ENHANCING LDPC CODE PERFORMANCE USING PILOT BITS

Jack F. Adolph, Jan C. Olivier, Brian P. Salmon

School of Engineering, University of Tasmania, Australia

ABSTRACT

Creating the optimal bipartite graph and decoding of a Low Density Parity Check (LDPC) codeword on it is deemed an NP-complete problem. Much work in the last decade has gone into proper construction of LDPC codes with maximum girth and optimized stopping sets; to ensure the BER approaches channel capacity. When an edge alteration to the graph is proposed, it completely changes the performance of the graph and usually leads to the re-analysis of the entire graph's properties. In this work we propose a method of lowering the error floor experienced in an LDPC code by intelligently inserting a set of known bits in the message frame. This deactivates paths in the graph located around trapping sets without modification of edges in the graph, and provides high log-likelihood information to the induced sub-graph for improved performance.

Index Terms— Channel coding, Linear codes, Parity check codes, Iterative decoding

1. INTRODUCTION

The proper design of a LDPC code is crucial in ensuring transmission with bit error rates that are near the Shannon limit. Chung et al. showed how a half rate LDPC code can operate within 0.0045 dB of the Shannon limit for a binary input additive white Gaussian noise channel [1]. These capacity approaching codes are obtained by finding code ensembles that have near Shannon noise thresholds. The noise threshold for a particular code ensemble is computed using an algorithm such as *density evolution* [2]. These types of algorithms work on the pair of degree distributions that completely describes a particular code ensemble and dictates the distribution of the connections used in the bipartite graph. Once the bipartite graph is constructed from the code ensemble, a Belief Propagation (BP) algorithm can be used to rapidly estimate the received encoded bits [3].

This is all based on the assumption that long block lengths are used in the code and that the bipartite graph has a tree structure. This in turn allows active trails among all variables to ensure optimal messages are passed among different variables nodes. However, the construction of these bipartite graphs is not perfect and many impairments are introduced into the code. One such impairment is an observable error floor at high Signal-to-Noise Ratios (SNRs) which is typical for codes operating near the Shannon limit.

Several strategies have been devised to lower or remove these error floors without affecting the LDPC's ability to operate near capacity [4, 5, 6]. In [7] a decoding method is presented to split the bipartite graph into clusters in an attempt to isolate trapping sets. Another approach is to create a lookup table of the most dominant trapping sets and use a bit flipping approach to solve that sub-graph [8]. In this paper we propose to strategically embed a minimal number of known (pilot) bits inside the message frame before encoding, with the idea that the position of these pilot bits will positively affect the messages passed by the belief propagation to dramatically improve impairments in the graph which should result in the lowering of the error floor. The placement of these pilot bits has the effect of deactivating trails within the graph to provide high reliable log-likelihood probabilities to certain parts of the graph and eliminate the effect of trapping sets and cycles. Trapping sets and cycles are believed to contribute significantly towards the observed error floors in LDPC codes. The advantage of the pilot bits are that they are punctured before transmission and will only have a small effect on the code rate.

The question of where to place these pilot bits, and how many pilot bits to use is addressed in this paper. In [9], several properties on trapping sets are listed and were used to determine positions of the pilots bits. Results are presented showing that the error floors at high SNR are reduced by utilizing well placed pilots.

The paper is organized as follows. Section 2 we discuss the concepts of cycles and trapping sets. In section 3 we present our method for insert pilot bits into a graph, and section 4 present our experimental results. Section 5 presents the conclusions.

2. CYCLES AND TRAPPING SETS

LDPC codes allow for practical decoding of long codewords with near Maximum Likelihood (ML) performance. However, reaching the noise threshold with a finite length code for a particular code ensemble given that the decoding is NP-complete is a daunting task. Given this scenario, the decoding of the codeword has a probability, given the distribution

Email: {jack.adolph,jc.olivier,bp.salmon}@utas.edu.au

of the erroneous log-likelihood ratios, of getting trapped in cycles which causes a decrease in the rate of reduction in the BER for an increase in SNR which ultimately results in an observable error floor.

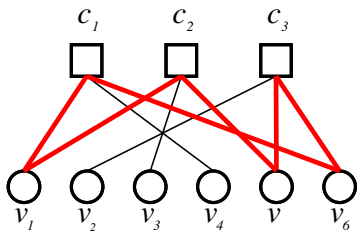

Fig. 1. An example of a 6-cycle in a bipartite graph

One cause of an error floor is when cycles are present in the graph. A cycle of length l is defined as an active path in a graph that consists of l unique edges that moves through the graph and ends on the same node of the graph where it started. An example of a length 6 cycle for a LDPC code with 6 variable nodes and 3 check nodes is shown in Figure 1. It is known that belief propagation passing messages on a graph with a tree structure will approximate the true aposteriori probabilities of the variable nodes, but the conditions for convergence in a graph with loops are still not well understood [10]. Even if tests for convergence are used, such as EXIT charts, it still does not verify that aposterior probabilities are correct. A known strategy for mitigating loops is to increase the girth, where the girth of a code is the length of the shortest cycle in the graph [11].

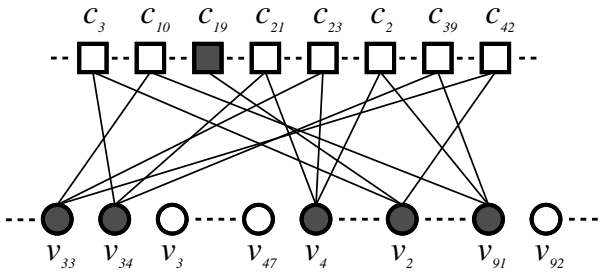

Fig. 2. A (5,1) trapping set of the 96.33.964 code

A more prominent cause of the error floor is *near codewords*, which are referred to as trapping sets. These are sets in a graph with a relative small number of variable nodes such that the induced sub-graph has a small number of odd degree check nodes. These create the lower bound to the error floor. An (a, b) trapping set is defined as a decoder state with a bit errors and b unsatisfied parity checks [9]. The more dominant trapping sets are generally combinations of short cycles with lower values for a and b and high ratio of $\frac{a}{b}$ [12]. An example of a (5,1) trapping set is shown in Figure 2. Here, the shaded variable nodes represent bit errors, and the shaded check nodes represent unsatisfied parity checks.

Another cause of the error floor is when the decoder chooses a valid but wrong codeword. In this case all parity checks are satisfied, yet the codeword produced is wrong, and hence $b = 0$ in this case. This does not often happen when the block length is large, but in this paper we will provide results for LDPC codes with smaller block length where this does happen, and we show that the pilot bits will mitigate this problem as well as the trapping sets.

3. IMPROVED DECODING USING PILOT BITS

Pilot bits are bits which position and value are known to both the receiver and transmitter. They are traditionally used in communication systems for a number of purposes, including synchronization and channel estimation. We propose in this paper the placing of pilot bits within the message frame that will be located within the known trapping sets and cycles after encoding. As everything about these bits are known they are punctured before transmission assuming the code is systematic. When the message is received at the decoder, they are re-inserted as high log-likelihood soft bits into their respective positions.

Using a Tanner graph representation of a (a, b) trapping set we follow the notation of [13] where a $\bullet$ represent a variable node, and a square represent an even degree check node (and a filled square an odd degree check node). Figure 3 shows the sub-graph induced by a (4,4) trapping set $T = v_1, v_2, v_3, v_4$ in a column-weight-three LDPC code. This sub-graph is a trapping set as check nodes $C_1, C_2, \cdots, C_4$ are satisfied if the variable nodes v_1, v_2, v_3, v_4 are either zero or one. No message passing decoding strategy will be able to resolve the variable nodes under these conditions. However if any one of the variable nodes are known with a high probability (pilot bit), it enables high reliability messages for the belief propagation when computing the aposterior probabilities in Figure 3 which will result in the variable nodes being correctly estimated; this process we call pilot bit insertion and it effectively *disables* the trapping set. In the case of Figure 3 the use of one pilot bit at any one of the variable nodes v_1, v_2, v_3, v_4 will disable this particular trapping set. Some trapping sets will require more than one bit to be disabled, but in general only a few pilot bits in a frame will be able to disable most of the dominating trapping sets.

The drawback however is the effective code rate is reduced as the same number of parity check bits are generated with less message bits. The effective code rate is $R_p = \frac{k-p}{n-p}$, $k, p, n \in \mathbb{N}$, where n denotes the length of codeword, k the length of the message and p the number of pilot bits. The effective code rate dictates that the number of pilot bits p, should be kept relatively small compared to k and n. As message passing algorithms pass probabilistic message, the value of the pilot bits are of no concern and deemed independent

for the analysis. However, as p needs to be kept as small as possible, the position of these p pilots needs to be optimized. It was found that the trapping sets in optimized codes with large girths tend to cluster together, which assist in keeping the number of pilot bits p low [9].

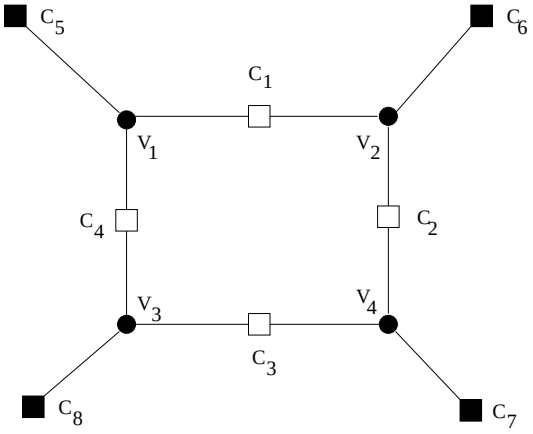

Fig. 3. The Tanner sub-graph representation of a (4,4) trapping set.

3.1. PROPOSED ALGORITHM

An algorithm is given here that attempts to find the optimal locations for the pilot bits for a particular LDPC code based on empirical analysis of the most dominant trapping sets. The proposed algorithm is explained below:

1. Search and identify trapping sets in a graph [13, 14, 15, 16, 17].

2. Compute the Hamming distance d between each trapping set and the all zero codeword.

3. Compute the contribution that each trapping set makes to the BER for a particular SNR γ using [14]

$$P_\epsilon(\text{trapping set}) \approx Q\left(\sqrt{2.d^2\gamma}\right). \quad (1)$$

4. Create a histogram of the summation of the contribution of errors per bit in the code, i.e.

$$f(i) = \sum_{\text{trap. sets}} \{P_\epsilon | i \in P_\epsilon\}. \quad (2)$$

5. Insert a random orientated pilot into the bit position i with the highest value.

6. Remove all trapping sets from the pool that contain this pilot bit.

7. Repeat steps 3-6 until the desired amount of pilot bits is reached.

An approximation of the reduction in BER can be computed by computing the difference of $\sum P_\epsilon$ for before and after that pilot bits.

4. SIMULATION RESULTS

In our experiments we initially investigated the use of pilot bits to increase the girth of a code, by inserting pilots into the shortest cycles and also positions were several cycles of different lengths cross. This would deactivate the cycles' active path in the graph. This approach however offered no real improvement on well design codes.

The next approach was used to target dominant trapping sets in the code. By using the proposed algorithm in section 3.1, we successfully eliminate the targeted trapping sets which in turn reduced the error floor.

The proposed algorithm was tested on three different LDPC codes[1] shown in table 1, each with relative short block code length. This explains why some decoding errors are attributed to wrong codewords being produced by the BP decoder [2]. LDPC codes with long block code length will not have this issue, and the error floor under those conditions can be attributed to trapping sets only.

Table 1. The three different LDPC codes used in our experiments.

Code name	n (bits)	k (bits)	Code rate
96.33.964	96	48	0.5
PEGReg252x504	504	252	0.5
PEGirReg252x504	504	252	0.5

The algorithm estimated the maximum number of pilots needed and the position of each pilot. This is summarized along with the new effective code rate in table 2.

Table 2. Pilot bit locations selected by the proposed algorithm

Code name	Pilot Bit positions	Effective code-rate	#Pilots (p)
96.33.964	{91, 55, 79}	0.484	3
PEGReg252x504	{493, 503, 473, 486}	0.496	4
PEGirReg252x504	{279, 352, 374, 258}	0.496	4

A descending list of trapping sets (a,b) and wrong codeword errors for both with and without pilot bits according to their respective percentage of failures is shown in table 3. The

[1] Available from http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[2] By wrong codewords we mean a valid decoded codeword, that is incorrect and result in errors at the receiver.

96.33.964 LDPC code at an E_b/N_0 of 6.5dB was used. It was observed that pilots mitigated the high failure rate on the dominant (6,0) source of error for this particular code.

Table 3. The trapping set distribution and incorrectly decoded codewords of the 96.33.964 LDPC code on a AWGN channel

Source of errors (No pilots)	%failures	Source of errors (3 pilots)	%failures
(6,0)	66.0	(5,1)	22.8
(5,1)	20.6	(7,1)	21.5
(8,0)	5.1	(6,2)	19.0
(7,1)	3.1	(8,0)	12.7
(4,2)	2.1	(4,2)	7.6
(8,2)	2.1	(8,2)	6.3
(10,0)	1.0	(9,1)	3.7
		(11,1)	2.5
		(9,3)	1.3

As stated previously the main trade-off was the decrease in the effective code rate. There is a point where loss in throughput by adding more pilot bits do not justify the gain in the BER. However significant gain in BER performance was achieved by using only a few pilot bits, which resulted in a negligible reduction in code rate. To ensure that a fair comparison was made for the slight differences in code rates, the standard deviation of the additive white Gaussian noise was scaled accordingly for the effective code rate and was plotted on a E_b/N_0 axis.

The BER performance in the AWGN channel for the three codes running with and without pilot bits are shown in figure 4. The maximum number of message passing iterations allowed in the belief propagation was set to 150.

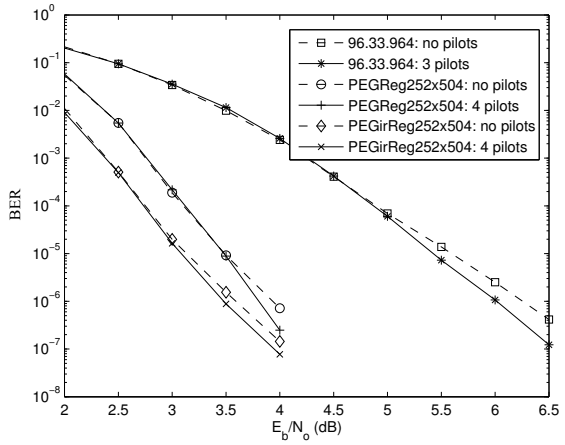

Fig. 4. Performance of the three LDPC codes with and without the assistance of the pilot bits in an AWGN channel.

The BER performance in a frequency non-selective fading channel was investigated using the fading model proposed in [18]. The channel was again scaled for the effective code rate as in the AWGN case for fair comparison.

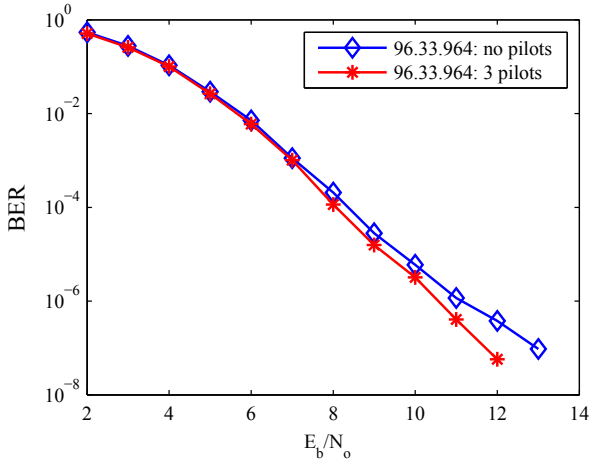

Fig. 5. Performance of the 96.33.964 LDPC in a frequency non-selective fading channel.

In both experiments shown in figure 4 and 5 it can be seen that the noise threshold on the codes do not improve, but at higher SNRs the error floor is visibly lower.

5. CONCLUSIONS

In this work we showed that the error floor could be effectively lowered by the insertion of just a few pilot bits. The pilot bits are used to deactivate paths in the bipartite graph and providing high reliable likelihood probabilities to the trapping sets found in the induced sub-graphs. The other advantage of this approach is that there is no change in the bipartite graph structure and only requires minimal modification at the encoder and decoder. The trade off however is the loss in effective code rate, which restrains the total number of pilot bits that can be inserted. We proposed an algorithm that can be used to determine the position and number of pilot bits and showed empirical simulations to show that the gain in lowering the error floor is large enough to support the loss in effective code rate. The method can also be applied to LDPCs with relative short block lengths ($n < 100$), but more pilots can be used with codes using larger block lengths. The method is effective when used with well designed bipartite graphs as the number of pilots must be kept to a minimum and will be less effective for codes with large number of trapping sets.

6. REFERENCES

[1] S. Chung *et al.*, "On the design of low-density parity-check codes within 0.0045 dB of the shannon limit," *IEEE Communication letters*, vol. 5, pp. 58–60, 2001.

[2] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on information theory*, vol. 47, pp. 599–618, 2000.

[3] J. Pearl, "Reverend Bayes on inference engines: A distributed hierarchical approach," in *Proceedings of the Second National Conference on Artificial Intelligence*, 1982, pp. 133–136.

[4] T. Tao *et al.*, "Construction of irregular LDPC codes with low error floors," in *IEEE International Conference on Communications, ICC 2003*, 2003, pp. 3125–3219.

[5] H. Xiao-Yu, E. Eleftheriou, and D. Arnold, "Progressive edge-growth tanner graphs," in *IEEE Global Telecommunications Conference, GLOBECOM*, 2001, pp. 995–1001.

[6] W. Wen-Yen, A. Ramamoorthy, and R. Wesel, "Lowering the error floors of irregular high-rate LDPC codes by graph conditioning," in *IEEE Conference in Vehicular Technology, VTC2004-Fall*, 2004, pp. 2549–2553.

[7] J. Sibel, S. Reynal, and D. Declercq, "Generalized belief propagation to break trapping sets in LDPC codes," in *Communications Theory Workshop*, 2014, pp. 132–137.

[8] E. Cavus and B. Daneshrad, "A performance improvement and error floor avoidance technique for belief propagation decoding of LDPC codes," in *IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, 2005, pp. 2386–2390.

[9] T. Richardson, "Error floors of LDPC codes," in *Proc. of the 41st Allerton Conf.*, 2003, pp. 1–10.

[10] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural computation*, vol. 12, pp. 1–41, 2000.

[11] M. O'Sullivan, "Algebraic construction of sparse matrices with large girth," *IEEE Transactions on information theory*, vol. 52, pp. 718–727, 2006.

[12] T. Tian *et al.*, "Construction of irregular LDPC codes with low error floors," in *IEEE International Conference in Communications*, 2003, pp. 3125–3129.

[13] B. Vasic, S. Chilappagari, D. Nguyen, and S. Planjery, "Trapping set ontology," in *IEEE Forty-Seventh Annual Allerton Conference Allerton House, UIUC, Illinois, USA September 30 - October 2, 2009*, 2009.

[14] C. Cole, S. Wilson, E. Hall, and T. Giallorenzi, "A general method for finding low error rates of ldpc codes," 2006. [Online]. Available: eprintarXiv: cs/0605051

[15] M. Karimi and A. H. Banihashemi, "Efficient algorithm for finding dominant trapping sets of ldpc codes," *Information Theory, IEEE Transactions on*, vol. 58, no. 11, pp. 6942–6958, 2012.

[16] S. Abu-Surra, D. Declercq, D. Divsalar, and W. E. Ryan, "Trapping set enumerators for specific ldpc codes," in *Information Theory and Applications Workshop (ITA), 2010*, 2010, Conference Proceedings, pp. 1–5.

[17] W. Chih-Chun, "On the exhaustion and elimination of trapping sets: Algorithms & the suppressing effect," in *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, 2007, Conference Proceedings, pp. 2271–2275.

[18] C. Xiao, Y. Zheng, and N. Beaulieu, "Novel sum-of-sinusoids simulation models for rayleigh and rician fading channels," *Wireless Communications, IEEE Transactions on*, vol. 5, no. 12, pp. 3667–3679, December 2006.