Enhanced Verification-Based Decoding for Packet-Based LDPC Codes

Bin Zhu, Defeng (David) Huang, *Senior Member, IEEE*, and Sven Nordholm, *Senior Member, IEEE*

Abstract—In this letter, an enhanced verification-based decoding algorithm (EVA) for packet level low-density parity-check (LDPC) codes is proposed. Compared with the verification algorithm (VA) in the literature, the proposed algorithm reduces the likelihood of false verification by enhancing the verification condition to achieve a better decoding performance. For example, simulation results in this letter show that EVA reduces frame error rate (FER) by two orders when comparing the VA over binary symmetric channel (BSC), while the increase of the computational load is less than 90%.

Index Terms—Binary symmetric channel (BSC), low-density parity-check (LDPC) codes, verification-based decoding.

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]–[8]. For packet erasure channel, the Fountain codes such as LT-codes [6] and Raptor codes [7] have been proposed to be used for realizing multicast 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*. In the verification stage, if the sum of all the neighboring variable nodes of a check node equals zero, all these variable nodes are verified. In 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. However, one limitation with VA is the way that the verification is done. There is a high likelihood of false verification because whether a variable node is verified or not only depends on one check node, which is not sufficient as will be shown in this letter by simulation results.

In this letter, we propose an *enhanced verification-based decoding algorithm* (EVA) for packet-based LDPC codes, which reduces the probability of false verification with only

moderate computation complexity increase. Unlike the VA, EVA verifies each variable node based on multiple incident check nodes, the number of which is declined from a preset value to one along with the decoding process. Compared with the VA algorithm, the verification of EVA is more reliable for variable nodes. As a result, the approach gives a better decoding performance. Further, the VA can be considered as a special case of EVA when the maximum number of check nodes required to be satisfied for a variable node is set to be one. Since EVA is a packet-based decoding algorithm, the *packet-level operations* are dominant in the decoding complexity. The proposed EVA maintains the total number of packet-level operations in the same order as VA. To achieve better trade-off between performance and complexity, a simplified enhanced verification algorithm has also been proposed.

This letter is organized as follows. In Section II, we describe the system model, present the proposed EVA algorithm and evaluate its computation complexity. In Section III, some simulation results for VA, EVA and simplified EVA are presented. Finally, this letter is concluded in Section IV.

II. ENHANCED VERIFICATION BASED DECODING ALGORITHM

A. System Model

Let $x := (x_1, x_2, \dots, x_n)$ be a codeword that is encoded with a LDPC parity check matrix H , where n is the length of the codeword and $x_i := [x_i^1, x_i^2, \dots, x_i^b]$ is a packet of a string of b bits. The codeword x is transmitted over a binary symmetric channel (BSC) with the crossover probability p in bit sequence. At the output of the channel, $y := (y_1, y_2, \dots, y_n)$ is obtained, where $y_i := [y_i^1, y_i^2, \dots, y_i^b]$.

The LDPC code can be represented by a factor graph (FG). The variable nodes represent the packets of the codeword. The check nodes determine the constraints, and the sums of their adjacent neighboring variable nodes equal zero, based on the parity check matrix H . Here the sum operation is a bitwise EXCLUSIVE-OR between packets. For regular LDPC codes, let d_c and d_v be the degrees of check nodes and variable nodes, respectively.

B. Proposed Algorithm

In the VA algorithm, a satisfied check node (i.e., the sum of its incident variable nodes equals zero) verifies all its incident variable nodes. As a result, there is a high likelihood that a variable node is incorrectly verified, thereby inducing a high frame error rate. In order to reduce the likelihood of such false verification of variable nodes, we enhance the verification requirement in the proposed EVA by requiring

Manuscript received October 11, 2007. The associate editor coordinating the review of this letter and approving it for publication was V. Stankovic. This research was supported in part under Australian Research Council's *Discovery Projects* funding scheme (project number DP0877616).

B. Zhu and S. Nordholm are with Western Australian Telecommunications Research Institute (WATRI), University of Western Australia, Crawley, WA, 6009 Australia (e-mail: {zhub, sven}@watri.org.au).

D. Huang is with the School of Electrical, Electronic and Computer Engineering, University of Western Australia, Crawley, WA, 6009 Australia (e-mail: huangdf@ee.uwa.edu.au).

Digital Object Identifier 10.1109/LCOMM.2008.071668.

multiple constraints be satisfied before a variable node can be claimed verified.

In principle, EVA can be implemented for both regular and irregular LDPC codes. For simplicity, we focus on regular LDPC codes. In the EVA algorithm, there are two possible states for variable nodes: either *verified* or *unverified*. Let C_{ver} and C_{unver} be the sets storing all the verified and unverified variable nodes, respectively. For each variable node v , there are three possible values: a final value f_v , a received value r_v and a current value c_v .

The complete procedure of EVA is listed at Algorithm 1. Initially the EVA will start with all variable nodes being unverified. Correspondingly, all current values of variable nodes equal their r_v s and these variable nodes are put into C_{unver} . In the following part, the EVA iteratively decodes the received codewords under different thresholds. There are two iterative stages named *verification* and *correction*. For each round i in the *verification* stage, there is a threshold value d such that at round i for each unverified variable node, if d incident check nodes satisfy the rule that the sum of the current values of their neighboring variable nodes equals zero, the corresponding variable node is verified and its current value is fixed as its final value, otherwise it remains unchanged both in state and value. As for the second stage *correction*, for each check node, if all its other connected variable nodes have been verified except one, this variable node is corrected and finalized by substituting the current value with the sum of others. In this way, these two stages are operated iteratively until no more unverified variable nodes in C_{unver} can be moved to C_{ver} under certain threshold value d or after a maximum of m iterations. In the next iteration, the threshold constraint is eased from d to $d-1$. The same procedure repeats again until the threshold value finally reduces to one. The VA can be viewed as EVA with the threshold d set to be constant one.

C. Computation Complexity

For packet-based LDPC codes, the complexity is dominated by the packet-level operations [9] especially when the number of bits in a packet is reasonably large. As listed in Algorithm 1, the packet-level calculations in the EVA include the d_c XOR calculations in (1) for verification and the d_c-1 XOR calculations in (2) for correction. At the first iteration, it is necessary to compute (1) for each check node. After that, we only need to re-compute (1) for the check nodes whose neighboring variable nodes' current values have been changed in the correction stage. Each correction requires the calculation of (2) once and the maximum number of correction operations using (2) equals the total number of unverified variable nodes after the verification step in the first iteration. As a result, the number of such computation operations is less than the codeword length n . Since the total re-computation times of (1) are dependent on the total number of variable nodes corrected in the correction stage, the number of re-computation operations using (1) is very limited when a large percentage of variable nodes can be verified at the verification step in the first iteration. As a result, compared with VA, the extra computation of the proposed algorithm is reasonably low when

Algorithm 1 Enhanced Verification-Based Decoding Algorithm

Initialization: Put all variable nodes into C_{unver} and make their current values equal their received values, set maximum iteration number m .

for $d = d_v$ to 1 **do**

$n = 0$

$state = false$

while $state = false$ **and** $n \leq m$ **do**

$n = n + 1$

Step 1: verification

Check each check node c whether

$$\sum_{v_k \in V_c} c_{v_k} = 0 \quad (1)$$

where V_c is the set of neighboring variable nodes linked with check node c .

For each variable node v in C_{unver} , count the number of incident check nodes satisfying (1), and denote the number by num_c .

If $num_c \geq d$, then variable node v is *verified*, $f_v = c_v = r_v$, remove variable node v into C_{ver} .

Step 2: correction

For each check node c ,

check all its neighboring variable nodes.

If all the neighbor variable nodes of check node c are verified except only one variable node v_l , then v_l is *corrected* and *verified* as:

$$f_{v_l} = c_{v_l} = \sum_{v_j \in \Omega, v_j \neq v_l} c_{v_j} \quad (2)$$

where Ω is the set of the verified variable nodes incident to check node c .

Remove variable node v_l from C_{unver} into C_{ver} .

if the number of variable nodes moved into C_{ver} in **Step 2** is equal to zero, **then**

$state = true$

end if

end while

end for

the channel parameter p is small. We note here that in the above analysis, extra process control is required. The extra process control only requires bit-level calculations that are in low complexity and thus has little influence on the total computation load compared with packet-level calculations.

We can further reduce the complexity of EVA by reducing the number of thresholds required. For example, instead of decreasing threshold from d_v , we can use a preset value x less than d_v as the upper threshold. This reduced complexity algorithm is called simplified enhanced verification-based algorithm. When the maximum threshold value is x , we denote it as EVA-x. Therefore, VA can be indicated as EVA-1.

III. SIMULATION RESULTS

Throughout the simulation, we use the (1008, 504) regular LDPC code [10] with $d_c = 6$ and $d_v = 3$, as an example.

Fig. 1. FER for different packet sizes using VA and EVA. The solid lines and dashed lines denote the performance of VA and EVA, respectively.

Fig. 2. Comparison of different decoding algorithms with 64 bits per packet.

The number of m is preset to 5, unless otherwise indicated. Moreover, the *Frame Error Rate* (FER) is employed to indicate the decoding performance.

Fig. 1 shows the decoding performance among the different packet sizes ranging from 32 bits to 256 bits. Compared with VA, it can be seen that the proposed EVA gives better performance for any packet size. Further, a small packet size gives a small FER for both VA and EVA. This is because a large packet size induces a large packet error probability, thereby resulting in a high FER.

In Fig. 2, we compare the performance among VA, EVA and EVA-2 for packet size of 64 bits. EVA exhibits the best performance and both EVA and EVA-2 give better performance than VA, which is more than one order better in FER when $p \leq 10^{-3}$. Further, VA exhibits similar performance for $m = 5$ and $m = 50$. From the figure, it is also found that EVA-2 is a good option when channel parameter p ranges from 10^{-2} to 10^{-3} due to its similar performance to EVA but low computation complexity.

In Fig. 3, the numerical results indicate the average numbers

Fig. 3. Ratio of average numbers of packet-level XOR operations in EVA-2 and EVA to that of VA for decoding one frame, and number of average packet-level XOR operations in VA.

of packet-level XOR operations for decoding one frame in VA, EVA-2 and EVA for a packet size of 64 bits. It can be found that the complexity of both EVA and EVA-2 are comparable to that of VA when either $m = 5$ or $m = 50$. The computation complexity of EVA increases by no more than 90%, compared with VA.

IV. CONCLUSION

In this letter, we have presented an enhanced verification-based decoding algorithm for packet-based LDPC codes, which reduces the likelihood of false verification by enhancing the verification condition. With moderate computation complexity increase, the proposed algorithm has outperformed the VA in more than 100 times better FER results when $p \leq 10^{-3}$ based on simulations.

REFERENCES

- [1] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. 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. Inform. 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. Inform. 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 Antonio, 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. Inform. Theory*, vol. 14, no. 6, pp. 2551–2567, July 2006.
- [8] M. G. Luby and M. Mitzenmacher, "Verification-based decoding for packet-based low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 120–127, Jan. 2005.
- [9] Y. Han and W. E. Ryan, "Packet-ldpc codes for tape drives," *IEEE Trans. Magn.*, vol. 41, no. 4, pp. 1340–1347, Apr. 2005.
- [10] D. J. C. MacKay, *LDPC code database*, On the internet at <http://www.inference.phy.cam.ac.uk/mackay/codes/data.html>.