International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:7, No:4, 2013

Enhancing the Error-Correcting Performance of LDPC Codes through an Efficient Use of Decoding Iterations

Insah Bhurtah, P. Clarel Catherine, and K. M. Sunjiv Soyjaudah

Abstract**—**The decoding of Low-Density Parity-Check (LDPC) codes is operated over a redundant structure known as the bipartite graph, meaning that the full set of bit nodes is not absolutely necessary for decoder convergence. In 2008, Soyjaudah and Catherine designed a recovery algorithm for LDPC codes based on this assumption and showed that the error-correcting performance of their codes outperformed conventional LDPC Codes. In this work, the use of the recovery algorithm is further explored to test the performance of LDPC codes while the number of iterations is progressively increased. For experiments conducted with small blocklengths of up to 800 bits and number of iterations of up to 2000, the results interestingly demonstrate that contrary to conventional wisdom, the error-correcting performance keeps increasing with increasing number of iterations.

Keywords**—** Error-correcting Codes, Information Theory, Low-Density Parity-Check Codes, Sum-Product Algorithm.

I. Introduction

LOW-DENSITY parity-check (LDPC) codes form a class of error-correcting codes originally developed by Gallager in 1962 [1]. In 1999, MacKay and Neal effectively rediscovered the codes and demonstrated their ability to perform at rates close to the Shannon's theoretical limit [2].

LDPC codes are usually identified by a parity-check matrix **H** containing mostly '0's and very few '1's [1]. Decoding is achieved using the belief propagation algorithm also known as the message passing algorithm or the sum-product algorithm (SPA). For large blocklengths, the performance of LDPC codes is known to operate very close to the Shannon limit [3]. However, when it comes to short blocklengths, sub-graph structures of the bipartite graph such as cycles and trapping sets decrease the performance of these codes [4]. As a means of dealing with the effects of these structures, a recovery technique for LDPC codes was introduced by Soyjaudah and Catherine in 2008 [5], resulting in an important coding gain over the conventional algorithm. For the present work, the recovery algorithm is further explored with different **H** sizes for an increasing number of iterations. The aim of the research is to demonstrate that the recovery algorithm exhibits very good performance with increasing iterations for the decoding of LDPC codes. Indeed, whereas using additional iterations does not usually help for conventional algorithm (with the decoder stuck in bad configurations from which it cannot escape), the same iteration resources are used more effectively in the recovery algorithm thereby bringing the dividends of an increased error-correcting capability.

The paper is organized as follows: in section II, an introduction to the recovery algorithm is provided. Section III explains the motivation of the present work and section IV highlights the further work done on the recovery algorithm. Results and the conclusion are presented in section V and VI respectively.

Insah Bhurtah is an MPhil/PhD student at Electrical and Electronic Engineering Department, University of Mauritius, Mauritius (e-mail: i.bhurtah@ieee.org).

P. Clarel Catherine is a lecturer at Industrial Systems Engineering, School of Innovative Technologies and Engineering (SITE), University of Technology, Mauritius (e-mail: c.catherine@ieee.org).

K. M. Sunjiv Soyjaudah is a professor at Electrical and Electronic Engineering Department, University of Mauritius, Mauritius (e-mail: ssoyjaudah@uom.ac.mu).

II. The Recovery Algorithm

Under the conventional decoding regime of LDPC codes, the decoder will usually move in a certain direction during the first few iterations [6] and keep that direction until it will either converge to the correct state or reach a wrong state. Alternately, it can also oscillate between two states without any definite convergence. As such, the first iterations of the decoding algorithm are usually critical and provide the general direction the decoder will take. Subsequent iterations thus cause a waste of valuable computing resource. To combat this problem, the redundant set-up of the bipartite graph as described in [5] is established.

The redundancy of the graph allows the use of a subset of nodes during decoding while the other nodes wait for the decoding metrics to be communicated to them to start their processing. Hence, a percentage of the nodes are erased. The deletion of the bit nodes enables the decoder to withdraw itself from a certain decoding path and to choose another one. This approach allows the decoder to regard other paths not present in the conventional SPA. Instead of running the algorithm for only one subset of the bipartite graph, multiple subsets are used. This allows the algorithm to get different outlooks of the graph. They may agree or disagree with each other, but they allow the decoder to calculate the reliability of each node. Another reason for the employment of multiple subsets is to account for the fact that good nodes might also be erased. A more detailed analysis of the recovery algorithm can be found in [5].

III. Motivation

In a conventional decoder, when the decoder has started in a certain wrong direction, it is difficult for it to retract from its path. Hence, the number of iterations used becomes meaningless. The erasure of nodes allows the decoder to withdraw from a direction and choose another. This allows the destruction of cycles and trapping sets which contribute to bad

structures of the bipartite graph. In [5], the recovery algorithm outperforms the conventional one and consumes fewer iterations. Therefore, the recovery algorithm is further tested with different **H** matrices and parameters to show its efficiency. The parameters are chosen at random to demonstrate the capability of the recovery algorithm, though their specific selection remains an open problem. The investigation whether iterations used by the recovery algorithm are being used more efficiently, is undertaken. Consequently, it can be known whether computing power is being used more efficiently.

Another aspect which has been looked into is the entropy variation of the decoded vector for each iteration of the recovery algorithm where the conventional one has failed. When one or more nodes of the bipartite graph are in error, they will convey the same wrong information to the other nodes to which they are connected. These receiving nodes will increase their false confidence, and they will in turn provide other nodes with wrong messages eventually leading to decoding failure. In addition to this, the use of many iterations does not serve any purpose, as the decoder goes into an oscillatory condition [6]. Conventional LDPC decoders had trouble from escaping the bad bipartite graph configurations due to the low entropy (high confidence) of its bit nodes. When decoding is proceeding in the wrong direction, high confidence of the bit nodes will ultimately lead to disaster. Indeed increasing the number of iterations usually does not help much since the decoder has already reached a state of low-entropy from which it is very difficult to back off. However, when the recovery algorithm is used, the entropy of the different sets used is initially raised (since entropy of all erased bits is 0.5) [7]. On that account, the decoders are again given a chance for correct decoding. It appears therefore that entropy plays a central role in the decoding of LDPC codes. Assuming that n is the number of bits in the decoded vector (X), the equation for the calculation of its entropy in terms of each bit x_i, $H(X)$ is [8]:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \quad (1)$$

where p is the a-posteriori probability of each bit x_i.

To obtain the variation of the entropy for a sample of unsuccessful conventional decoding and for each of the subsets n_s for every iteration, the following notations for the parameters were used:

- n_s : number of sets used.
- I_c: number of iterations with all bit nodes used with I_c = 100 iterations.
- I_r: number of iterations for each set of the recovery algorithm with I_r = 20 iterations.
- I_C: number of iterations for a conventional SPA run such that $I_C = I_r.n_s + I_c$ [7].

The procedure is found below for a particular block number where the conventional decoding fails and the recovery algorithm comes into play:

1) For the conventional SPA algorithm with I_c iterations, the number of bits of the codeword equals to the number of columns n of the **H** matrix. For each iteration, the entropy of the decoded vector is obtained according to (1). During decoding, the a-posteriori probability p of each bit of the decoded vector is initially generated and updated after every iteration. In this work, I_C = 200, 500 and 2000 have been used.
2) For each subset of the recovery algorithm (1 to n_s), the entropy of the decoded vector is obtained according to (1) for each iteration with I_r set to 20.

Fig. 1 illustrates the variation of the conventional and for each of the n_s = 5 subsets for a 50x100 **H** matrix with I_C = 200 and (E_b/N_o) = 1 dB for a particular transmitted block. For conventional decoding, the entropy oscillates, that is the decoder does not which way to go. A similar result was obtained in [6]. Regarding the recovery algorithm, the entropy of the set number 3 decreases smoothly and withdraws the right codeword. In table I, sets number 62, 67 and 78 have retrieved the right codeword in 95 subsets for a 200x400 **H** matrix with I_C = 2000 and (E_b/N_o) = 3 dB for a particular transmitted block. These sets have consumed fewer iterations than the conventional algorithm (I_c = 100 iterations) leading to an efficient use of computing power for decoding. In addition, the final entropy of the conventional algorithm is 16.2 as compared to sets number 62, 67 and 78 having entropy of 1.62, 4.23 and 13.99 respectively. It was mentioned earlier that in the case of decoding failure, the low-entropy state (high confidence of bit nodes) prevents the decoder to back off. Fig. 2 illustrates this type of behaviour from the failed conventional algorithm whose entropy falls to a low value that is the decoder has a high degree of belief in a bad state.

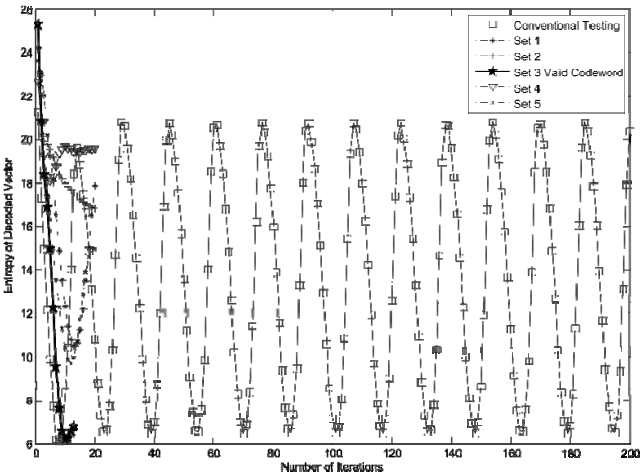

Fig. 1 Graph of Entropy of Decoded Vector against Number of Iterations For 50x100 **H** Matrix I_C = 200, n_s = 5 with (E_b/N_o) = 1 dB

IV. Simulation Work

Depending on the number of subsets used, the same number of bipartite graphs were created. Some nodes were erased while others participated in the decoding process. δ denotes

TABLE I
ENTROPY VARIATION FOR SUBSETS GIVING A VALID CODEWORD FOR A 200X400 **H** MATRIX WITH $I_C = 2000$, $n_s = 95$, $I_r = 20$ AND $(E_b/N_o) = 3$ dB

Iteration #	Set#62	Set#67	Set#78
1	55.96	57.18	54
2	50.98	51.43	49.67
3	52.83	52.33	53.61
4	49.7	52.13	52.13
5	45.46	48.66	51.25
6	45.12	45.97	50.61
7	46.47	38.02	49.53
8	45.32	30.83	45.74
9	40.87	26.85	40.59
10	34.23	23.78	35.6
11	29.58	22.96	31.6
12	23.11	20.19	28.41
13	18.6	15.94	22.13
14	17.33	9.9	13.99
15	14.73	4.23	
16	12.46		
17	11.14		
18	9.88		
19	5.09		
20	1.62		

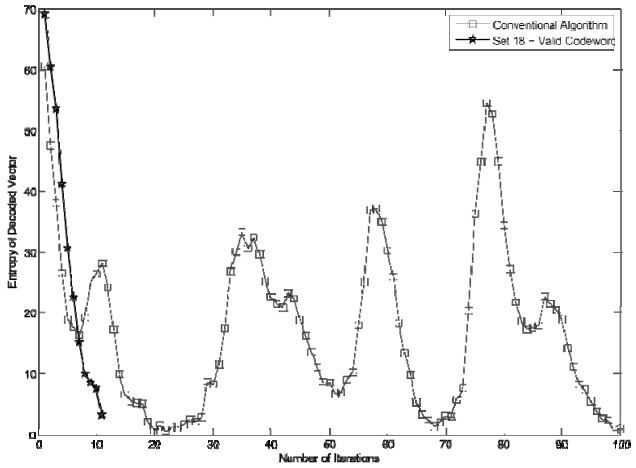

Fig. 2 Graph of Entropy of Decoded Vector against Number of Iterations for 200x400 H matrix $I_C = 500$, $n_s = 20$ with $(E_b/N_o) = 2$ dB

the ratio of participating bit nodes. δ was set to 0.95 meaning 95% of the nodes participated while 5% were erased.

At first, I_c = 100 iterations were run by the conventional SPA where no node was erased. If a valid codeword was obtained, the process was stopped and success was declared. However, in case of conventional decoding failure, it was run for the different n_s bipartite graphs with specific erased nodes for each, for a small number of iterations, I_r = 20 iterations. Any valid codeword retrieved from the sets was stored. The codeword closest to the received vector is chosen as the recovered one and success is declared. If no valid codewords were obtained, the process was stopped denoting failure.

In a conventional SPA, the number of iterations set do not usually serve any purpose especially if the decoder has chosen a wrong path. The aim of this work was to show that a performance gain could be obtained with increasing iterations by the use of the recovery algorithm which partitions the iterations into n_s bipartite graphs having δ erased nodes. Very good performance is obtained when I_C is increased and with very few iterations I_r of the subsets, as will be demonstrated in section V.

V. RESULTS

Results for the Block Error Rate (BER) against (E_b/N_o) are shown with **H** matrices of different sizes such as m = 200 n = 400 and m = 400 n = 800 where m denotes the number of rows (size of source bits) and n, the number of columns (size of codeword). For codewords of length n = 400 in Fig. 3, a coding gain of around 0.7 dB was reported for a BER of 1.33×10^{-5} with I_C = 2000 iterations. For codewords of length n = 800 with a BER of 4.94×10^{-6}, a coding gain of 0.6 dB was obtained with I_C = 2000 iterations as shown in Fig. 4. For 200x400 and 400x800 matrices, it can be clearly observed that the conventional codes show the presence of an error floor whereas the recovery ones tend to converge. Hence, the BER performance increases with iterations for the recovery algorithm as opposed to the conventional one. The use of more subsets enables better BER performance. Fig. 5 further confirms the capability of the recovery algorithm to provide better error-correction with an increase in iterations. One eventual application of the recovery algorithm could be in space and interplanetary communications, most essentially in Delay-Tolerant Networks (DTNs) where nodes usually have to wait a long time for a communication link to be available for the transfer of messages (bundles) to their destination [9]. For that reason, the recovery algorithm is best suited to act on a node by making an efficient use of iterations and providing better error-correction capability thus solving problems associated with high error rates. A further work could be an appropriate design of the DTN architecture to incorporate the recovery algorithm.

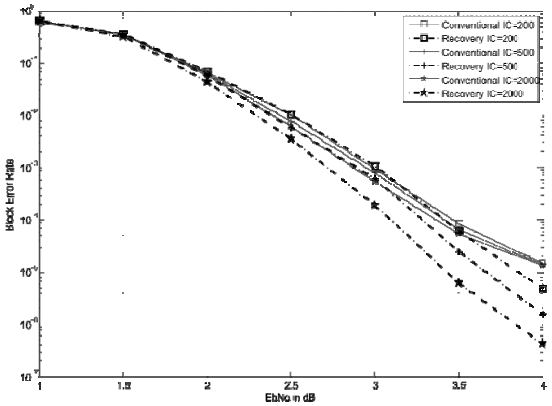

Fig. 3 Graph of Block Error Rate against (E_b/N_o) for **H** matrix m = 200 n = 400

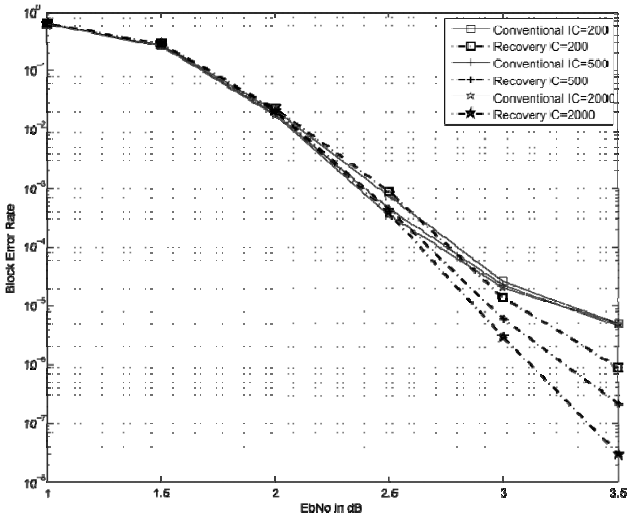

Fig. 4 Graph of Block Error Rate against (E_b/N_o) for **H** matrix m = 400 n = 800

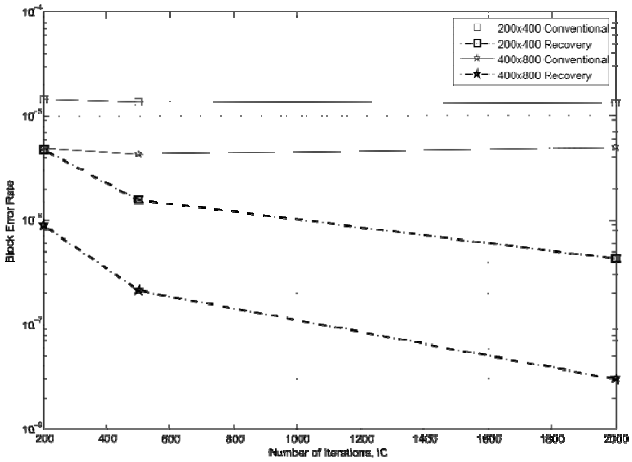

Fig. 5 Graph of Block Error Rate against Number of Iterations, I_C for 200x400 **H** Matrix (E_b/N_o) = 4 dB and 400x800 **H** Matrix (E_b/N_o) = 3.5 dB

VI. Conclusion

The advantage of the recovery algorithm is that better performance is obtained with increasing iterations (computing resources used more efficiently) compared to the conventional algorithm where the number of iterations do not help in successful decoding. Analysis of the entropy of the codeword has also shown that conventional schemes are inefficient in terms of iteration use. Subsets giving the correct codeword also tend to use fewer iterations. Future work may involve the design and testing of a DTN architecture incorporating the recovery algorithm for error-correction in intermittent connectivities.

Acknowledgment

The financial support of the Tertiary Education Commission of Mauritius is gratefully acknowledged.

References

[1] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[2] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431,March 1999.

[3] T. J. Richardson and R.L. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[4] O.H. Kazanci, "Performance of pseudo-random and quasi-cyclic low density parity-check codes," Master's thesis, The Graduate School of Natural and Applied Sciences of Middle East Technical University, Turkey, 2007.

[5] K.M.S. Soyjaudah and P.C. Catherine, "Efficient recovery technique for low-density parity-check codes using reduced-set decoding," *Journal of Circuits, Systems, and Computers (JCSC)*, vol. 17, no. 2, pp. 333–351, Apr. 2008.

[6] M.C. Davey, "Error-correction using low-density parity-check codes," Ph.D. dissertation, University of Cambridge, United Kingdom, 1999.

[7] P.C. Catherine and K. M. S. Soyjaudah, "Erasing bit nodes on the bipartite graph for enhanced performance of LDPC codes," *4th International Conference on Emerging Trends in Engineering and Technology (ICETET)*, pp. 107–111, Nov. 2011.

[8] C.E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[9] C. Caini, H. Cruickshank, S. Farrell, and M. Marchese, "Delay- and disruption-tolerant networking (DTN): an alternative solution for future satellite networking applications," *Proceedings of the IEEE*, vol. 99, no. 11, pp. 1980–1997, Nov. 2011.