INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume3, Issue3)
Available online at www.ijariit.com

A Novel Approach for Efficient LDPC Decoding

B. LasyaSri
HAL
sujatha.podili@gmail.com

B. Ashok Raj
HAL
ashokaraja29@gmail.com

L. Nirmala Devi
Osmania University, Telangana, India
nagiitkgp@yahoo.co.in

Abstract: ***Recently, LDPC codes have been proved to be capable of closely approaching the channel capacity and hence they are strongly promoted in many international standards. Though decoding is simpler compared to Turbo code decoding, efficient decoding techniques for LDPC codes are in demand to make it further simpler. The theoretical performance of LDPC codes appears simple and close to Shannon's capacity of the channel, but the practical implementation is still challenging due to tasks like numerical inaccuracy, limited memory resources, etc. We investigate methods to reduce the complexity and make decoding simple while still keeping the loss in performance negligible.***

The aim of this paper is to reduce the complexity by simplification of the component decoders. For this, we analyze the min-sum algorithm and derive a theoretical framework. With this framework, we are able to modify the algorithm in order to achieve good performance for regular as well as irregular LDPC codes.

Keywords: LDPC Codes, Processing, Decoding, Algorithm, Tanner Graph.

I. INTRODUCTION

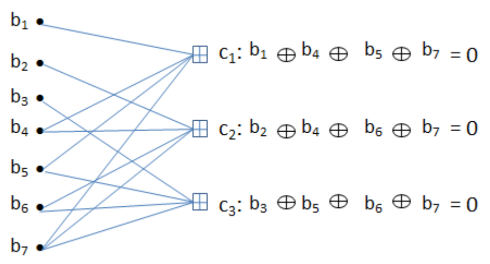
LDPC codes popularly known as Gaggler codes have been introduced by Gaggler in 1962 but these codes were largely ignored until the invention of Turbo codes [1]. The high decoding complexity made the application at that time impossible. When LDPC codes have been reinvented in the late 1990s by Mackay and Neil and by Wiberg in 1996, the throughput demands have been moderate.

Any linear block code can be defined by its parity-check matrix. If this matrix is sparse, i.e. it contains only a small number of 1s per row or column, and then the code is called a low-density parity-check code. The sparsity of the parity-check matrix is a key property of this class of codes. If the parity-check matrix is sparse, an efficient iterative decoding algorithm can be applied.

II. LDPC CODES

LDPC codes are usually defined in terms of a sparse bipartite graph, which is called Tanner graph. It is an undirected graph consist of two types of nodes, message nodes, and checks nodes. Fig-1 provides the Tanner graph of the Hamming code B (7, 4, 3) where an equivalent Hamming code is used. The n(7) nodes on the left are the message nodes. These nodes are associated with the n symbols b1.....bn of a code word. The r(3) nodes c1.....cr on the right are called check nodes and represent parity-check equations.

The entry h_{ji} of the parity-check matrix H is 1 if and only if the jth check node is connected to the ith message node. Consequently, the jth row of the parity check matrix determines the connections of the check node cj, i.e. cj is connected to all message nodes corresponding to 1s in the jth row. We call those nodes the neighbourhood of cj. The neighbourhood of cj is represented by the set Pj = {i : hji = 1}. Similarly, the 1s in the ith column of the parity-check determine the connections of the message node bi. We call all check nodes that are connected to bi the neighbourhood of bi and denote it by the set Mi = {j : hji = 1}.


b_1
b_2
b_3
b_4
b_5
b_6
b_7
$c_1: b_1 \oplus b_4 \oplus b_5 \oplus b_7 = 0$
$c_2: b_2 \oplus b_4 \oplus b_6 \oplus b_7 = 0$
$c_3: b_3 \oplus b_5 \oplus b_6 \oplus b_7 = 0$

$$H = \begin{bmatrix} 1001101 \\ 0101011 \\ 0010111 \end{bmatrix}$$

Fig-1: Tanner Graph and Sparse Matrix

A Tanner graph where all message nodes have the same degree and all check nodes have the same degree results in a regular code. In an irregular code, the different message nodes have different degrees (different numbers of connected edges).

III.LDPC DECODING

There exists a class of Decoding algorithms that are all iterative procedures where, at each round of the algorithm, messages are passed from message nodes to check nodes and from check nodes back to message nodes. Therefore, these algorithms are called message-passing algorithms. One important message-passing algorithm is the Belief Propagation algorithm i.e. called Sum of Product algorithm. The messages passed along the edges in the belief propagation algorithm are log-likelihood values. Each round of the algorithm consists of two steps. In the first half-iteration, a message is sent from each message node to all neighboring check nodes. In the second half-iteration, each check node sends a message to the neighbouring message nodes.

Let L_l[bi → cj] denote the message from a message node 'bi' to a check node 'cj' in the 'l' th iteration. This message is computed on the basis of the observed channel value 'ri' and some of the messages received from the neighbouring check nodes except 'cj' according to the (1). It is an important aspect of belief propagation that the message sent from a message node 'bi' to a check node 'cj' must not take into account the message sent in the previous round from 'cj' to 'bi'. Therefore, the message is explicitly excluded in the following (1). 'Mi' denotes the neighbourhood of the node 'bi'.

Messages from message nodes to check nodes:

$$L_l[b_i \to c_j] = \begin{cases} L(r_i) & \text{if } l = 1 \\ L(r_i) + \sum_{j' \in M_i \setminus \{j\}} L_{l-1}[c_{j'} \to b_i], & \text{if } l > 1 \end{cases} \qquad \text{.......................................} (1)$$

The message Ll [cj → bi] from the check node 'cj' to the message node 'bi' is computed as

$$L_l[c_j \to b_i] = Z[c_j \to b_i] = 2\tanh^{-1}\left(\prod_{j' \in P_j \setminus \{i\}} \tanh \frac{L_l[b_i \to c_j]}{2} \right) \qquad \text{................................} (2)$$

From (1), It is understood that the operations at the message node decoder are easy to implement. However, the implementation of the check node decoder (2) is less trivial because of the following reasons [2]:

a. Nonlinearity:

The implementation of the hyperbolic tangent and its inverse is usually realized with a lookup table. In order compute many check nodes in parallel, many lookup tables have to be implemented which increases the number of required resources significantly.

b. Multiplication:

The algorithm consists of real-valued multiplications which require many resources on dedicated hardware.

c. Stability:

When the magnitude of the argument of the inverse hyperbolic tangent is close to one, the inverse operation becomes numerically unstable.

To avoid the above-said complexity, we propose a framework which consists of approximation, Post processing.

IV.POST PROCESS

Decoding of single parity-check codes can be approximated by using the min-sum algorithm (MSA) which approximates as per Eq (3).

$$L_l[c_j \to b_i] = \min_{j' \in P_j \setminus \{i\}} L_l[b_i \to c_j] \cdot \prod_{j' \in P_j \setminus \{i\}} sgn(L_l[b_i \to c_j]) \qquad \text{................................} (3)$$

As per (3), there are no nonlinear functions involved that require a lookup table and a real-valued multiplication is not required anymore (the multiplication can be realized using Ex-OR operations). While the min-sum algorithm not only lowers the implementation complexity, it also leads to a decoding algorithm where a multiplicative factor at the input, i.e. scaling all channel L-values by a constant, results only in a scaling of all messages passed in the graph by this factor.

However, the advantage of reduced complexity and elimination of the noise estimation has to be paid by a degradation of the decoding threshold of approximately 0.5 to 1.0dB for regular codes. For irregular codes, the loss in performance is even higher. It has been recognized that this performance degradation can be partly compensated by transforming the check node output of the min-sum algorithm. The correction terms that result in the lowest decoding threshold have been found using density evolution and it has been shown that the thresholds of the resulting algorithms are close to that of the sum-product algorithm. However, when applying these methods to irregular LDPC codes, it has been observed that these codes exhibit an error floor which is not caused by the properties of the code, but by the decoding algorithm.
First, we consider the case of computing an outgoing message of a check node using the sum-product algorithm, where the term message denotes a log likelihood ratio. Given that dc binary variables X1.......Xdc satisfies a parity-check equation, we aim to compute the L-value of one binary variable given the L-values of the dc − 1 other variables, where the received L-values are the outputs of binary input, symmetric, memory fewer channels with capacity Iac,i2. Without loss of generality, we will compute 'Ldc' as a function of L1 , . . . , Ldc −1as (4) .

$$L\left(L_1,\ldots,L_{d_c-1}|X_{d_c}\right)=2\tanh^{-1}\prod_{i=1}^{d_c-1}\tanh\frac{L_i}{2}.$$
.... (4)

For the min-sum approximation, we can state a similar problem depicted in Fig-2. After the min-sum algorithm, we have to compute the L-value of the desired variable given the output of the check node approximation denoted as Eq (5)

$$Z\left(L_1,\ldots,L_{d_c-1}\right)=\min_{i=1}^{d_c-1}|L_i|\cdot\prod_{i=1}^{d_c-1}\operatorname{sgn}\left(L_i\right).$$
.... (5)

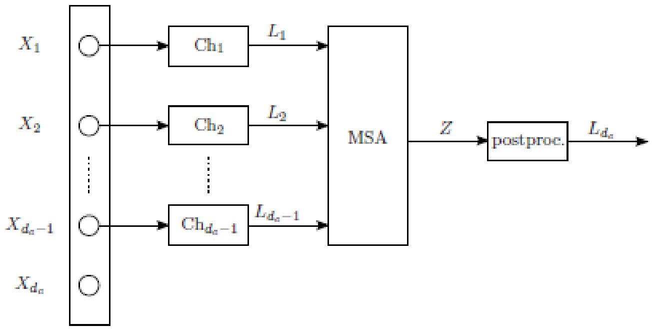


Fig-2: Model for check Node Operation using the Min-Sum Algorithm.

We are still able to analytically compute the L-value as a function of z, using the conditional probability density function

$$L(z|X_{d_c})=\log\frac{p(Z=z|X_{d_c}=+1)}{p(Z=z|X_{d_c}=-1)}$$
.... (6)

$$\begin{aligned}p(Z=z|X_{d_c}=x_{d_c}) &= \frac{\partial}{\partial z}Pr(Z<z|X_{d_c}=x_{d_c})\\ &= \frac{1}{2}\sum_{j=1}^{d_c-1}\Bigg\{(q_j(z)+q_j(-z))\prod_{i=1;i\neq j}^{d_c-1}\gamma_{i+}(z)\\ &\quad + x_{d_c}(q_j(z)-q_j(-z))\prod_{i=1;i\neq j}^{d_c-1}\gamma_{i-}(z)\Bigg\}.\end{aligned}$$
.... (7)

Although the L-value of the desired message can be computed analytically, the computation requires the knowledge of the distributions of all incoming messages.

V. REGULAR LDPC CODES

In this section, we investigate how the proposed post-processing function derived can be simplified for regular LDPC codes. We approximate using Constant Linear and Offset Processing and the results are compared.

1. Constant Linear Post-Processing

The resulting post-processing function is nonlinear and the dependency on the mutual information requires that it has to be recomputed in every iteration. This function is approximated by a linear function, where the approximation takes the probability density function of the minimum Z into account. We define α(dc , Iac) as the scaling factor which minimizes the expected squared error.

The correction terms α (in the case of Linear Processing) depends on *Iac*, which requires a new value for every iteration in the decoding process.

$$\alpha(d_c, I_{ac}) = argmn \int_{-\infty}^{\infty} [f(z; d_c, I_{ac} - \propto^{\sim} z]^2 \; p(Z = z;\, d_c, I_{ac}) dz, \ldots\ldots\ldots\ldots\ldots\ldots \quad (8)$$

2. Offset Processing:

The offset approximation can be combined with the linear approximation of the previous section. However, the resulting gain in performance is negligible. Similar to α, β also depends on *Iac*.

$$\beta(d_c, I_{ac}) = argmin_{\beta} \int_{-\infty}^{\infty} [f(z; d_c, I_{ac}) - sgn(z).\max(|z| - \beta^{\sim}, 0)]^2 \; p(Z = z;\, d_c, I_{ac}) dz, \ldots\ldots\ldots \quad (9)$$

We can further simplify the post-processing function by choosing 'α' or 'β' for Iac where it is most crucial that the operations at the message nodes are optimal, i.e. where the tunnel between the EXITS curves is most narrow. For regular LDPC codes, where the curves touch each other only at a single point in general, these results in a scaling factor or an offset term that can be kept constant over the iterations

For a regular LDPC code with dv = 3 and dc = 6, we find that the EXIT functions have their narrowest gap at Iac = 0.76 as shown in Fig-3.

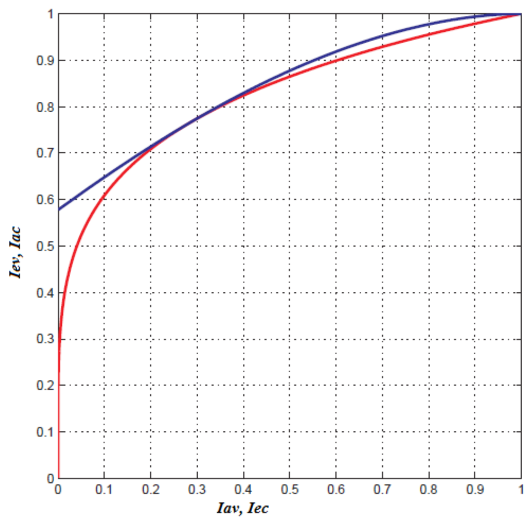


Fig-3: Constant processing for critical point

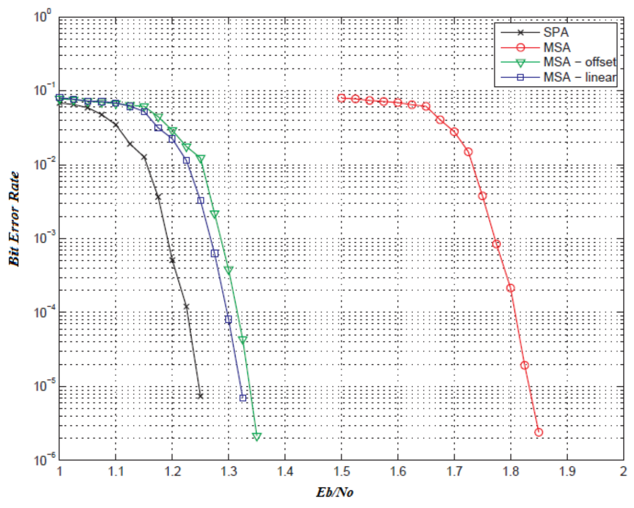


Fig-4: Offset and Linear processing with MSA

3. Simulation Results

The bit error rate simulation of a regular LDPC code with dv = 3 and dc = 6 is shown in Fig-4. The code used has a block length of N = 105 and a rate R = 0.5 and was decoded using the sum-product, the min-sum algorithm, constant linear post-processing (black) and constant offset post-processing (green).

At a bit error rate of 10−4, the min-sum decoder has a gap to the sum-product decoder of approximately 0.6dB. Using constant linear post-processing or constant offset post-processing, the gap can be lowered to approximately 0.1dB. Offset processing is preferred in hardware as the implementation is easy for subtractions than multiplications.

VI. IRREGULAR LDPC CODES

For Irregular codes [5], we apply a Gaussian approximation since Iac is not the same for every channel; it now denotes the average mutual information with respect to the variable node degree distribution (edge perspective). Constant post-processing function results in a trade-off between good decoding threshold and low error floor. If the code is not check-regular, there exists an individual post-processing function (or one of its approximations) for every check node degree.

We used an irregular LDPC code that checks regularly with dc = 9 and the nonzero coefficients of the variable node distribution. The resulting code rate is R = 0.5. For the code construction, we set the block length to N = 105 in order to minimize the effects that are caused by cycles in the graph.

1. The sequence of Correction Terms:

Since the EXIT functions of variable and check node decoder touch each other almost everywhere if the code is designed in order to achieve capacity, it is not possible to apply a constant post-processing function for every iteration. One way to overcome this is to vary the post-processing function over the number of iterations resulting in a sequence of scaling factors or a sequence of offset values. These sequences can be found by predicting the trajectory in the EXIT chart just above the threshold and computing a correction term for every iteration. The correction terms can then be obtained using density evolution, or by using the analytical derivation as per (8) and (9). The sequence of scaling factors for 100 iterations as shown in Fig-5. It can be observed, that this sequence is increasing and it converges to 1.0, i.e. no post-processing, which is due to the fact that the decoder is converging to Iac = 1.0 where post-processing becomes an identity function

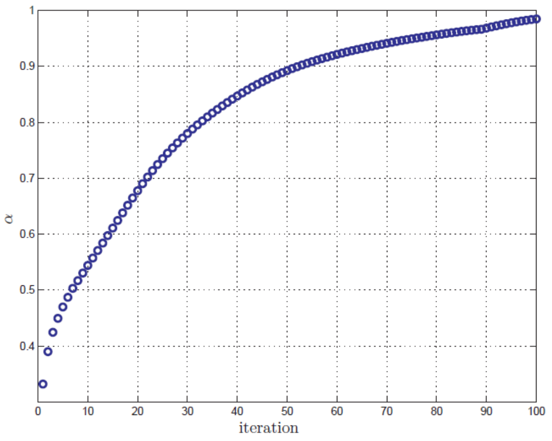


Fig-5: Sequences of scaling factors for irregular codes based on iterations

2. Constant Nonlinear Post-Processing:

When implementing an LDPC decoder in hardware, a scaling factor can easily be applied by using a lookup table and thus avoiding the need for a multiplication. However, the application of a sequence of scaling factors would require either a new lookup table for every iteration or a multiplication unit. Both options are not convenient for small and fast architectures. Therefore, we aim to apply a post- processing function that can be kept constant over the iterations without loss in performance, i.e. degradation of the threshold.

We propose to apply a universal, nonlinear post-processing function that is derived from a combination of post-processing functions weighted with the probability density function of the output of the min-sum approximation. We use a heuristic method to derive this universal post-processing function, which is computed as the weighted average between the minimum and the maximum Iac at the check node input as Eq.10

$$f_u(z;d_c) = \int_{I_{ac,min}}^{I_{ac,max}} f(z;d_c,I_{ac}) \cdot p(Z = z;d_c,I_{ac})\, d\, I_{ac} \qquad \text{...} (10)$$

In Figure 4.9, we show the universal post-processing function obtained using (10), where we set *Iac, min = 0.517* (which is the threshold of the example code) and *Iac,max = 1.0.* For comparison, we also show the post-processing functions for Iac at 0.517 and 1.0. It can be observed that for small magnitudes of Z , the universal function is close to the function for Iac = 0.517 and for large magnitudes the universal function gets close to the post-processing function for Iac = 1.0 (which is the identity function).

3. Simulation Results

Fig-7 shows bit error rate simulations for the irregular example code, using a maximum of 100 iterations. We used post-processing with varying scaling factors shown in Fig-5, that were obtained and constant nonlinear post-processing. For comparison, we included the curves for the sum-product algorithm, the min-sum algorithm without post-processing and the min-sum algorithm with constant scaling factors. It can be observed that the universal post-processing function has the smallest gap to the optimal sum-product algorithm and that the performance is very close to that of varying scaling factors. Compared to the min-sum algorithm without post-processing, we notice an improvement of approximately 1.0dB. For the case of constant scaling factors (as in the case of regular LDPC codes), we observe a trade-off between error floor and decoding threshold. Small scaling factors lead to a good decoding threshold but suffer from a high error floor. This is because small scaling factors cause the decoder to get stuck after a few iterations and therefore, the trajectory does not reach the top right corner of the EXIT chart. On the other hand, using large scaling factors causes the decoder to perform sub-optimal in the first iterations, leading to a higher decoding threshold.

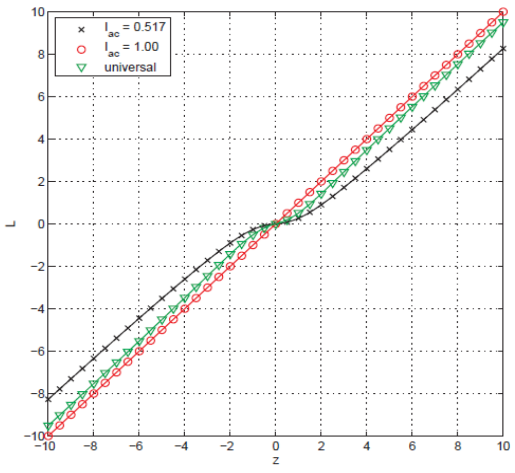


Fig-6: Post processing for dc = 9

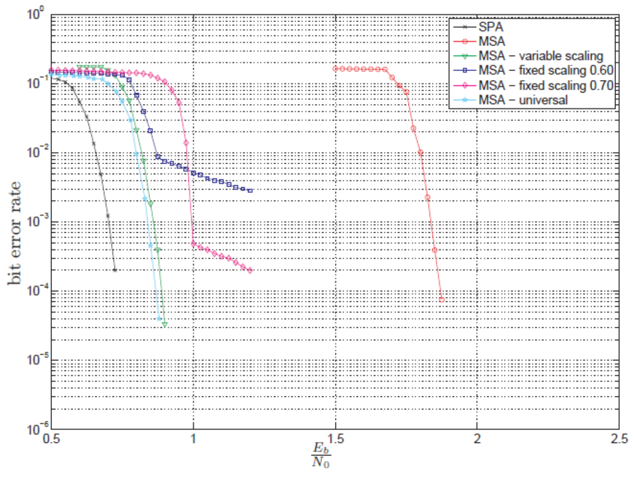


Fig-7: BER with and without post-processing

CONCLUSIONS

We have Analysed that (showed how) the min-sum algorithm for decoding LDPC codes can be improved by means of post-processing, by re-interpreting the minimum as an observation and computing the a-posteriori L-value of the code digits given this observation.

The resulting post-processing function is nonlinear and has to be changed for every iteration in general. However, it is sufficient to approximate this post-processing function by either a linear or an offset function. Furthermore, we showed that for regular codes it is sufficient to optimize the post-processing function for a single point in the EXIT chart, allowing us to keep it constant for every iteration.

For irregular LDPC codes, we showed that using a constant approximation of the post-processing function results in a trade-off between good decoding threshold and low error floor. To achieve decoding with a good threshold and low error floor one has to use either a sequence of linear post-processing functions or nonlinear post- processing. Using these techniques decreases the gap between optimal sum-product decoding and sub-optimal min-sum decoding significantly.

REFERENCES

[1] "Some results on spatially coupled photograph LDPC codes" – Information theory and application workshop by SVS Ranganathan, Kasra vakilinia, Lara Dolecek, Dariush Divsalar.

[2] "Performance Comparison of LDPC Block and Spatially Coupled Codes over GF(q)" – IEEE transactions on communication Vol 63 issue 3 march 2015 by Kechao Huang, Lai Wei, Daniel J Costello

[3] "On the Block Error Rate Performance of Spatially Coupled LDPC Codes for Streaming Applications" – IEEE information theory workshop 2016 by David G. M. Mitchell, Ali E. Pusane, Michael Lentmaier, and Daniel J. Costello

[4] "Efficient Encoding of Low-Density Parity-Check Codes" - IEEE transactions on information theory, vol. 47, no. 2, February 2001 by Thomas J. Richardson and Rüdiger L. Urbanke

[5] "A Family of Irregular LDPC Codes With Low Encoding Complexity" - IEEE communications letters, vol. 7, no. 2, February 2003 by Sarah J. Johnson, Student Member, IEEE, and Steven R. Weller, Member, IEEE