoi@univ-brest.fr).

K. Amis is with the Telecom Bretagne, Signal and Communications Department, Institut Mines-Telecom, 29238 Brest, France, and also with CNRS, UMR 6285 Lab-STICC (e-mail: karine.amis@telecom-bretagne.eu).

Among reliability-based message-passing algorithms, the parallel symbol flipping decoding (PSFD) algorithm recently introduced in [10] offers one of the best trade-off between performance and complexity. Using a flipping function which combines both the weighted check-based message of the normalized check-sums and the variable-based message of the received sequence, the algorithm identifies, in each decoding iteration, the relatively unreliable symbols in the hard-decision symbol sequence and decodes them based on the corresponding flipping symbols. The unreliable symbols are found using a voting system whereby each unsatisfied check node (CN) gives one vote to the variable node (VN) with the largest flipping function checked by it. Variable nodes thus accumulate votes and those with a total number of votes exceeding a predefined threshold are identified as unreliable. By only playing with the reliabilities of hard-decisions, PSFD is a good choice to decode high rate finite field NB-LDPC codes. However, simulation results show that PSFD is suitable for decoding only regular NB-LDPC codes whose parity-check matrices have large column weights, e.g., column weights of at least 8. In addition, PSFD employs a code-dependent voting threshold, which should be optimized through simulation. Furthermore, the computation of its flipping function involves a pair of scaling factors, (η, λ), which also depend on the code and is optimized through simulation.

This letter proposes a new algorithm, referred to as multiple-votes PSFD (MV-PSFD). When compared with PSFD, it introduces a method of error estimation which results in avoiding the use of the voting threshold, and passes multiple votes from the unsatisfied CNs to the corresponding VNs. The proposed algorithm significantly outperforms PSFD for low column weight parity check matrices, as shown by simulation results. Such improvements are achieved with a similar computational complexity.

The organization of this letter is as follows: In Section II, some preliminaries and the PSFD algorithm are briefly reviewed. The proposed algorithm and its complexity analysis are introduced in Section III. Simulation results are presented in Section IV. Finally, Section V concludes the letter.

II. Preliminaries

A. Notations and Definitions

A regular NB-LDPC code $\mathcal{C}$ of length N and dimension K over the Galois field of order q $(\mathrm{GF}(q))$ is completely described by a row-column (RC)-constrained parity-check matrix of size $M \times N$ $(M \geq K)$ $\mathbf{H} = [h_{j,i}]$ over $\mathrm{GF}(q)$, which has a constant column weight d_v and a constant row weight d_c. For practical purpose, we only consider binary extension fields, where $q = 2^p$. Let $\mathcal{M}_j = \{i : 0 \leq i < N, h_{ji} \neq 0\}$ be the set of all indices of VNs which connect to the CN c_j and $\mathcal{N}_i = \{j : 0 \leq j < M, h_{ji} \neq 0\}$ be the set of all indices of CNs which connect to the VN v_i.

Suppose that a regular (N, K) NB-LDPC code is used for error control over a binary-input additive white Gaussian noise (BIAWGN) channel with zero mean and two-sided power spectral density $N_0/2$. Assume binary phase-shift-keying (BPSK) signaling with unit energy. A codeword $\boldsymbol{c} = (c_0, c_1, \ldots, c_{N-1})$ in $\mathcal{C}$, where $c_n = (c_{n,0}, c_{n,1}, \cdots, c_{n,p-1}) \in \mathrm{GF}(2^p)$ with $c_{n,i} \in \mathrm{GF}(2)$, is mapped into the sequence $\boldsymbol{x} = (x_0, x_1, \ldots, x_{N-1})$ before its transmission, where $x_n = (x_{n,0}, x_{n,1}, \cdots, x_{n,p-1})$ with the mapping rule $x_{n,i} = (2c_{n,i} - 1) \in \{+1, -1\}$. Let $\boldsymbol{y} = (y_0, y_1, \ldots, y_{N-1})$ be the soft-decision received sequence at the output of the receiver matched filter. For $0 \le i < N$, $y_n = (y_{n,0}, y_{n,1}, \cdots, y_{n,p-1}) = (x_{n,0} + \nu_{n,0}, x_{n,1} + \nu_{n,1}, \cdots, x_{n,p-1} + \nu_{n,p-1})$ in which the $\nu_{n,i}$'s are statistically independent Gaussian random variable with zero mean and variance $N_0/2$. The hard-decision symbol sequence at the decoder is denoted as $\mathbf{z} = (z_0, z_1, \ldots, z_{N-1})$.

For any hard-decision symbol sequence $\mathbf{z}$, the syndrome vector $\boldsymbol{s}$ is computed as $\mathbf{s} = \mathbf{z} \cdot \mathbf{H}^T$, where $s_j = h_{j0}z_0 + h_{j1}z_1 + \cdots + h_{j(N-1)}z_{N-1}$ with $0 \le j < M$. s_j is known as the j^{th} check-sum of $\mathbf{z}$. $\mathbf{z}$ is a valid codeword in $\mathcal{C}$ if and only if $\mathbf{s} = \mathbf{0}$. Let $\mathcal{S}_i(z) = \{s_j : j \in \mathcal{N}_i\}$ be a set of check-sums that contain the symbol z_i at the VN v_i and $\tilde{\mathcal{S}}_i(z) = \{\tilde{s}_{ji}(z) = h_{ji}^{-1}s_j : s_j \in \mathcal{S}_i(z)\}$ be a normalized check-sum set. Finally, let $\tilde{\mathcal{S}}_i^*(z)$ be $\tilde{\mathcal{S}}_i(z)$ excluding zero elements, i.e., $\tilde{\mathcal{S}}_i^*(z) = \tilde{\mathcal{S}}_i(z) \setminus \{0\}$.

B. PSFD Algorithm [10]

Given the initial hard-decision symbol vector $\mathbf{z}^{(0)} = (z_0^{(0)}, z_1^{(0)}, \ldots, z_{N-1}^{(0)})$, the PSFD algorithm iteratively flips erroneous symbols in $\mathbf{z}^{(0)}$ until either a codeword is found or a maximum number of iteration is reached. In other words, the algorithm finds an error symbol vector $\mathbf{e} = [e_0, e_1, \ldots, e_{N-1}]$, such that $\mathbf{z} = \mathbf{z}^{(0)} - \mathbf{e}$ is a codeword. The error symbol vector is recursively computed by $\mathbf{e}^{(0)} = \mathbf{0}$ and $\mathbf{e}^{(l+1)} = \mathbf{e}^{(l)} + \boldsymbol{\varepsilon}^{(l)}$, where each component of $\boldsymbol{\varepsilon}^{(l)}$ is chosen so as to maximize a flipping function that is updated at each iteration and which is defined by

$$F_i^{(l)} = \max_{\alpha \in \tilde{\mathcal{S}}_i^{*(l)}(z)} F_i^{(l)}(\alpha). \tag{1}$$

In (1), $F_i^{(l)}(\alpha)$ with $\alpha \in GF(q)$ is given by

$$F_i^{(l)}(\alpha) = E_i^{(l)}(\alpha) - I_i\left(\alpha + e_i^{(l)}\right), \tag{2}$$

where the weighted check-based message about $z_i(l)$, $E_i^{(l)}(\alpha)$, and the variable-based message, $I_i(\alpha + e_i^{(l)})$, are respectively defined by

$$E_i^{(l)}(\alpha) = \sum_{j' \in \mathcal{N}_i : \tilde{s}_{j'i}^{(l)}(z) = \alpha} W_{j'i} - \sum_{j' \in \mathcal{N}_i : \tilde{s}_{j'i}^{(l)}(z) = 0} W_{j'i}, \tag{3}$$

and

$$I_i(\alpha) = \eta \phi_i(\alpha), \tag{4}$$

where η is a scaling factor. W_{ji} in (3) is the weighting coefficient contributed to z_i by other $z_{i'}$'s checked by s_j and is defined by

$$W_{ji} = \lambda \min_{i' \in \mathcal{M}_j \setminus \{i\}} R_{i'}, \tag{5}$$

where λ is a scaling factor. R_i is the reliability of the initial hard-decision symbol $z_i^{(0)}$ and is given by

$$R_i = \min_{\alpha \in GF(q)} \phi_i(\alpha), \tag{6}$$

with $\phi_i(\alpha)$ being a log-likelihood ratio representing the gap of likelihood between $z_i^{(0)}$ and $z_i^{(0)} - \alpha$ defined by

$$\phi_i(\alpha) = \frac{N_0}{4} \ln \frac{P\left(v_i = z_i^{(0)} | y_i\right)}{P\left(v_i = z_i^{(0)} - \alpha | y_i\right)}. \tag{7}$$

To speed up the decoding, it is essential to be able to flip multiple symbols in each decoding iteration. The PSFD algorithm achieves this by adopting a voting system whereby each unsatisfied CN gives one vote to the VN with the largest flipping function checked by it. At the l^{th} iteration, VN v_i accumulates the number of votes, $\mathcal{V}_i^{(l)}$, from all the unsatisfied CNs. That is

$$\mathcal{V}_i^{(l)} = \sum_{j \in \mathcal{N}_i} V_{ji}^{(l)}, \tag{8}$$

where $V_{ji_0}^{(l)} = 1$ if $s_j^{(l)} \neq 0$; otherwise $V_{ji}^{(l)} = 0$ with $i_0 = \arg\max_{i' \in \mathcal{M}_j} F_{i'}^{(l)}$. The hard-decision $z_i^{(l)}$ will be flipped to $z_i^{(l+1)} = z_i^{(l)} - \varepsilon_i^{(l)}$ when $\mathcal{V}_i^{(l)} > V_{th}$, where V_{th} is a code-dependent threshold.

III. Multiple-Votes PSFD Algorithm

A. Algorithm

The voting threshold, V_{th}, in the PSFD is code-dependent and should be optimized through simulation [10]. In the sequel, we design a new decoding algorithm which mitigates this drawback while maintaining the strong aspects of the PSFD.

First, we introduce multiple voting levels, allowing each unsatisfied CN to pass more than one vote to the VNs checked by it. Without loss of generalization, let us introduce two voting levels $\zeta_0 > \zeta_1 > 0$. At each VN v_i, the voting function is still given by (8) but with the voting principle defined as follows

For $s_j^{(l)} \neq 0$,

- $V_{ji_0}^{(l)} = \zeta_0$ with $i_0 = \arg\max_{i' \in \mathcal{M}_j} F_{i'}^{(l)}$.
- $V_{ji_1}^{(l)} = \zeta_1$ with $i_1 = \arg\max_{i' \in \mathcal{M}_j \setminus \{i_0\}} F_{i'}^{(l)}$.
- $V_{ji}^{(l)} = 0$ elsewhere.

The unsatisfied CN c_j gives ζ_0 to the VN that has the largest flipping function and gives ζ_1 to the VN that has the second largest flipping function. Though it might appear that the two factors ζ_0 and ζ_1 should be optimized, we will show through simulations, in the next section, that their optimal values are not code-dependent.

Secondly, we derive a relationship between the syndrome weight, the number of errors, and the parity-check matrix column weight d_v from the following Lemma [12].

Lemma 1: For regular LDPC matrices, the average syndrome weight increases linearly with the number of errors.

Proof: (Heuristic.) For any regular LDPC matrix, a plot of the average syndrome weight in terms of the number of errors yield a straight line with slope d_v. □

Based on the above Lemma, at iteration l, the number of errors in the hard-decision vector $\mathbf{z}^{(l)}$, and hence the number of symbols to be flipped, can be estimated by

$$N_e{}^{(l)} = \left\lfloor \frac{w^{(l)}(\mathbf{s})}{d_v} \right\rfloor, \tag{9}$$

where $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x, and $w^{(l)}(\mathbf{s})$ is the syndrome weight at iteration l defined by

$$w^{(l)}(\mathbf{s}) = \sum_j \left(s_j^{(l)} \neq 0 \right). \tag{10}$$

The use of a voting threshold can therefore be avoided by flipping the first $N_e{}^{(l)}$ symbols with the highest votes $\mathcal{V}_i^{(l)}$.

The steps of the new algorithm, which we call Multiple-Votes Parallel Symbol-Flipping Decoding (MV-PSFD), can be summarized as follows.

MV-PSFD algorithm

1: **Inputs**: Maximum number of iterations l_{max}, hard-decision $\mathbf{z}^{(0)}$ and the received symbol vector $\mathbf{y}$.
2: **Initialization**: Set iteration index $l = 0$, maximum number of iterations to l_{max}, and error symbol vector $\mathbf{e} = \mathbf{0}$. Find $\phi_i(\alpha)$ by (7) and the variable-based message $I_i(\alpha)$ by (4). Compute the coefficients W_{ji} by (5) and store them.
3: **while** $l \leq l_{max}$ **do**
4: Compute the syndrome $\mathbf{s}^{(l)}$
5: **if** $\mathbf{s}^{(l)} = \mathbf{0}$ **then**
6: Stop the while loop
7: **end if**
8: With $0 \leq i < N$, compute the flipping function $F_i^{(l)}$ and find its corresponding flipping symbol $\alpha = \varepsilon_i^{(l)}$ by (1)
9: Compute $\mathcal{V}_i^{(l)}$ by (8) using the multiple-votes principle
10: Estimate the number of errors $N_e{}^{(l)}$ by (9)
11: With $0 \leq i < N$, sort VNs according to descending order of $\mathcal{V}_i^{(l)}$
12: Flip the first $N_e{}^{(l)}$ VNs by setting new hard-decision symbols $z_i^{(l+1)} = z_i^{(l)} - \varepsilon_i^{(l)}$ and new error symbols $e_i^{(l+1)} = e_i^{(l)} + \varepsilon_i^{(l)}$
13: $l \leftarrow l + 1$
14: **end while**
15: **Output**: $\mathbf{z}^{(l)}$.

B. Complexity Computation

We evaluate the computational complexity of the new algorithm based on the number of integer addition (IA), multiplication (IM) and comparison (IC) operators as in [4].

At the initialization, the computations of $I_i(\alpha)$ and R_i (BPSK modulation) need $\mathcal{O}(Nqlog_2(q))$ IAs, $\mathcal{O}(Nq)$ IMs, and $\mathcal{O}(N(log_2(q)-1))$ ICs. Additionally, $\mathcal{O}(M(2d_c-3))$ ICs and $\mathcal{O}(Md_c)$ IMs are needed for W_{ji}.

At each iteration, check-sums computations require $\mathcal{O}(M(d_c-1))$ IAs and $\mathcal{O}(Md_c)$ IMs. The normalized check-sums $\tilde{\mathcal{S}}_i^{*(l)}$ need $\mathcal{O}(Md_c)$ IMs. The flipping functions and flipping symbols require at most $\mathcal{O}(2N(d_v-1))$ IAs for $F_i^{(l)}(\alpha)$ and at most $\mathcal{O}(N(d_v-1))$ ICs for $F_i^{(l)}$. The error estimation needs at most $\mathcal{O}(M-1)$ IAs for $d_s{}^{(l)}$ and $\mathcal{O}(1)$ IM for $N_e{}^{(l)}$.

TABLE I
COMPLEXITY BY OPERATORS AT EACH ITERATION OF NB-LDPC DECODING ALGORITHMS

	IA	IM	IC
PSFD [10]	3σ	2σ	$2\sigma - M$
MV-PSFD	$3\sigma + M - 1$	$2\sigma + 1$	$2\sigma - M - N + Nlog_2(N)$

For voting function $\mathcal{V}_i^{(l)}$, it requires at most $\mathcal{O}(M(d_c-1))$ ICs and at most $\mathcal{O}(M)$ IAs. If the Quick Sort algorithm [13] is used, the sorting needs at most $\mathcal{O}(Nlog_2(N))$ ICs. Note that the Quick Sort is not the best choice in terms of operators saving, but it is the fastest algorithm to achieve high system throughput. Finally, $\mathcal{O}(2N)$ IAs are used for $z_i^{(l+1)}$ and $e_i^{(l+1)}$.

The total computational complexity of MV-PSFD and PSFD algorithm for each iteration is shown in Table I. From [10], it is proved that PSFD is the algorithm that costs the lowest decoding complexity among the reliability-based decoding algorithms. The MV-PSFD requires slightly more comparison operators than PSFD due to the sorting process. However, in practice, the cost of comparators is much cheaper than those of multiplications and additions. Thus, for small to moderate length codes, we can conclude that the complexity of MV-PSFD is close to the original PSFD.

IV. SIMULATION RESULTS

In this section, using Monte Carlo simulations, the error performance in terms of bit-error rate (BER), frame-error rate (FER), and average number of iterations as a function of the rate-normalized signal-to-noise ratio (SNR) per information bit (E_b/N_0), of the proposed algorithm is compared with that of the PSFD algorithm on a number of NB-LDPC codes of various constructions with short to medium block lengths. These constructions include Euclidean geometry [14], progressive edge-growth [15] and the original method of Gallager. The maximum number of iterations is set to 50 for both algorithms. At each SNR value, at least 100 erroneously received codewords are detected.

The optimal values of η, λ, V_{th}, ζ_1, and ζ_0, are selected through simulations for each code. It should be noted that only the ratios $\frac{\eta}{\lambda}$ and $\frac{\zeta_1}{\zeta_0}$ are actually needed. An interesting result is that the optimum ratio $\frac{\zeta_1}{\zeta_0}$ is $\frac{2}{3}$ for all codes. Moreover, no significant difference in error performance was observed using $\frac{\zeta_1}{\zeta_0} = \frac{2}{3}$ and $\frac{\zeta_1}{\zeta_0} = 1$. Thus, the parameters ζ_1 and ζ_0 in MV-PSFD are not code-dependent.

Due to space considerations, we only present results for two codes, referred to as Code 1 and 2. Code 1 is a $(d_v = 3, d_c = 6)$ [16] and Code 2 is a $(d_v = 5, d_c = 10)$ [17]; both are regular (102, 204) NB-LDPC codes over $\mathrm{GF}(2^4)$ constructed based on Gallager's method, whose parity-check matrix satisfies the RC-constraint.

Fig. 1 shows the BER and FER performances of Code 1. It can be observed that, unlike PSFD, the performance of the proposed algorithm is not sensitive to the parameters η and λ. Moreover, the proposed algorithm outperforms the optimal PSFD by about 0.5 dB at the BER of 10^{-5}. Note that the optimal PSFD is plotted with the optimized $\frac{\eta}{\lambda}$ and V_{th}.

Fig. 2 shows the BER and FER performances of Code 2. Contrary to the previous case, the performance of the proposed algorithm is sensitive to the parameters η and λ. However, using the optimal value of $\frac{\eta}{\lambda}$, the MV-PSFD outperforms the PSFD

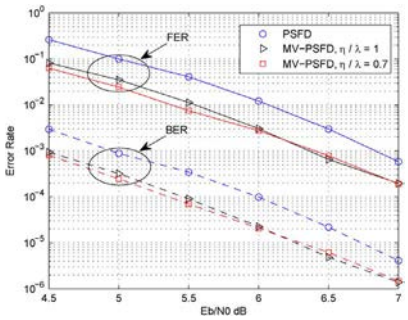

Fig. 1. FER (solid) and BER (dashed) performance versus rate-normalized SNR of MV-PSFD and PSFD for Code 1.

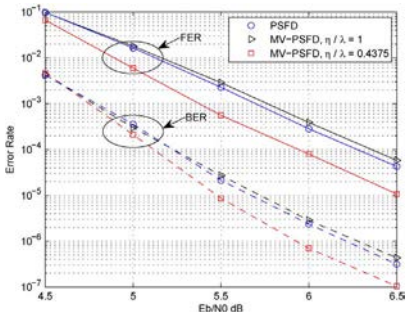

Fig. 2. FER (solid) and BER (dashed) performance versus rate-normalized SNR of MV-PSFD and PSFD for Code 2.

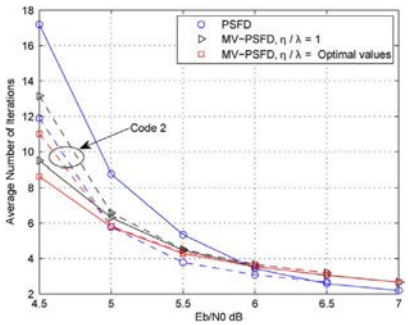

Fig. 3. Average number of iterations versus rate-normalized SNR of MV-PSFD and PSFD for Code 1 (solid) and 2 (dashed).

with a gain of about 0.3 dB at the BER of 10^{-5} and 0.4 dB at the BER of 10^{-6}. It can also be observed that, when MV-PSFD is used without the optimal values of η and λ, the performance is close to that of PSFD.

We would like to point out that, although the proposed algorithm could be generalized to more than two voting levels, no significant performance improvement was observed for more than two voting levels for all the codes simulated. Therefore, taking into account the additional complexity, two voting levels appears to be the best choice for the proposed algorithm. Furthermore, for regular NB-LDPC codes whose parity-check matrices have large column weights, i.e., column weights of at least 8, the proposed algorithm only slightly outperforms the standard one. Results are not included here due to the space consideration.

With these performance studies in mind, we move on to the average number of iterations comparison. Fig. 3 depicts the average number of iterations for Codes 1 and 2. We see that, at low to medium SNR, the proposed algorithm with optimized parameters converges faster than the standard one. However, at high SNR, the convergence of the two algorithms is similar. Overall, one can conclude that there is no significant difference between the two algorithms in terms of convergence rate. As such, the excellent performance of the proposed algorithm is not obtained at the expense of the convergence rate.

V. Conclusion

In this letter, a new algorithm based on the PSFD algorithm for regular NB-LDPC codes has been proposed. Simulation results performed on a number of NB-LDPC codes of various column weights show that the proposed algorithm significantly outperforms the standard one for low column weight parity-check matrices, i.e., column weights less than 8, with a similar complexity.

Acknowledgment

We would like to thank the Editor, Prof. Zheng Ma, and the anonymous reviewers for their constructive feedback. We also would like to acknowledge Prof. Emmanuel Boutillon and Dr. Deyuan Chang for their helpful discussions.

References

[1] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.
[2] M. C. Davey and D. J. MacKay, "Low-density parity-check codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165–167, Jun. 1998.
[3] X. Zhang, F. Cai, and S. Lin, "Low-complexity reliability-based message-passing decoder architectures for non-binary LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 11, pp. 1938–1950, Jul. 2012.
[4] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over GF(q)," in *Proc. IEEE ICC*, 2004, pp. 772–776.
[5] D. Declercq and M. Fossorier, "Decoding algorithms for non-binary LDPC codes over GF(q)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, Apr. 2007.
[6] E. Boutillon and L. Conde-Canencia, "Bubble check: A simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders," *Electron. Lett.*, vol. 46, no. 9, pp. 633–634, Apr. 2010.
[7] C.-L. Wang, X. Chen, Z. Li, and S. Yang, "A simplified min-sum decoding algorithm for non-binary LDPC codes," *IEEE Trans. Commun.*, vol. 61, no. 1, pp. 24–32, Feb. 2013.
[8] C. Chen, B. Bai, X. Ma, and X. Wang, "A symbol-reliability based message-passing decoding algorithm for non-binary LDPC codes over finite fields," in *Proc. IEEE ISTC*, 2010, pp. 251–255.
[9] B. Liu, J. Gao, W. Tao, and G. Dou, "Weighted symbol-flipping decoding algorithm for non-binary ldpc codes with flipping patterns," *J. Syst. Eng. Elect.*, vol. 22, no. 5, pp. 848–855, Oct. 2011.
[10] C.-C. Huang, C.-J. Wu, C.-Y. Chen, and C.-C. Chao, "Parallel symbolflipping decoding for non-binary LDPC codes," *IEEE Commun. Lett.*, vol. 17, no. 6, pp. 1228–1231, Jun. 2013.
[11] F. Garia-Herrero and D. Declercq, "Non-binary LDPC decoder based on symbol flipping with multiple votes," *IEEE Commun. Lett.*, vol. 18, no. 5, pp. 749–752, May 2014.
[12] M. Bossert, *Channel Coding for Telecommunications*. Hoboken, NJ, USA: Wiley, 1999.
[13] C. A. Hoare, "Quicksort," *Comput. J.*, vol. 5, no. 1, pp. 10–16, Nov. 1962.
[14] L. Zeng *et al.*, "Construction of non-binary cyclic, quasi-cyclic and regular LDPC codes: A finite geometry approach," *IEEE Trans. Commun.*, vol. 56, no. 3, pp. 378–387, Mar. 2008.
[15] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth Tanner graphs," in *Proc. IEEE GLOBECOM*, 2001, vol. 2, pp. 995–1001.
[16] David Mackay's Code Resource. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/EN/C/204.33.486
[17] David Mackay's Code Resource. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/EN/C/204.55.153