Performance Analysis of Verification-Based Decoding for Packet-Based LDPC Codes over Binary Symmetric Channel

Bin Zhu*†, Defeng (David) Huang† and Sven Nordholm*
*Western Australian Telecommunications Research Institute (WATRI)
39 Fairway, Crawley WA, Australia 6009
email: {zhub, sven}@watri.org.au
†School of Electrical, Electronic and Computer Engineering
University of Western Australia, Crawley WA, Australia 6009
email: huangdf@ee.uwa.edu.au

***Abstract*—In this paper, we propose a statistical model to analyze the performance of verification-based algorithm (VA) for packet-based low-density parity-check (LDPC) codes over binary symmetric channel (BSC). In contrast to the analysis of VA in the literature, we propose to take the false verification into consideration. For a given ensemble of LDPC codes and channel parameters, the proposed analysis model gives an efficient way to find the average performance of packet-based LDPC codes with verification-based decoding. Through numerical results, we find that the proposed method can provide a close estimation of frame error rate (FER) for packet-based LDPC codes with verification-based decoding over BSC for all crossover probabilities of practical interests.**

I. Introduction

Research on low-density parity-check (LDPC) codes [1]–[4] has traditionally focused on small alphabets. With the development of communication networks and high speed wireless systems, the operation units are normally blocks of bits organized as packets. As a result, some efforts have been made both in code design and decoding algorithms of packet-based channel codes [5]–[9]. For packet erasure channel, the Fountain codes such as LT-codes [6] and Raptor codes [7] have been proposed to be used for realizing multi-cast and content delivering networks. As for packet-based LDPC codes, verification-based decoding approach (VA) has been proposed by Luby and Mitzenmacher [8]. Their suggested decoding algorithm consists of two iterative stages: *verification* and *correction*. At the verification stage, if the sum of all the neighboring variable nodes of a check node equals zero, all these variable nodes are verified. At the correction stage, one unverified variable node can be corrected if all its other neighboring variable nodes incident to a check node have been verified. Its updated value is computed as the sum of other neighbor variable nodes.

Luby *et al.* also analyzed the performance of VA over q-ary symmetric channels, where q is a large number. In this scenario, the probability of a variable node that is falsely verified was assumed to be negligible. In this paper, we propose a different statistical model to investigate the performance of VA over binary symmetric channel (BSC). The main contribution of the proposed model is to take the falsely verified variable nodes into consideration, which is one of the dominant factors to affect the performance of VA. Numerical results verify that the proposed model provides a good estimation of frame error rate (FER) performance in various channel conditions for an ensemble of LDPC codes at any number of iterations.

This research was supported in part under Australian Research Council's *Discovery Projects* funding scheme (project number DP0877616).

The rest of the paper is organized as follows. In Section II, the preliminary of LDPC codes and the verification-based decoding are introduced. The proposed statistical model is described in Sections III. In Section IV, numerical results are presented. Section V concludes the paper.

II. Verification-based Decoding for Packet-based LDPC Codes

A. Packet-based LDPC Codes

For packet-based LDPC codes, we refer a packet of l bits as a symbol and the sum operation is a bitwise EXCLUSIVE-OR between packets. An LDPC code can be represented by a bipartite graph consisting of variable nodes and check nodes. The variable nodes represent the symbols of a codeword and the check nodes determine the constraint that the sum of neighboring variable nodes equals zero, based on the parity check matrix H. A variable node and a check node are denoted as v and c, respectively. A pair of nodes $\{v, c\}$ represents an edge, which is the path for message passing.

We use $\mathcal{C}(n, \lambda, \rho)$ to denote an ensemble of LDPC codes, where n denotes the length of a codeword and (λ, ρ) denote the degree distributions. For a regular LDPC code, the degree of variable nodes and the degree of check nodes are constant, which are denoted as d_v and d_c, respectively. For an irregular LDPC code, the degrees of both variable nodes and check nodes are governed by degree distributions, which can be represented by degree distribution polynomials [10].

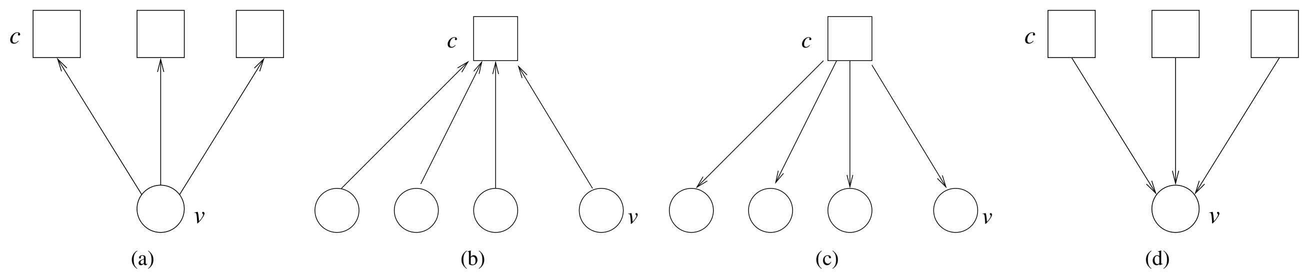

Fig. 1. (a) Messages passing from variable node v to neighboring check nodes. (b) Node processing at check node c using incoming messages from neighboring variable nodes. (c) Message passing from check node c to its neighboring variable nodes. (d) Node processing at variable node v using incoming messages from neighboring check nodes.

B. Verification-Based Decoding

In the verification-based decoding algorithm [8], there are two iterative stages called *verification* and *correction*, respectively. With each variable node v, there are three possible values: a received value r_v, a current value c_v and a final value f_v. For verification, a variable node v is verified by a check node c if the following equation is satisfied.

$$\sum_{v \in V_c} c_v = 0 \tag{1}$$

where V_c is the set of neighboring variable nodes linked with check node c and c_v is the current value of variable node v.

As for correction, an unverified variable node can be corrected if all its other neighboring variable nodes incident to a check node have been verified. Specifically, let Ω denote the set of verified neighboring variable nodes of a check node c and v_u is the *only* unverified variable node incident to check node c. The correction of variable node v_u is achieved as follows:

$$f_{v_u} = c_{v_u} = \sum_{v_j \in \Omega, v_j \neq v_u} c_{v_j}. \tag{2}$$

In VA, one variable node can be either of two states: *verified* or *unverified*. Once a variable node is verified, its value and state are finalized. However, a verified variable node can also be either *correct* or *false*. As a result, if a variable node is verified and it contains the correct value, we call it a *correct verification*. On the other hand, if a variable node is verified and it contains an incorrect value, we call it a *false verification*. If a variable node is unverified, we call it an *un-verification*. Since the verified variable node fixes its value and state during the decoding process, a false verification results in an error frame directly, regardless of the number of iterations. An un-verification at the last iteration also results in an error frame since unverified codewords are normally discarded and not passed to higher layers of a communication protocol [11].

In the following, we reformulate the VA as a message passing algorithm[1]. In message passing algorithm, messages are passed along the edges between nodes and nodes process the incoming messages to determine the outgoing messages. The decoding process proceeds in iterations. In each iteration, it includes the two stages *verification* and *correction*. As shown in Fig. 1, each stage involves cycles of message passing followed by processing at the variable nodes and the check nodes.

[1]We note here the message passing reformulation of VA algorithm is from performance analysis perspective, not from implementation perspective.

At the verification stage, the states of variable nodes are determined. The cycle of message passing and processing contains *four* steps as shown in Fig. 1.

- *Step 1. Message passing from variable nodes to check nodes.* As shown in Fig. 1(a), each variable node sends a message which contains its current value through the edges to its neighboring check nodes.
- *Step 2. Processing at the check node.* As shown in Fig. 1(b), the check node processes *all* incoming messages from its neighboring variable nodes by using (1), generating messages that convey the suggested state: *verified* or *unverified* depending on whether (1) is held or not.
- *Step 3. Message passing from check nodes to variable nodes.* As shown in Fig. 1(c), the check node delivers the verified or unverified messages to its neighboring variable nodes.
- *Step 4. Processing at the variable node.* As shown in Fig. 1(d), the variable node determines its state based on *all* incoming messages from adjacent check nodes. If at least one of the received messages includes verified state, the variable node is deemed to be verified, thereby finalizing its current value.

At the correction stage, an unverified variable node can change its value and state. There are also *four* steps at this stage as shown in Fig. 1.

- *Step 1. Message passing from variable nodes to check nodes.* As shown in Fig. 1(a), each variable node sends the message which contains its current value and *state* through the edges to its adjacent check nodes.
- *Step 2. Processing at the check node.* As shown in Fig. 1(b), the check node calculates the proposed value of one unverified variable node by using (2) if *all other* incoming messages from its neighboring variable nodes contain verified states.
- *Step 3. Message passing from check nodes to variable nodes.* As shown in Fig. 1(c), the check node passes the variable node a message which contains the updated value

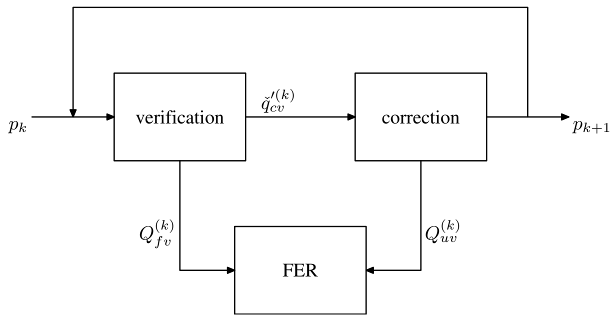

Fig. 2. Schematic of the proposed statistical model for FER estimation. p_k is the input bit error probability at the kth iteration. $\check{q}'^{(k)}_{cv}$ denotes the proportion of correctly verified variable nodes among the variable nodes that are not falsely verified at *Step 1* of the correction stage of the kth iteration. $Q^{(k)}_{fv}$ denotes the probability of a codeword containing at least one variable node with false verification at the kth iteration, given that no variable nodes of the codeword are falsely verified in the first $k-1$ iterations. $Q^{(k)}_{uv}$ denotes the probability of a codeword containing at least one unverified variable node but no falsely verified variable node at the kth iteration, given that no variable nodes of the codeword are falsely verified in the first $k-1$ iterations.

and verified state, if a correction is required. Otherwise, the message confirms the value and state in *Step 1*.

- *Step 4. Processing at the variable node.* As shown in Fig. 1(d), an unverified variable node changes its value and state as long as it receives one correction message from its neighbouring check nodes.

III. FER Performance Analysis of VA

In this section, a statistical model as shown in Fig. 2 is proposed to analyze the message passing algorithm in Section II. The statistical model operates iteratively and includes two stages for each iteration. For the kth iteration, the system input is the bit error probability p_k and the system output is the updated bit error probability p_{k+1}, which is also used as the input for the next iteration. When $k=0$, p_0 is the initial crossover probability of BSC. In our analysis, we assume that the values and states of different variable nodes and check nodes are statistically independent. Hence, the bit error probability is the only input-output parameter of the proposed model, which can be applied to determine the probabilities of variable nodes in different values and states, thereby estimating FER.

In Fig. 2, $Q^{(k)}_{fv}$ and $Q^{(k)}_{uv}$ are defined as follows:

- $Q^{(k)}_{fv}$ denotes the probability of a codeword containing at least one variable node with false verification at the kth iteration, given that no variable nodes of the codeword are falsely verified in the first $k-1$ iterations.
- $Q^{(k)}_{uv}$ denotes the probability of a codeword containing at least one unverified variable node but no falsely verified variable node at the kth iteration, given that no variable nodes of the codeword are falsely verified in the first $k-1$ iterations.

Using $Q^{(k)}_{fv}$, $Q^{(k)}_{uv}$ and the law of total probability [12], the FER after running VA by m iterations is derived as follows:

$$\begin{aligned} FER^{(m)} =& Q^{(1)}_{fv} + Q^{(2)}_{fv}Q^{(1)}_{uv} + ... + \underbrace{Q^{(m)}_{fv}Q^{(m-1)}_{uv}...Q^{(1)}_{uv}}_{m\text{th iteration}} \\ &+ Q^{(m)}_{uv}Q^{(m-1)}_{uv}...Q^{(1)}_{uv} \\ =& Q^{(1)}_{fv} + \sum_{k=2}^{m} Q^{(k)}_{fv} \prod_{j=1}^{k-1} Q^{(j)}_{uv} + \prod_{k=1}^{m} Q^{(k)}_{uv}. \end{aligned} \tag{3}$$

In deriving (3), we use the fact that the FER can be calculated by accumulating the probability of a frame with false verification (*i.e.*, at least one variable node of the frame is with false verification) in each iteration and the probability of a frame with un-verification (*i.e.*, at least one variable node of the frame is with un-verification) after a maximum number of iterations.

To calculate $Q^{(k)}_{fv}$, let $\check{q}^{(k)}_{fv}$ denote the probability of a variable node that is falsely verified at *step 4* of the verification stage at the kth iteration. If a variable node is falsely verified, it fixes the incorrect value and state, which causes an error frame. Since the states and values of variable nodes of a codeword are assumed statistically independent, we can find that $Q^{(k)}_{fv} = 1 - \left(1 - \check{q}^{(k)}_{fv}\right)^n$.

Let $Q^{(k)}_{cv}$ denote the probability of a codeword that *all* its variable nodes are correctly verified in the verification stage or corrected in the correction stage at the kth iteration, given that the variable nodes of the codeword are not all correctly verified and no variable nodes are falsely verified in the first $k-1$ iterations. We can then, using $Q^{(k)}_{fv}$ and $Q^{(k)}_{cv}$, obtain that $Q^{(k)}_{uv} = \left(1 - Q^{(k)}_{fv}\right)\left(1 - Q^{(k)}_{cv}\right)$.

At the correction stage, let $\check{q}^{(k)}_{cor}$ denote the probability of a variable node that is corrected at the kth iteration. Let $\check{q}'^{(k)}_{cv}$ denote the proportion of correctly verified variable nodes among the variable nodes that are not falsely verified at *Step 1* of the correction stage of the kth iteration. We can then have, $Q^{(k)}_{cv} = \left(\check{q}^{(k)}_{cor} + \check{q}'^{(k)}_{cv}\right)^n$.

To run the iterative performance analysis algorithm of the statistical model, p_{k+1} needs to be found. Let p'_k denote the bit error probability among the codewords with no variable nodes falsely verified before the correction stage at the kth iteration. We can then obtain $p_{k+1} = p'_k\left(1 - \check{q}^{(k)}_{cor}\right)$.

From above discussion, to obtain FER, we need to calculate $\check{q}^{(k)}_{fv}$, $\check{q}'^{(k)}_{cv}$, $\check{q}^{(k)}_{cor}$ and p'_k, which are presented in the Appendix.

IV. Numerical Results

We take rate-1/2 regular-(3,6) LDPC codes as in [4] for an example. 50 codes are randomly generated. The codeword length is 1008 and the packet size of a variable node is 32 bits. Further, for comparison purpose, the numerical result of the VA without considering false verification is employed as a reference.

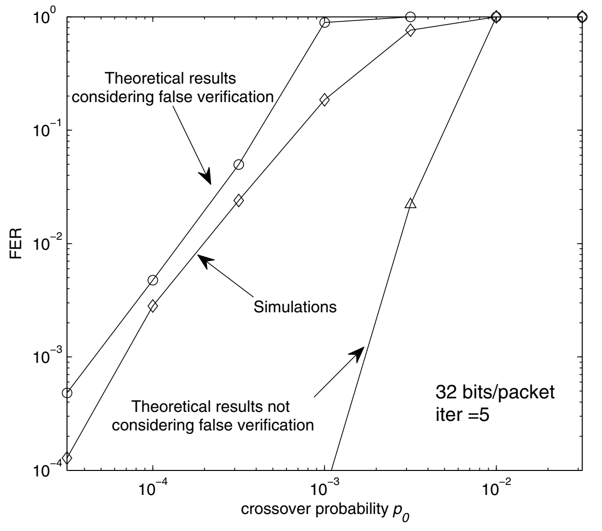

Fig. 3. Theoretical FER results considering and not considering false verification, compared with simulation results by averaging 50 random 1008-$(3, 6)$ LDPC codes for 5 iterations. The size of the packet is 32 bits.

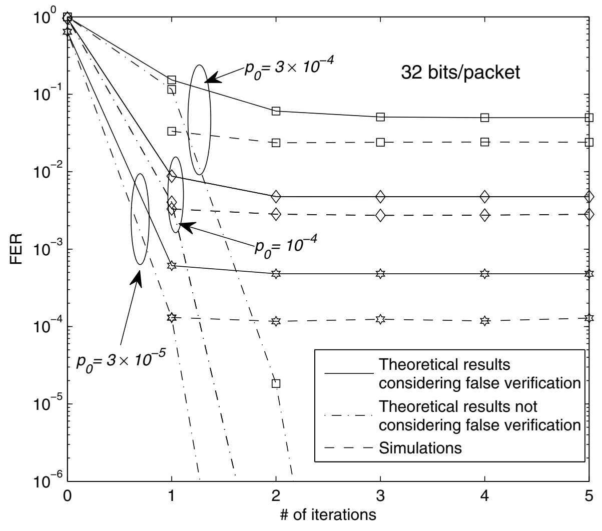

Fig. 4. Theoretical FER results considering and not considering false verification compared with simulation results by averaging 50 random 1008-$(3, 6)$ LDPC codes for different iterations. The size of the packet is 32 bits.

Fig. 3 shows the FER performance of the VA algorithm with 5 iterations. It can be found that the proposed statistical model provides a good approximation to the simulation results of verification-based decoding for packet-based LDPC codes over BSC channels. In contrast, without considering false verification, the theoretical results significantly deviate from the simulation results. For example, when the channel parameter is 3×10^{-4}, the numerical result without considering false verification is smaller than the simulation result by more than three orders, while the numerical result considering false verification is in the same order with the simulation result.

Fig. 4 shows the relationship between FER performance and the number of iterations of the VA algorithm. It can be found that the proposed model considering false verification gives good estimation of the simulation results for any number of iterations of the VA algorithm. In contrast, the theoretical results not considering false verification significantly deviate from the simulation results, especially when the number of iterations is increased. For example, when $p_0 = 10^{-4}$ and the number of iteration is 3, the FER of the proposed method is about 5×10^{-3} which is close to the simulation result, while the FER of the theoretical results without considering false verification is less than 10^{-6} which is far smaller than the simulation result.

V. Conclusions

In this paper, we have proposed a statistical model to analyze the verification-based algorithm for packet-based LDPC codes over BSC channels. The proposed model takes the false verification into consideration and provides an good estimate for FER at any number of iterations over all channel parameters of interest in practice. The numerical results and simulations demonstrate the validness of the proposed statistical model.

Appendix
Calculations of Probabilities $\check{q}_{fv}^{(k)}$, $\check{q}_{cv}^{\prime(k)}$, $\check{q}_{cor}^{(k)}$ and p_k'

In the following, we only consider the kth iteration for simplicity, thus omitting the superscript k. To find the probabilities $\check{q}_{fv}$, $\check{q}'_{cv}$, $\check{q}_{cor}$ and p'_k, we need to define the following probabilities and events:

- $\check{q}_{cv}$ denotes the probability of a variable node that is correctly verified at *step 4* of the verification stage.
- At *step 1* of the correction stage, q_{cv} and q_{fv} denote the probabilities that *messages*[2] from variable nodes to check nodes contain correctly verified and falsely verified values, respectively.
- q'_{cv} denotes the proportion of *messages* from variable nodes to check nodes that include correctly verified values among the messages that include no falsely verified values at *Step 1* of the correction stage.
- E_{v_t} is the event that the message from v to an adjacent check node contains a correct value at *Step 1* of the verification stage.
- $E_{v_f}^{\beta}$ is the event that the message from v to an adjacent check node contains an incorrect value with β error bits at *Step 1* of the verification stage.
- E_{ve}^{i} is the event that (1) is held for a degree-i check node, which indicates that a verification is achieved at *step 2* of the verification stage.

Using the above definitions, we can find that $P(E_{ve}^i|E_{v_t})$ is the conditional probability that (1) is held when the message

[2]We note here that in contrast to $\check{q}_{fv}^{(k)}$, which is defined from a node perspective, probabilities q_{cv} and q_{fv} are from an edge perspective.

from a variable node to the degree-i check node includes a correct value. At *step 2* of the verification stage, if (1) is held, a verified message is generated. As a result, the probability that the message from the degree-i check node to a variable node with a correct current value contains a verified state equals $P(E^i_{ve}|E_{v_t})$.

Let ρ_i denote the percentage of edges connected to degree-i check node. Let p_{cv} denote the probability that the message carries a verified state from a check node to a variable node with a correct current value. It can then be found that:

$$p_{cv} = \sum_i \rho_i P(E^i_{ve} \mid E_{v_t}). \tag{4}$$

Similarly, p^β_{fv} denotes the probability that the message carries a verified state from a check node to a variable node with β bit errors, which reads as follows

$$p^\beta_{fv} = \sum_i \rho_i P(E^i_{ve} \mid E^\beta_{v_f}). \tag{5}$$

The calculations of $P(E^i_{ve}|E_{v_t})$ and $P(E^i_{ve}|E^\beta_{v_f})$ can be solved by analyzing the error structures of the adjacent variable nodes for a degree-i check node. In the numerical results, the error structures can be found for the small number of error bits by using integer partition. The large number of error bits is not considered here, because it results in very small probabilities which can be ignored in the calculations.

At *step 4* of the verification stage, q_{cv} can be found using (4) as follows:

$$q_{cv} = (1-p_k)^l \left[1 - \sum_j \lambda_j \left(1-p_{cv}\right)^j\right] \tag{6}$$

where λ_j denotes the fraction of edges connected to degree-*j* variable nodes. Similarly, q_{fv} can be obtained using (5) as:

$$q_{fv} = \sum_{\beta=1}^{l} P(E^\beta_{v_f}) \left[1 - \sum_j \lambda_j \left(1-p^\beta_{fv}\right)^j\right] \tag{7}$$

As a result, $\check{q}_{cv}$ and $\check{q}_{fv}$ can be obtained as follows:

$$\check{q}_{cv} = (1-p_k)^l \left[1 - \sum_j L_j \left(1-p_{cv}\right)^j\right]. \tag{8}$$

and

$$\check{q}_{fv} = \sum_{\beta=1}^{l} P(E^\beta_{fv}) \left[1 - \sum_j L_j \left(1-p^\beta_{fv}\right)^j\right]. \tag{9}$$

where L_j is the fractions of degree-j variable nodes among all variable nodes.

q'_{cv} and $\check{q}'_{cv}$ can be obtained from q_{cv}, q_{fv}, $\check{q}_{cv}$ and $\check{q}_{fv}$ as follows:

$$q'_{cv} = \frac{q_{cv}}{1-q_{fv}} \quad and \quad \check{q}'_{cv} = \frac{\check{q}_{cv}}{1-\check{q}_{fv}}.$$

$\check{q}'_{uv}$ denotes the proportion of variable nodes that are unverified among the variable nodes that are correctly verified or unverified, which can be found as $\check{q}'_{uv} = 1 - \check{q}'_{cv}$.

Let $\check{q}_{uv\&v_f}$ denote the probability of an unverified variable node with incorrect current value, which can be found as follows:

$$\begin{aligned}\check{q}_{uv\&v_f} &= \check{q}'_{uv} \frac{1-(1-p_k)^l - \check{q}_{fv}}{1-\check{q}_{fv}-\check{q}_{cv}} \\ &= \frac{1-(1-p_k)^l - \check{q}_{fv}}{1-\check{q}_{fv}}.\end{aligned} \tag{10}$$

At the correction stage, the variable nodes are either unverified or correctly verified. Since the probability $\check{q}_{uv\&v_f}$ equals $1-(1-p'_k)^l$, p'_k can be derived from (10) as:

$$p'_k = 1 - \left(1 - \check{q}_{uv\&v_f}\right)^{1/l}. \tag{11}$$

Let p_{cor} denote the probability that the message from a check node to an unverified variable node with incorrect current value includes the correct value and verified state. It can be obtained as follows:

$$p_{cor} = \sum_i \rho_i \left(q'_{cv}\right)^{i-1}. \tag{12}$$

For a degree-j variable node, the unverified variable node with incorrect current value can be corrected if it receives one message that contains its correct value and verified state from neighboring check nodes. Therefore, by using (12), $\check{q}_{cor}$ can be found as follows:

$$\check{q}_{cor} = \check{q}_{uv\&v_f} \left[1 - \sum_j L_j \left(1-p_{cor}\right)^j\right]. \tag{13}$$

REFERENCES

[1] T. Richardson and R. Urbanke, “Efficient encoding of low-density parity-check codes,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.

[2] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. Spielman, “Improved low-density parity-check codes using irregular graph,” *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 585–598, Feb. 2001.

[3] R. G. Gallager, *Low Density Parity-Check Codes*. Cambridge MA: MIT Press, 1963.

[4] D. J. C. MacKay, “Good error correcting codes based on very sparse matrices,” *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[5] A. Shokrollahi, “Capacity-approaching codes on the q-ary symmetric channel for large q,” in *Proc. IEEE ITW ’04*, San Antonic, Texas, Oct. 24–29, 2004, pp. 204–208.

[6] M. Luby, “LT codes,” in *Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science (FOCS)*, Vancouver, BC, Canada, Nov. 2002, pp. 271–280.

[7] A. Shokrollahi, “Raptor codes,” *IEEE Trans. Inf. Theory*, vol. 14, no. 6, pp. 2551–2567, Jul. 2006.

[8] M. G. Luby and M. Mitzenmacher, “Verification-based decoding for packet-based low-density parity-check codes,” *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 120–127, Jan. 2005.

[9] B. Zhu, D. Huang, and S. Nordholm, “Enhanced verification-based decoding for packet-based ldpc codes,” *IEEE Commun. Lett.*, vol. 12, no. 2, pp. 136–138, Feb. 2008.

[10] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, Mar. 2008.

[11] S. Haykin and M. Moher, *Modern Wireless Communications*. Prentice Hall, Mar. 2004.

[12] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, Dec. 2001.