On Combining Chase-2 and Sum-Product Algorithms for LDPC Codes

Sheng Tong and Huijuan Zheng

This letter investigates the combination of the Chase-2 and sum-product (SP) algorithms for low-density parity-check (LDPC) codes. A simple modification of the tanh rule for check node update is given, which incorporates test error patterns (TEPs) used in the Chase algorithm into SP decoding of LDPC codes. Moreover, a simple yet effective approach is proposed to construct TEPs for dealing with decoding failures with low-weight syndromes. Simulation results show that the proposed algorithm is effective in improving both the waterfall and error floor performance of LDPC codes.

Keywords: LDPC codes, Chase algorithm, sum-product (SP) algorithm.

I. Introduction

The Chase algorithm [1] is a suboptimal decoding procedure and usually used along with a hard decision decoder to improve the decoding performance at the cost of increased computational complexity. There are three variants of the Chase algorithm [1], among which the Chase-2 algorithm is the most promising and provides a good tradeoff between performance and complexity.

In this letter, rather than combining the Chase-2 algorithm with a hard decoding algorithm, we consider the association of the Chase-2 algorithm with a soft decision decoding algorithm, that is, the sum-product (SP) algorithm [2], for the decoding of low-density parity-check (LDPC) codes [3]. The purpose is to provide an approach to achieve a tradeoff between performance and complexity for LDPC decoding.

Manuscript received Dec. 3, 2011; revised Feb. 21, 2012, accepted Mar. 7, 2012.

This work was supported by the 973 Program (Nos. 2012CB316100 and 2010CB328300), NSFC (Nos. 61001130, 60972046, and 61101148), and the Fundamental Research Funds for the Central Universities (Nos. K50510010026 and K50510010006).

Sheng Tong (phone: +86 029 158 0925 0930, ts_xd@163.com) is with the State Key Lab. of ISN, Xidian University, Xi'an, China.

Huijuan Zheng (zhj@xupt.edu.cn) is with the Department of Telecommunication, Xi'an University of Posts and Telecommunications, Xi'an, China.

http://dx.doi.org/10.4218/etrij.12.0211.0510

II. Combining Chase and Sum-Product Algorithms

For an (n, k, d) binary linear code, the Chase-2 algorithm selects p least reliable positions (LRPs) in the received sequence and constructs a set of 2^p test error patterns (TEPs) whose non-zero elements are confined in the p LRPs. Then, each of the 2^p TEPs are added to the hard decision of the received sequence and the sum vector is fed into a hard decision decoder. Finally, all decoded code words, known as candidate code words, are compared and the one with the best metric with respect to the received sequence is chosen as the final decoding output. The parameter p determines the complexity, which is originally set to be $[d/2]$ [1]. By varying p, tradeoffs between performance and complexity can be made.

The SP algorithm is a suboptimal iterative decoding algorithm for the decoding of LDPC codes, which consists of two steps, that is, the variable node update (VNU) and check node update (CNU). The details of the SP algorithm are provided by [2]. As VNU is relatively simple and irrelevant to the following development, we only revisit the well-known tanh rule [4] for CNU. Consider a check node c. Denote the set of all of its neighboring variable nodes (VNs) as $N(c)$. The incoming log-likelihood ratio (LLR) message from a VN $v \in N(c)$ to c is denoted as $m_{v\to c}$, and the outgoing LLR message from c to v is denoted as $m_{c\to v}$. Then, the tanh rule is given by

$$\tanh\left(\frac{m_{c\to v}}{2}\right) = \prod_{w\in N(c), w\neq v} \tanh\left(\frac{m_{w\to c}}{2}\right). \quad (1)$$

When combining the Chase-2 and SP algorithms, one needs to incorporate TEPs into the SP decoding of LDPC codes. This

can easily be realized by modifying the tanh rule. To describe this modification, some notations are required. Denote as $D(v)$ ($\in\{0, 1\}$) the hard decision of a coded bit or equivalently its corresponding VN, v. The set of VNs corresponding to the non-zero elements in a TEP is denoted as V. Then, the modified tanh rule can be written as

$$\tanh\left(\frac{m_{c\to v}}{2}\right)=\prod_{u\in N(c)\cap V}(-1)^{D(u)}\prod_{w\in N(c)\setminus V,w\neq v}\tanh\left(\frac{m_{w\to c}}{2}\right). \quad (2)$$

The reason for the above modification is explained as follows. When the hard decision is made for a non-zero position, or a VN w, in a TEP, its associated LLR L_w is $+\infty$ for $D(w)=0$ or $-\infty$ for $D(w)=1$. Note that $\tanh(L_w/2)$ is 1 for $L_w=+\infty$ or -1 for $L_w=-\infty$, that is, $\tanh(L_w/2)=(-1)^{D(w)}$. Equation (2) follows directly from (1) by replacing $\tanh(L_w/2)$ with $(-1)^{D(w)}$ for $w\in T$. Thus, a TEP is naturally incorporated into the SP decoding of LDPC codes. The resulting algorithm is denoted as Chase-SP(p), where p is the number of LRPs in the Chase-2 algorithm.

Some remarks about the application of the above Chase-SP algorithm are made as follows. The Chase-SP algorithm can be used after an SP decoder. Only when a decoding failure occurs, that is, the former SP decoder fails to output a valid LDPC code word, will the Chase-SP decoding be invoked. This is reasonable since a well-designed LDPC code typically possesses no undetectable errors. Once a valid LDPC code word is produced, it is almost guaranteed that the output code word is really the transmitted one. Following the same reason, the Chase-SP decoder stops to process the remaining TEPs once a valid code word is generated. This strategy is different from the conventional practice of the Chase algorithm, where all TEPs are required to be processed. By using the above two tricks, the total complexity can be greatly reduced when using the Chase-SP algorithm and, in fact, is comparable to that of SP decoding in the high signal-to-noise ratio (SNR) region, as will be shown in the following simulations.

To investigate the performance of Chase-SP decoders, a length-504 (3, 6) regular LDPC code is used. Assume an additive white Gaussian noise (AWGN) channel with binary phase-shift keying (BPSK) modulation. At $E_b/N_0=2.6$ dB, 2×10^5 code words are simulated using SP decoding with a maximum iteration number of 100 and 681 decoding failures are observed, among which 197 failures can be recovered by Chase-SP(3), accounting for 28.9 percent of all failures. We also observe that most of the recoverable failures by Chase-SP(3) have high weight syndromes (say, a few dozen). Most failures with syndrome weights no greater than six (that is, six unsatisfied checks at most), accounting for 30.2 percent of the remaining ones, cannot be recovered by Chase-SP(3) decoding. This implies that LRP-based TEPs are useful for some failures with high syndrome weights but not for those with low syndrome weights. If we could devise effective TEPs for treating failures with low-weight syndromes and recover a majority of them, the error rate could be greatly reduced. The information that follows presents a simple approach to generating effective TEPs for treating decoding failures with low-weight syndromes with the help of syndromes.

III. Syndrome-Based Test Error Patterns

Denote as $W(S)$ the Hamming weight of a syndrome S. For a small $W(S)$, we can construct a relatively small number of TEPs as follows. After an unsuccessful SP decoding, we obtain the hard decision code word c'. Note that each 1 in S corresponds to an unsatisfied check, in which there is at least one bit in error joining. Thus, for each unsatisfied check, choose one of its neighboring VNs and flip the corresponding bit in c'. In this way, for a (j, k) regular LDPC code, we can obtain $k^{W(S)}$ TEPs. Among these TEPs, there must exist a TEP that can correct $W(S)$ errors, which is expected to be helpful in recovering all of the remaining errors. To distinguish from LRP-based TEPs, the constructed TEPs are referred to as syndrome-based TEPs. To reduce the number of TEPs, one can use only a few, say q, of the $W(S)$ unsatisfied checks, thus leading to a total number of k^q TEPs. To decide whether to use the LRP-based TEPs or syndrome-based TEPs, we set a threshold, T, for $W(S)$. When $W(S)>T$, the LRP-based TEPs are used. Otherwise, syndrome-based TEPs are adopted. The resulting algorithm is referred to as Chase-SP(p, T, q).

The reason why the Chase-SP(p) algorithm does not work well for failures with low-weight syndromes is intuitively explained as follows. Failures with low-weight syndromes are closely related to concepts such as near code words [5] or trapping sets (TSs) [6], which dominate the error floor performance. Unless Chase-SP(p) decoding happens to select some bits in TSs, the TSs will cause decoding deadlocks. However, a TS usually involves a few bits and thus Chase-SP(p) decoding has a relatively small probability of choosing bits from TSs, especially from small-size ones. In contrast, with syndrome-based TEPs, bits involved in a TS are revealed by the syndrome to some extent. By flipping a few bits in a TS, the decoding deadlock caused by the TS is expected to be broken. This is true for many instances. However, as observed in our experiments, for some stubborn TSs, which are combinations of two or more smaller TSs, the above flipping method may not work well. Even so, the syndrome-based TEP approach is effective in dealing with a large portion of TSs, thus resulting in improved error performance.

According to the above explanation, the threshold parameter, T, should be chosen according to the sizes of the dominant TSs.

Usually, a larger threshold is expected to offer a better performance at the cost of a higher decoding complexity. Thus, threshold should be carefully chosen to balance the performance and complexity.

To ease the understanding of the proposed decoding algorithm, a flow chart description of the Chase-SP(p, T, q) decoding procedure is shown in Fig. 1, where I and I_{MAX} are the iteration number and the maximum iteration number, respectively. From Fig. 1, we see that there are two decoding phases in the Chase-SP decoding. For a given received code word, the conventional SP decoding phase is firstly used, and then the Chase decoding phase is invoked if and only if the conventional SP decoding phase fails to output a valid code word. During the Chase decoding phase, according to the syndrome weight $W(S)$ and the given threshold T, either syndrome-based TEPs or LRP-based TEPs are used. For any given TEP, a modified SP decoding is called, in which the CNU is run by using (2) instead of (1). When the two decoding phases are finished, I records the total number of the SP iterations used in both the SP decoding and Chase decoding phases. For simplicity, we neglect the complexity involved in constructing TEPs, which are marginal compared to an SP iteration. Thus, when Chase decoding is invoked, compared to the conventional SP decoder, the increased decoding complexity can be roughly assessed as the number of SP iterations used in the Chase decoding phase, which can be easily calculated as (I–I_{MAX}). In the following section, we simply use the average iteration numbers to compare the complexities of different algorithms. For the Chase-SP decoding algorithm, the average iteration number is calculated as the ratio of the sum of the Is of all the received code words and the number of received code words.

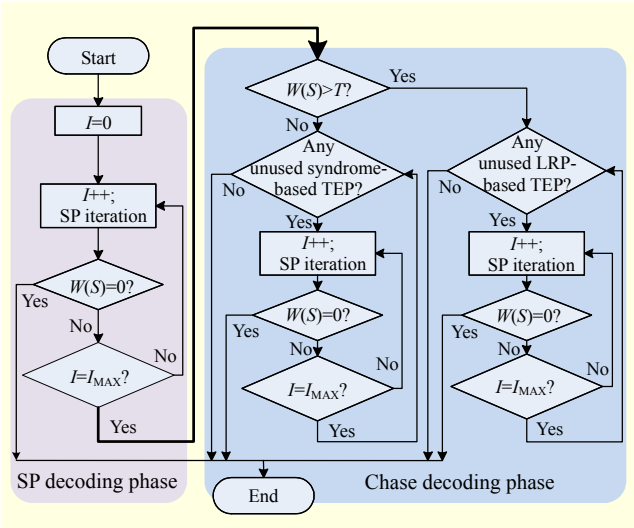

Fig. 1. Flow chart of Chase-SP(p, T, q) decoding algorithm. S denotes syndrome. I and I_{MAX} stand for iteration number and the maximum iteration number, respectively. $W(S)$ is Hamming weight of S. T is threshold, which is non-negative integer.

IV. Simulation Results

The same rate-1/2, length-504 (3, 6) regular LDPC code is used in this study. The performance of the Chase-SP algorithm with four different parameter settings is simulated and compared with that of the SP decoding. From Fig. 2, we see that Chase-SP algorithms outperform SP decoding in both the waterfall and error floor regions. In the parameter setting (3, 0, –), the threshold, T, is set to be 0. Thus, syndrome-based TEPs are not used, and only LRP-based TEPs are treated. Note that the parameter q is useless in this setting and thus is not

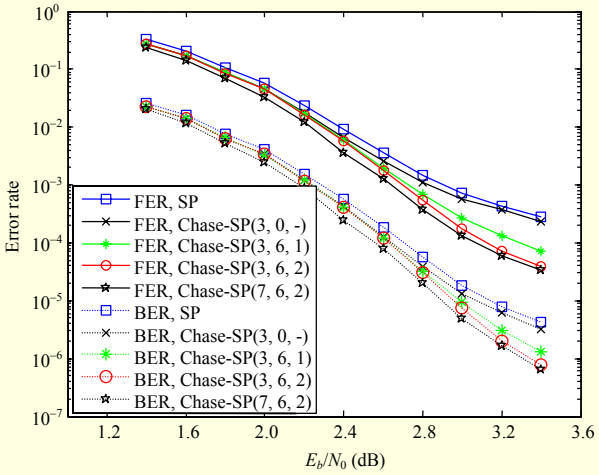

Fig. 2. Performance of length-504 (3, 6) regular LDPC codes used over AWGN channel with BPSK modulation.

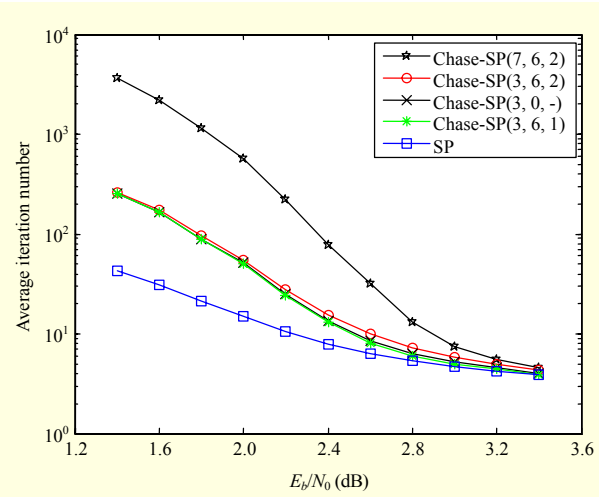

Fig. 3. Average iteration numbers vs. E_b/N_0.

provided here. Comparing the performance curves of Chase-SP(3, 0, –) and Chase-SP(3, 6, 1), we can see that syndrome-based TEPs are especially useful in improving the error floor performance. From Fig. 2, we also see that Chase-SP(3, 6, 1) performs similarly to Chase-SP(3, 6, 2) in the waterfall region, while Chase-SP(3, 6, 2) outperforms Chase-SP(3, 6, 1) in the error floor region. This implies that increasing q in Chase-SP(p, T, q) can improve the error floor performance but not the waterfall performance. However, by increasing p from 3 to 7, we see observable performance improvement in both the waterfall and error floor regions, which implies that p affects the performance of both the waterfall and error floor.

Moreover, the complexities of both algorithms are also compared in terms of average iteration number, as shown in Fig. 3. We see that for high SNRs, the increases in the average iteration number for Chase-SP algorithms are marginal when compared to the SP algorithm.

V. Conclusion

In this letter, we investigated the combination of the Chase-2 and SP algorithms. An effective approach to constructing TEPs for treating errors with low-weight syndromes was presented. Simulation results showed that the proposed Chase-SP algorithm provides a good tradeoff between performance and complexity.

References

[1] D. Chase, "Class of Algorithms for Decoding Block Codes with Channel Measurement Information," *IEEE Trans. Inf. Theory*, vol. 18, Jan. 1972, pp. 170-182.

[2] F.R. Kschischang, B.J. Frey, and H. Andrea Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, Feb. 2001, pp. 498-519.

[3] R.G. Gallager, *Low-Density Parity-Check Codes*, PhD dissertation, MIT, Cambridge, MA, USA, July 1963.

[4] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 3, Mar. 1996, pp. 429-445.

[5] D. MacKay and M.S. Postol, "Weaknesses of Margulis and Ramanujan-Margulis Low-Density Parity-Check Codes," *Electron. Notes Theoretical Computer Sci.*, vol. 74, 2003.

[6] T.J. Richardson, "Error Floors of LDPC Codes," *Proc. 41st Annual Allerton Conf. Commun., Control, Computing*, Monticello, IL, USA, Sept. 2003, pp. 1426-1435.